External Memory Interface Handbook

Volume 1: Altera Memory Solution Overview and Design Flow

101 Innovation Drive
San Jose, CA 95134
www.altera.com

EMI_GS-2.0

Document last updated for Altera Complete Design Suite version: 12.1
Document publication date: November 2012

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Contents

Chapter Revision Dates iii

Section I. Overview

Chapter 1. Introduction to Altera Memory Solution

Soft and Hard Memory IP 1–1
Memory Solutions 1–2
Protocol Support Matrix 1–5
Low Latency 1–5
Efficiency 1–6
Document Revision History 1–7

Chapter 2. Recommended Design Flow

Design Flow 2–1
Design Checklist 2–3
Document Revision History 2–6

Chapter Revision Dates

The chapters in this document, External Memory Interface Handbook, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Introduction to Altera Memory Solution
Revised: *November 2012*
Part Number: *EMI_GS_001-2.0*

Chapter 2. Recommended Design Flow
Revised: *June 2012*
Part Number: *EMI_GS_002-3.0*

Section I. Overview

This section provides an overview of Altera memory solutions and the recommended memory IP design flow.

This section includes the following chapters:

- Chapter 1, Introduction to Altera Memory Solution
- Chapter 2, Recommended Design Flow

For information about the revision history for chapters in this section, refer to "Document Revision History" in each individual chapter.

1. Introduction to Altera Memory Solution

EMI_GS_001-2.0

This chapter describes the memory solutions that Altera provides.

Altera provides the fastest, most efficient, and lowest latency memory controllers. The controllers are designed to allow you to easily interface with today's higher speed memories.

Altera supports a wide variety of memory interfaces suitable for applications ranging from routers and switches to video cameras. You can easily implement Altera's intellectual property (IP) using the memory MegaCore functions through the Quartus II software. The Quartus II software also provides an external memory toolkit that helps you test the implementation of the IP in the FPGA device.

Refer to the External Memory Interface Spec Estimator page for the maximum speeds supported by Altera FPGAs.

Soft and Hard Memory IP

Altera's latest devices, the 28-nm FPGAs provide two types of memory solutions: soft memory IP and hard memory IP. Arria V (GX, GT, SX, and ST) and Cyclone V devices offer both soft and hard memory IP, while Stratix V and Arria V GZ devices offer only soft memory IP.

The soft memory IP gives you the flexibility to design your own interfaces to meet your system requirements and still benefit from the industry leading performance. The hard memory IP is designed to give you a complete out-of-the-box experience when designing a memory controller.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

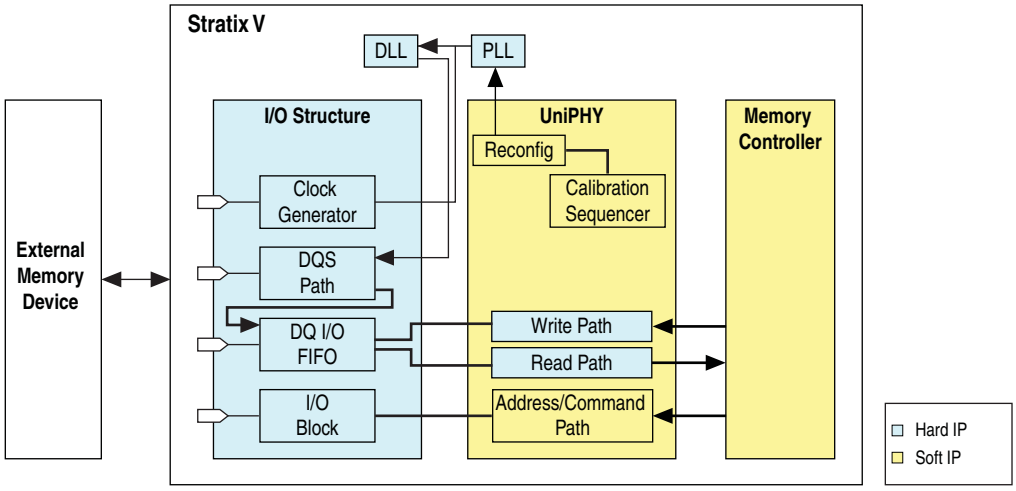
Table 1–1 lists the features of the soft and hard memory IP.

Table 1–1. Features of the Soft and Hard Memory IP

Soft Memory IP	Hard Memory IP
■ Consists of a DDR2, DDR3, or LPDDR2 SDRAM high-performance memory controller with UniPHY IP. ■ Has hardened read and write data paths to ensure your design meets timing at the highest speeds. The data paths include I/O, phase-locked loops (PLLs), delay-locked loop (DLL), and read and write FIFO buffers. ■ Allows you to choose the location to place the memory controller and the ability to size the memory controller based on the system requirements, especially in the Stratix V and Arria V GZ devices.	■ Consists of a DDR2 DDR3, or LPDDR2 SDRAM high-performance memory controller with a hard UniPHY IP, and a multi port front-end block. ■ Has a fixed location on the die and a fixed maximum width; ×32 and ×40 for Arria V devices and ×16 and ×40 for Cyclone V devices. ■ Runs at full rate to allow decreased latency and to minimize the required bus width of signals going into the core of the device. ■ Simplifies the overall memory design in Arria V and Cyclone V devices, and provides a truly out-of-the-box experience for every designer.

Figure 1–1 shows the hardened data paths in the soft memory IP of a Stratix V device.

Figure 1–1. Hardened Data Paths in the Soft Memory IP



Memory Solutions

Altera FPGAs achieve optimal memory interface performance with external memory IP. The IP provides the following components:

- Physical layer interface (PHY) which handles the timing on the data path itself.
- Memory controller block which implements all the memory commands and addresses.
- Multi port front-end (MPFE) block which allows multiple processes inside the FPGA device to share a common bank of memory. The MPFE block is a new feature in Arria V and Cyclone V devices.

These MPFE blocks are critical to the design and the use of the memory interface block.

Altera provides modular memory solutions that allow you to customize your memory interface design to any of the following configurations:

- PHY with your own controller
- PHY with Altera controller
- PHY with Altera controller and the MPFE block

You can also build a custom PHY, a custom controller, or both, as desired.

Table 1–2 lists the recommended memory types and controllers that Altera offers with the PHY IP.

Table 1–2. Altera Memory Types, PHY, and Controllers in the Quartus II Software (Part 1 of 2)

Quartus II Version	Memory	PHY IP	Controller IP
12.1	DDR/DDR2/DDR3	ALTMEMPHY (AFI) (1)	HPC II
	DDR2/DDR3//LPDDR2	UniPHY	HPC II
	QDR II/QDR II+	UniPHY	QDR/RLD II controller
	RLDRAM 3	UniPHY (4)	Custom (5)
	RLDRAM II	UniPHY	QDR/RLD II controller
	Other	ALTDQ_DQS (2)	Custom (5)
	Other	ALTDQ_DQS2 (3)	Custom (5)
12.0	DDR/DDR2/DDR3	ALTMEMPHY (AFI) (1)	HPC II
	DDR2/DDR3//LPDDR2	UniPHY	HPC II
	QDR II/QDR II+	UniPHY	QDR/RLD II controller
	RLDRAM II	UniPHY	QDR/RLD II controller
	Other	ALTDQ_DQS (2)	Custom
	Other	ALTDQ_DQS2 (3)	Custom
11.1	DDR/DDR2/DDR3	ALTMEMPHY (AFI) (1)	HPC II
	DDR2/DDR3	UniPHY	HPC II
	QDR II/QDR II+	UniPHY	QDR/RLD II controller
	RLDRAM II	UniPHY	QDR/RLD II controller
	Other	ALTDQ_DQS (2)	Custom
	Other	ALTDQ_DQS2 (3)	Custom
11.0	DDR/DDR2/DDR3	ALTMEMPHY (AFI)	HPC II
	DDR2/DDR3	UniPHY	HPC II
	QDR II/QDR II+	UniPHY	QDR/RLD II controller
	RLDRAM II	UniPHY	QDR/RLD II controller
	Other	ALTDQ_DQS (2)	Custom
	Other	ALTDQ_DQS2 (3)	Custom

Table 1–2. Altera Memory Types, PHY, and Controllers in the Quartus II Software (Part 2 of 2)

Quartus II Version	Memory	PHY IP	Controller IP
10.1	DDR/DDR2/DDR3	ALTMEMPHY (AFI)	HPC HPC II
	DDR2/DDR3	UniPHY Nios-based Sequencer	HPC II
	QDR II/QDR II+	UniPHY RTL Sequencer	QDR/RLD II controller
	RLDRAM II	UniPHY RTL Sequencer	QDR/RLD II controller
	Other	ALTDQ_DQS (2)	Custom
	Other	ALTDQ_DQS2 (3)	Custom
10.0	DDR/DDR2/DDR3	ALTMEMPHY (AFI)	HPC HPC II
	DDR2/DDR3	UniPHY Nios-based Sequencer	HPC II
	QDR II/QDR II+	UniPHY RTL Sequencer	QDR/RLD II controller
	RLDRAM II	UniPHY RTL Sequencer	QDR/RLD II controller
	Other	ALTDQ_DQS (2)	Custom
	Other	ALTDQ_DQS2 (3)	Custom
9.1	DDR/DDR2/DDR3	ALTMEMPHY (AFI)	HPC HPC II
	QDR II/QDR II+	UniPHY	QDR II controller
	RLDRAM II	UniPHY	RLDRAM II controller
	Other	ALTDQ_DQS (2)	Custom

Notes to Table 1–2:

(1) AFI = Altera PHY interface
(2) Applicable for Arria II, Stratix III, and Stratix IV devices.
(3) Applicable only for Cyclone V, Arria V and Stratix V devices.
(4) Applicable only for Arria V GZ and Stratix V devices.
(5) No Altera controller support. You must create your own controllers.

For more information about the controllers with the UniPHY or the ALTMEMPHY IP, refer to the *Functional Descriptions* section in Volume 3 of the *External Memory Interface Handbook*.
For more information about the ALTDQ_DQS megafunction, refer to the *ALTDLL and ALTDQ_DQS Megafunctions User Guide*.
For more information about the ALTDQ_DQS2 megafunction, refer to the *ALTDQ_DQS2 Megafunction User Guide*.

Protocol Support Matrix

Table 1–3 lists the device family and IP architecture support for each memory protocol.

Table 1–3. Protocol Support Matrix *(1) (2) (3)*

Protocol	Family										IP Architecture				
	Stratix V/ Arria V GZ	Arria V GX, GT, SX, ST	Cyclone V	Stratix IV	Stratix III	Arria II GZ	Arria II GX	Cyclone III	Cyclone IV	HardCopy III/IV	Hard/Soft	Rate	Burst Length	Sequencer	Controller
DDR3	—	U	U	—	—	—	—	—	—	—	Hard	Full	8	Nios II	HPC II
	U	U	U	U	U	U	A	—	—	U	Soft	Half	8	Nios II	HPC II
	U	U	—	—	—	—	—	—	—	—	Soft	Quarter	8	Nios II	HPC II
DDR2	—	U	U	—	—	—	—	—	—	—	Hard	Full	4,8	Nios II	HPC II
	U	—	—	U	U	U	A	A	A	U	Soft	Full	4,8	Nios II	HPC II
	U	U	U	U	U	U	A	A	A	U	Soft	Half	4,8	Nios II	HPC II
LPDDR2	—	U	U	—	—	—	—	—	—	—	Soft	Half	4,8,16	Nios II	HPC II
	—	—	U	—	—	—	—	—	—	—	Hard	Full	4,8,16	Nios II	HPC II
RLDRAM 3	U	—	—	—	—	—	—	—	—	—	Soft	Half	2,4,8	Nios II	—
	U	—	—	—	—	—	—	—	—	—	Soft	Quarter	2,4,8	Nios II	—
RLDRAM II	U	—	—	U	U	U	—	—	—	U	Soft	Full	2,4,8	RTL	RLDRAM II
	U	U	—	U	U	U	—	—	—	U	Soft	Half	4,8	Nios II	RLDRAM II
	U	U	—	U	U	U	—	—	—	U	Soft	Half	4,8	RTL	RLDRAM II
QDR II/II+	U	—	—	U	U	U	U	—	—	U	Soft	Full	2,4	RTL	QDR II/II+
	U	U	—	U	U	U	—	—	—	U	Soft	Half	4	Nios II	QDR II/II+
	U	U	—	U	U	U	U	—	—	U	Soft	Half	4	RTL	QDR II/II+

Notes to Table 1–3:

(1) **U**= Supported by UniPHY-based IP.

(2) **A** = Supported by ALTMEMPHY-based IP.

(3) **--** = Not supported.

Low Latency

Altera generally offers low latency solutions that are significantly better than Altera's competitors. Altera's 28-nm FPGA devices have a balanced clocked network in the periphery to reduce switching noise. The hardened read data FIFO buffer guarantees timing and makes it easier for the fitter to place the controller. Together with the latest UniPHY IP, these design changes provide major reduction in latency.

Table 1–4 lists latency comparison for Altera and its closest competition.

Table 1–4. Latency Comparison for Quarter-Rate DDR3 SDRAM Controllers

Latency Type	Latency (Memory Clock Cycles)		Advantage
	Competitor (1)	Altera	
Write Command	46	29	Altera
Read Command	46	29	Altera
Read Data	31	11	Altera

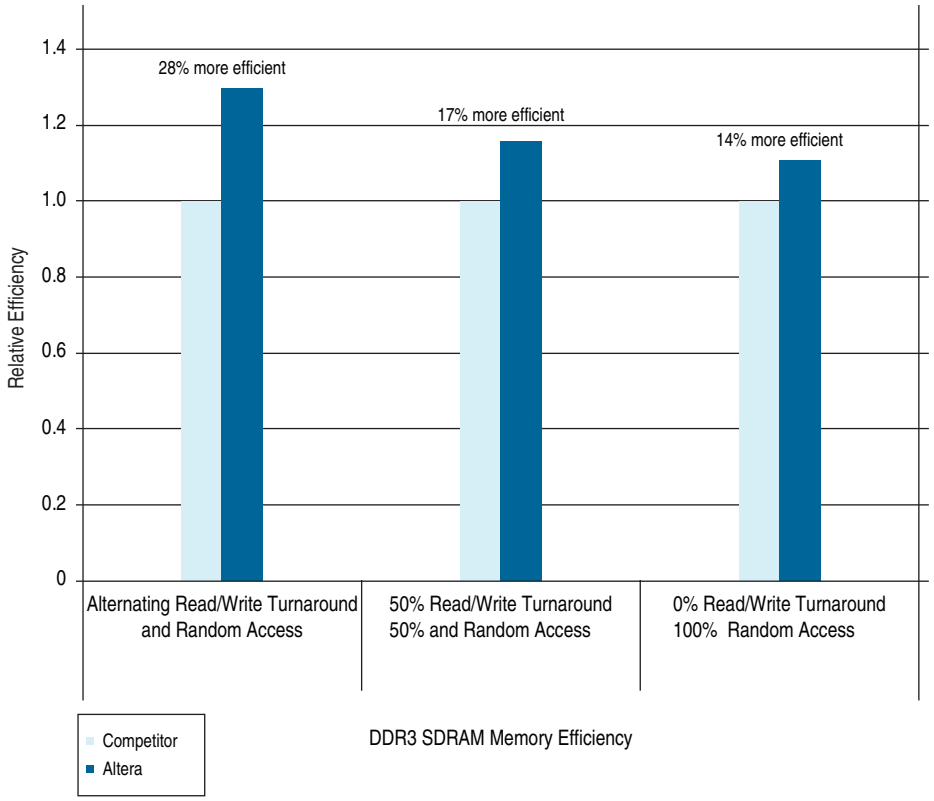
Note to Table 1–4:

(1) Does not include AXI latency.

Efficiency

Altera memory controllers are also highly efficient. Figure 1–2 shows the memory efficiency of a DDR3 SDRAM memory controller with UniPHY IP.

Figure 1–2. Memory Efficiency of DDR3 SDRAM Memory Controllers with UniPHY



Document Revision History

Table 1–5 lists the revision history for this document.

Table 1–5. Document Revision History

Date	Version	Changes
November 2012	2.0	■ Added Arria V GZ information. ■ Added RLDRAM III information to "Protocol Support Matrix" and "Memory Solutions".
June 2012	1.2	Change to Table 1–3.
June 2012	1.1	■ Added "Protocol Support Matrix". ■ Added Feedback icon.
November 2011	1.0	Initial release.

2. Recommended Design Flow

EMI_GS_002-3.0

This chapter describes the Altera-recommended design flow for successfully implementing external memory interfaces in Altera® devices. Altera recommends that you create an example top-level file with the desired pin outs and all interface IP instantiated, which enables the Quartus® II software to validate your design and resource allocation before PCB and schematic sign off. Use the "Design Checklist" on page 2–3, to verify whether you have performed all the recommended steps in creating a working and robust external memory interface.

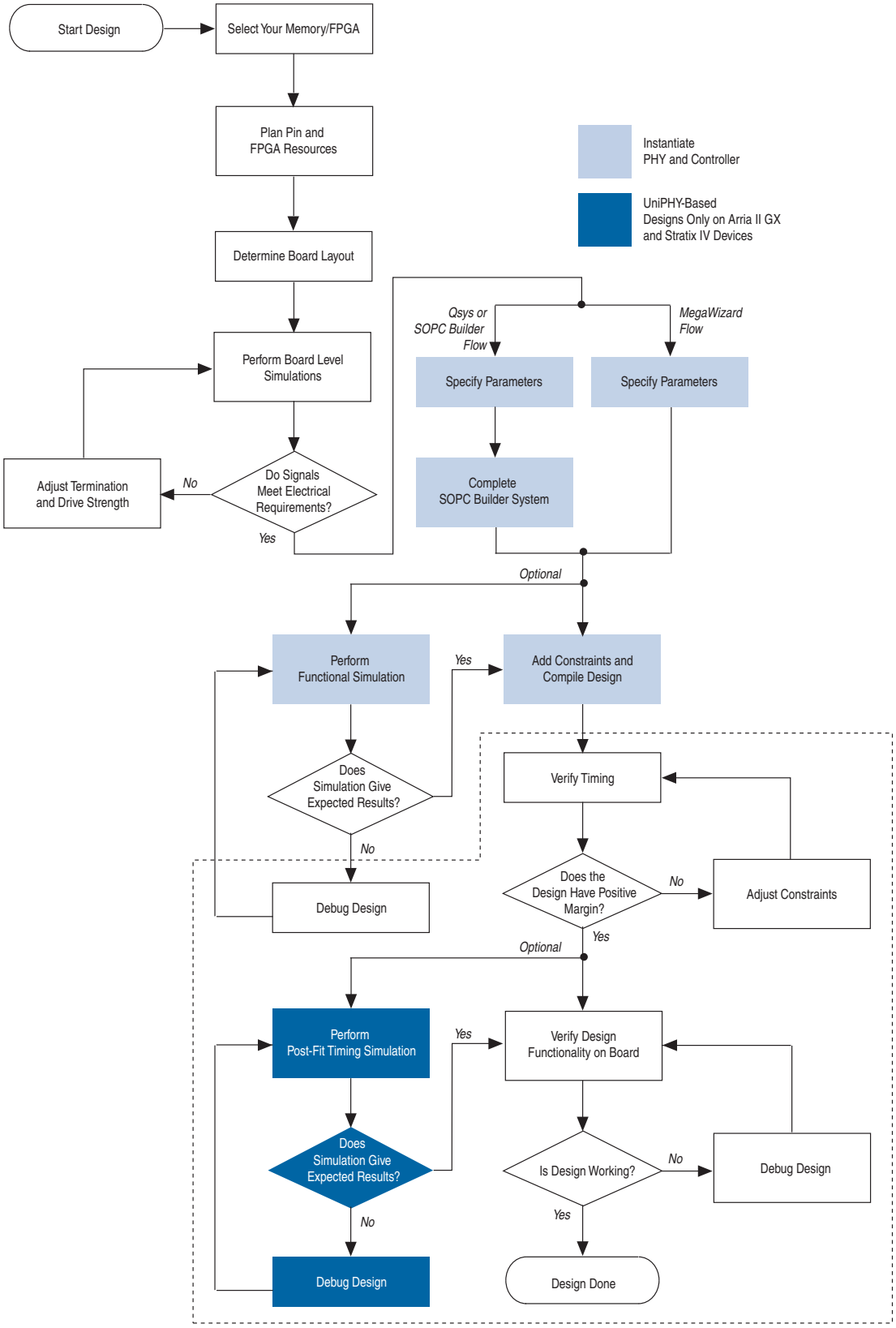
Design Flow

Figure 2–1 shows the design flow to provide the fastest out-of-the-box experience with external memory interfaces in Altera devices. This design flow assumes that you are using Altera IP to implement the external memory interface.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure 2–1. External Memory Interfaces Design Flowchart



Design Checklist

Use this design checklist when implementing external memory interfaces in Altera devices.

	Done	Action	Reference
		Select Your Memory	
1.	☐	Select the memory interface frequency of operation and bus width.	■ *Selecting Your Memory* chapter in the *External Memory Interface Handbook*.
		Select Your FPGA	
2.	☐	Select the FPGA device density and package combination that you want to target.	■ *Selecting Your FPGA* chapter in the *External Memory Interface Handbook*.
		Plan Pin and FPGA Resources	
3.	☐	Ensure that the target FPGA device supports the desired clock rate and memory bus width. Also the FPGA must have sufficient I/O pins for the DQ/DQS read and write groups.	For detailed device resource information, refer to the relevant device handbook chapter on external memory interface support.
		Determine Board Layout	
4.	☐	Select the termination scheme and drive strength settings for all the memory interface signals on the memory side and the FPGA side.	■ *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter in the *External Memory Interface Handbook*.
5.	☐	Ensure you apply appropriate termination and drive strength settings on all the memory interface signals, and verify using board level simulations.	■ *Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines* chapter in the *External Memory Interface Handbook*.
6.	☐	Use board level simulations to pick the optimal setting for best signal integrity. On the memory side, Altera recommends the use of external parallel termination on input signals to the memory (write data, address, command, and clock signals).	■ *RLDRAM II Board Design Guidelines* chapter in the *External Memory Interface Handbook*. ■ *QDR II SRAM Board Design Guidelines* chapter in the *External Memory Interface Handbook*.
7.	☐	Perform board level simulations, to ensure electrical and timing margins for your memory interface	
8.	☐	Ensure you have a sufficient eye opening using simulations. Use the latest FPGA and memory IBIS models, board trace characteristics, drive strength, and termination settings in your simulation. Any timing uncertainties at the board level that you calculate using simulations must be used to adjust the input timing constraints to ensure the accuracy of Quartus II timing margin reports. For example crosstalk, ISI, and slew rate deration.	
		Parameterize and Implement the Memory IP	
9.	☐	Parameterize and instantiate the Altera external memory IP for your target memory interface.	■ *Implementing and Parameterizing Memory IP* chapter in the *External Memory Interface Handbook*

	Done	Action	Reference
10.	☐	Ensure that you perform the following actions: ■ Pick the correct memory interface data rates, width, and configurations. ■ For DDR, DDR2, and DDR3 SDRAM interfaces, ensure that you derate the tIS, tIH, tDS, and tDH parameters, as necessary. ■ Include the board skew parameters for your board.	
11.	☐	Connect the PHY's local signals to your driver logic and the PHY's memory interface signals to top-level pins. Ensure that the local interface signals of the PHY are appropriately connected to your own logic. If the ALTMEMPHY IP is compiled without these local interface connections, you may encounter compilation problems, when the number of signals exceeds the pins available on your target device. You may also use the example top-level file as an example on how to connect your own custom controller to the Altera memory PHY.	■ *Functional Description: HPC II* chapter in the *External Memory Interface Handbook*. ■ *Functional Description: QDR II and QDR II+ SRAM Controller* chapter in the *External Memory Interface Handbook*. ■ *Functional Description: RLDRAM II Controller* chapter in the *External Memory Interface Handbook*.
		Perform Functional Simulation	
12.	☐	Simulate your design using the RTL functional model. Use the IP functional simulation model with your own driver logic, testbench, and a memory model, to ensure correct read and write transactions to the memory. You may need to prepare the memory functional model by setting the speed grade and device bus mode.	■ *Simulating Memory IP* chapter in the *External Memory Interface Handbook*
		Add Constraints	
13.	☐	Add timing constraints. The wizard-generated **.sdc** file adds timing constraints to the interface. However, you may need to adjust these settings to best fit your memory interface configuration.	
14.	☐	Add pin settings and DQ group assignments. The wizard-generated **.tcl** file includes I/O standard and pin loading constraints to your design.	
15.	☐	Ensure that generic pin names used in the constraint scripts are modified to match your top-level pin names. The loading on memory interface pins is dependent on your board topology (memory components).	
16.	☐	Add pin location assignments. However, you need to assign the pin location assignments manually using the Pin Planner.	
17.	☐	Ensure that the example top-level file or your top-level logic is set as top-level entity.	

	Done	Action	Reference
18.	☐	Adjust optimization techniques, to ensure the remaining unconstrained paths are routed with the highest speed and efficiency: a. On the Assignments menu click **Settings**. b. Select **Analysis & Synthesis Settings**. c. Select **Speed** under **Optimization Technique**. d. Expand **Fitter Settings**. e. Turn on **Optimize Hold Timing** and select **All Paths**. f. Turn on **Optimize Fast Corner Timing**. g. Select **Standard Fit** under **Fitter Effort**.	
19.	☐	Provide board trace delay model. For accurate I/O timing analysis, you specify the board trace and loading information in the Quartus II software. This information should be derived and refined during your board development process of prelayout (line) simulation and finally post-layout (board) simulation. Provide the board trace information for the output and bidirectional pins through the board trace model in the Quartus II software.	
		Compile Design and Verify Timing	
20.	☐	Compile your design and verify timing closure using all available models.	
21.	☐	Run the wizard-generated <*variation_name*>**_report_timing.tcl** file, to generate a custom timing report for each of your IP instances. Run this process across all device timing models (slow 0° C, slow 85° C, fast 0° C).	
22.	☐	If there are timing violations, adjust your constraints to optimize timing	
23.	☐	As required, adjust PLL clock phase shift settings or appropriate timing and location assignments margins for the various timing paths within the IP.	■ *Analyzing Timing of Memory IP* chapter in the *External Memory Interface Handbook*
		Perform Post-Fit Timing Simulation	
24.	☐	Perform post-fit timing simulation to ensure that all the memory transactions meet the timing specifications with the vendor's memory model.	■ *Simulating Memory IP* chapter in the *External Memory Interface Handbook*.
		Verify Design Functionality	
25.	☐	Verify the functionality of your memory interface in the system	■ *Debugging Memory IP* chapter in the *External Memory Interface Handbook*

Document Revision History

Table 2–1 shows the revision history for this document.

Table 2–1. Document Revision History

Date	Version	Changes
June 2012	3.0	■ Removed overlapping information. ■ Addd Feedback icon.
November 2011	2.1	Updated the design flow and the design checklist.
July 2010	2.0	Updated for 10.0 release.
January 2010	1.1	■ Improved description for *Implementing Altera Memory Interface IP* chapter. ■ Added timing simulation to flow chart and to design checklist.
November 2009	1.0	First published.

External Memory Interface Handbook

Volume 2: Design Guidelines

101 Innovation Drive
San Jose, CA 95134
www.altera.com

EMI_DG-3.0

Document last updated for Altera Complete Design Suite version: 12.1
Document publication date: November 2012

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO
9001:2008
Registered

Contents

Chapter Revision Dates xi

Section I. Design Flow Guidelines

Chapter 1. Selecting Your Memory

Memory Overview 1–1
DDR, DDR2, and DDR3 SDRAM 1–2
DDR SDRAM 1–2
DDR2 SDRAM 1–3
DDR3 SDRAM 1–3
DDR, DDR2, and DDR3 SDRAM Comparison 1–4
QDR, QDR II, and QDR II+ SRAM 1–5
RLDRAM II and RLDRAM 3 1–6
LPDDR2 1–7
Memory Selection 1–7
High-Speed Memory in Embedded Processor Example 1–9
High-Speed Memory in Telecom Example 1–11
Document Revision History 1–13

Chapter 2. Selecting Your FPGA Device

Device Family Selection 2–1
Cost 2–1
Memory Standards 2–2
I/O Interfaces 2–2
Wraparound Interfaces 2–3
Read and Write Leveling 2–3
Dynamic OCT 2–4
Device Settings Selection 2–4
Speed Grade 2–4
Operating Temperature 2–4
Package Size 2–4
Device Density and I/O Pin Counts 2–5
Device Density 2–5
I/O Pin Counts 2–5
Document Revision History 2–6

Chapter 3. Planning Pin and FPGA Resources

Interface Pins 3–1
DDR, DDR2, and DDR3 SDRAM 3–4
Clock Signals 3–4
Command and Address Signals 3–5
Data, Data Strobes, DM, and Optional ECC Signals 3–6
DIMM Options 3–7
QDR II+ and QDR II SRAM 3–9
Clock Signals 3–9
Command Signals 3–10
Address Signals 3–10
Data and QVLD Signals 3–10

RLDRAM II and RLDRAM 3 . . . 3–11
Clock Signals . . . 3–11
Commands and Addresses . . . 3–12
Data, DM and QVLD Signals . . . 3–12
LPDDR2 . . . 3–13
Clock Signal . . . 3–13
Command and Address Signal . . . 3–13
Data, Data Strobe, and DM Signals . . . 3–14
Maximum Number of Interfaces . . . 3–14
OCT Support for Arria II GX, Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices . . . 3–25
General Pin-out Guidelines . . . 3–26
Pin-out Rule Exceptions . . . 3–29
Exceptions for ×36 Emulated QDR II and QDR II+ SRAM Interfaces in Arria II, Stratix III and Stratix IV Devices . . . 3–29
Timing Impact on x36 Emulation . . . 3–32
Rules to Combine Groups . . . 3–32
Determining the CQ/CQn Arrival Time Skew . . . 3–33
Exceptions for RLDRAM II and RLDRAM 3 Interfaces . . . 3–34
Interfacing with ×9 RLDRAM II CIO Devices . . . 3–34
Interfacing with ×18 RLDRAM II and RLDRAM 3 CIO Devices . . . 3–35
Interfacing with RLDRAM II and RLDRAM 3 ×36 CIO Devices . . . 3–36
Exceptions for QDR II and QDR II+ SRAM Burst-length-of-two Interfaces . . . 3–37
Pin Connection Guidelines Tables . . . 3–37
Additional Placement Rules for Cyclone III and Cyclone IV Devices . . . 3–41
Additional Guidelines for Arria V GZ and Stratix V Devices . . . 3–46
Performing Manual Pin Placement . . . 3–46
Additional Guidelines for Arria V (Except Arria V GZ) Devices . . . 3–48
Performing Manual Pin Placement . . . 3–48
Additional Guidelines for Cyclone V Devices . . . 3–49
I/O Pins Connect to Ground for Hard Memory Interface Operation . . . 3–49
PLLs and Clock Networks . . . 3–50
Using PLL Guidelines . . . 3–56
PLL Cascading . . . 3–57
DLL . . . 3–57
Other FPGA Resources . . . 3–59
Document Revision History . . . 3–59

Chapter 4. DDR2 and DDR3 SDRAM Board Design Guidelines

Leveling and Dynamic ODT . . . 4–2
Read and Write Leveling . . . 4–3
Dynamic ODT . . . 4–5
Dynamic OCT in Stratix III and Stratix IV Devices . . . 4–5
Dynamic OCT in Stratix V Devices . . . 4–7
Board Termination for DDR2 SDRAM . . . 4–7
External Parallel Termination . . . 4–8
On-Chip Termination . . . 4–9
Recommended Termination Schemes . . . 4–10
Dynamic On-Chip Termination . . . 4–13
FPGA Writing to Memory . . . 4–14
FPGA Reading from Memory . . . 4–16
On-Chip Termination (Non-Dynamic) . . . 4–17
Class II External Parallel Termination . . . 4–20
FPGA Writing to Memory . . . 4–20

FPGA Reading from Memory 4–23
Class I External Parallel Termination 4–25
FPGA Writing to Memory 4–25
FPGA Reading from Memory 4–27
Class I Termination Using ODT 4–28
FPGA Writing to Memory 4–28
FPGA Reading from Memory 4–30
No-Parallel Termination 4–30
FPGA Writing to Memory 4–31
FPGA Reading from Memory 4–32
Board Termination for DDR3 SDRAM 4–35
Single-Rank DDR3 SDRAM Unbuffered DIMM 4–36
DQS, DQ, and DM for DDR3 SDRAM UDIMM 4–37
Memory Clocks for DDR3 SDRAM UDIMM 4–39
Commands and Addresses for DDR3 SDRAM UDIMM 4–41
Stratix III, Stratix IV, and Stratix V FPGAs 4–42
DQS, DQ, and DM for Stratix III, Stratix IV, and Stratix V FPGA 4–42
Memory Clocks for Stratix III, Stratix IV, and Stratix V FPGA 4–43
Commands and Addresses for Stratix III and Stratix IV FPGA 4–43
Multi-Rank DDR3 SDRAM Unbuffered DIMM 4–43
DDR3 SDRAM Registered DIMM 4–45
DDR3 SDRAM Load-Reduced DIMM 4–45
DDR3 SDRAM Components With Leveling 4–45
DDR3 SDRAM Components With or Without Leveling 4–45
Stratix III, Stratix IV, and Stratix V FPGAs 4–49
Drive Strength 4–50
How Strong is Strong Enough? 4–51
System Loading 4–52
Component Versus DIMM 4–52
FPGA Writing to Memory 4–52
FPGA Reading from Memory 4–54
Single- Versus Dual-Rank DIMM 4–55
Single DIMM Versus Multiple DIMMs 4–57
Design Layout Guidelines 4–57
Layout Guidelines for DDR2 SDRAM Interface 4–58
Layout Guidelines for DDR3 SDRAM Interface 4–61
Length Matching Rules 4–66
Layout Guidelines for DDR3 SDRAM Wide Interface (>72 bits) 4–68
Fly-By Network Design for Clock, Command, and Address Signals 4–68
Package Deskew 4–70
DQ/DQS/DM Deskew 4–70
Address and Command Deskew 4–71
Deskew Example 4–71
Package Migration 4–72
Document Revision History 4–73

Chapter 5. Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines

DDR2 SDRAM 5–1
Stratix II High Speed Board 5–2
Overview of ODT Control 5–3
DIMM Configuration 5–5
Dual-DIMM Memory Interface with Slot 1 Populated 5–5
FPGA Writing to Memory 5–5
Write to Memory Using an ODT Setting of 150Ω 5 5

Reading from Memory 5–7
Dual-DIMM with Slot 2 Populated 5–8
FPGA Writing to Memory 5–8
Write to Memory Using an ODT Setting of 150Ω 5 9
Reading from Memory 5–10
Dual-DIMM Memory Interface with Both Slot 1 and Slot 2 Populated 5–12
FPGA Writing to Memory 5–12
Write to Memory in Slot 1 Using an ODT Setting of 75-Ω 5 12
Write to Memory in Slot 2 Using an ODT Setting of 75-Ω 5 14
Reading From Memory 5–15
Dual-DIMM DDR2 Clock, Address, and Command Termination and Topology 5–20
Address and Command Signals 5–21
Control Group Signals 5–21
Clock Group Signals 5–21
DDR3 SDRAM 5–22
Comparison of DDR3 and DDR2 DQ and DQS ODT Features and Topology 5–22
Dual-DIMM DDR3 Clock, Address, and Command Termination and Topology 5–23
Address and Command Signals 5–23
Control Group Signals 5–23
Clock Group Signals 5–23
Write to Memory in Slot 1 Using an ODT Setting of 75 ΩWith One Slot Populated 5–24
Write to Memory in Slot 2 Using an ODT Setting of 75 ΩWith One Slot Populated 5–25
Write to Memory in Slot 1 Using an ODT Setting of 150 ΩWith Both Slots Populated 5–26
Write to Memory in Slot 2 Using an ODT Setting of 150 ΩWith Both Slots Populated 5–27
Read from Memory in Slot 1 Using an ODT Setting of 150 Ωon Slot 2 with Both Slots Populated . . 5–28
Read From Memory in Slot 2 Using an ODT Setting of 150 Ωon Slot 1 With Both Slots Populated . 5–29
FPGA OCT Features 5–30
Arria V, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices 5–30
Arria II GX Devices 5–30
Document Revision History 5–31

Chapter 6. LPDDR2 SDRAM Board Design Guidelines

I/O Standards 6–1
LPDDR2 SDRAM Configurations 6–2
Signal Terminations 6–4
Outputs from the FPGA to the LPDDR2 Component 6–5
Input to the FPGA from the LPDDR2 SDRAM Component 6–8
Termination Schemes 6–9
PCB Layout Guidelines 6–10
Document Revision History 6–13

Chapter 7. RLDRAM II and RLDRAM 3 Board Design Guidelines

I/O Standards 7–1
RLDRAM II Configurations 7–2
RLDRAM 3 Configurations 7–4
Signal Terminations 7–6
Input to the FPGA from the RLDRAM Components 7–7
Outputs from the FPGA to the RLDRAM II and RLDRAM 3 Components 7–7
RLDRAM II Termination Schemes 7–8
RLDRAM 3 Termination Schemes 7–9
PCB Layout Guidelines 7–9
RLDRAM II and RLDRAM 3 Layout Approach 7–11
Document Revision History 7–12

Chapter 8. QDR II SRAM Board Design Guidelines

I/O Standards . . . 8–1
QDR II SRAM Configurations . . . 8–2
Signal Terminations . . . 8–4
Output from the FPGA to the QDR II SRAM Component . . . 8–5
Input to the FPGA from the QDR II SRAM Component . . . 8–13
Termination Schemes . . . 8–17
PCB Layout Guidelines . . . 8–19
Document Revision History . . . 8–22

Chapter 9. Implementing and Parameterizing Memory IP

Installation and Licensing . . . 9–1
Free Evaluation . . . 9–2
OpenCore Plus Time-Out Behavior . . . 9–2
Design Flow . . . 9–3
MegaWizard Plug-In Manager Flow . . . 9–4
Specifying Parameters . . . 9–4
Constraining the Design . . . 9–6
Add Pins and DQ Group Assignments . . . 9–6
Compiling the Design . . . 9–7
SOPC Builder Flow . . . 9–8
Specifying Parameters . . . 9–8
Completing the SOPC Builder System . . . 9–9
Qsys System Integration Tool Design Flow . . . 9–10
Specify Parameters . . . 9–10
Complete the Qsys System . . . 9–11
Qsys and SOPC Builder Interfaces . . . 9–12
Generated Files . . . 9–37
Generated Files for Memory Controllers with the ALTMEMPHY IP . . . 9–38
Generated Files for Memory Controllers with the UniPHY IP . . . 9–42
Parameterizing Memory Controllers with ALTMEMPHY IP . . . 9–46
Memory Settings . . . 9–46
Show in 'Memory Preset' List . . . 9–47
Memory Presets . . . 9–47
Preset Editor Settings for DDR and DDR2 SDRAM . . . 9–48
Preset Editor Settings for DDR3 SDRAM . . . 9–52
Derating Memory Setup and Hold Timing . . . 9–58
PHY Settings . . . 9–60
Board Settings . . . 9–62
Controller Settings . . . 9–63
Parameterizing Memory Controllers with UniPHY IP . . . 9–65
PHY Settings . . . 9–65
Memory Parameters . . . 9–68
LPDDR2, DDR2 and DDR3 SDRAM . . . 9–68
QDR II and QDR II+ SRAM . . . 9–71
RLDRAM II . . . 9–71
Memory Timing . . . 9–73
Board Settings . . . 9–78
Setup and Hold Derating . . . 9–78
Intersymbol Interference . . . 9–81
Board Skews . . . 9–81
Controller Settings . . . 9–90
Diagnostics . . . 9–94

Document Revision History 9–95

Chapter 10. Simulating Memory IP

Memory Simulation Models 10–1
Simulation Options 10–1
Simulation Walkthrough with UniPHY IP 10–3
Simulation Scripts 10–3
Preparing the Vendor Memory Model 10–4
Functional Simulations 10–7
Verilog HDL 10–7
VHDL 10–8
Simulating the Example Design 10–9
Abstract PHY 10–10
PHY-Only Simulation 10–11
Post-fit Functional Simulation 10–12
Simulation Issues 10–14
Simulation Walkthrough with ALTMEMPHY IP 10–16
Before Simulating 10–17
Preparing the Vendor Memory Model 10–18
Simulating Using NativeLink 10–20
IP Functional Simulations 10–21
VHDL 10–21
Verilog HDL 10–23
Simulation Tips and Issues 10–24
Tips 10–24
DDR3 SDRAM (without Leveling) Warnings and Errors 10–25
Document Revision History 10–26

Chapter 11. Analyzing Timing of Memory IP

Memory Interface Timing Components 11–2
Source-Synchronous Paths 11–2
Calibrated Paths 11–3
Internal FPGA Timing Paths 11–3
Other FPGA Timing Parameters 11–3
FPGA Timing Paths 11–4
Arria II Device PHY Timing Paths 11–4
Stratix III and Stratix IV PHY Timing Paths 11–6
Arria V, Cyclone V, and Stratix V Timing paths 11–8
Cyclone III and Cyclone IV PHY Timing Paths 11–9
Timing Constraint and Report Files 11–10
ALTMEMPHY Megafunction 11–10
UniPHY IP 11–12
Timing Analysis Description 11–13
Address and Command 11–14
PHY or Core 11–14
PHY or Core Reset 11–14
Read Capture and Write 11–14
Cyclone III and Stratix III 11–14
Arria II, Arria V, Cyclone IV, Cyclone V, Stratix IV and Stratix V 11–21
Read Resynchronization 11–22
Mimic Path 11–22
DQS versus CK—Arria II GX, Cyclone III, and Cyclone IV Devices 11–22
Write Leveling t_{DQSS} 11–23

Write Leveling t_{DSH}/t_{DSS} . . . 11–23
DK versus CK (RLDRAM II with UniPHY) . . . 11–23
Bus Turnaround Time . . . 11–23
Timing Report DDR . . . 11–24
Report SDC . . . 11–27
Calibration Effect in Timing Analysis . . . 11–28
Calibration Emulation for Calibrated Path . . . 11–28
Calibration Error or Quantization Error . . . 11–28
Calibration Uncertainties . . . 11–28
Memory Calibration . . . 11–28
Timing Model Assumptions and Design Rules . . . 11–29
Memory Clock Output Assumptions . . . 11–30
Cyclone III Devices . . . 11–31
Stratix III Devices . . . 11–31
Write Data Assumptions . . . 11–32
Cyclone III Devices . . . 11–33
Stratix III Devices . . . 11–33
Read Data Assumptions . . . 11–35
Cyclone III Devices . . . 11–35
Stratix III Devices . . . 11–36
Mimic Path Assumptions . . . 11–36
DLL Assumptions . . . 11–36
PLL and Clock Network Assumptions . . . 11–37
Stratix III Devices . . . 11–37
Cyclone III Devices . . . 11–37
Timing Closure . . . 11–38
Common Issues . . . 11–38
Missing Timing Margin Report . . . 11–38
Incomplete Timing Margin Report . . . 11–38
Read Capture Timing . . . 11–38
Write Timing . . . 11–39
Address and Command Timing . . . 11–39
PHY Reset Recovery and Removal . . . 11–40
Clock-to-Strobe (for DDR and DDR2 SDRAM Only) . . . 11–40
Read Resynchronization and Write Leveling Timing (for SDRAM Only) . . . 11–40
Optimizing Timing . . . 11–41
Timing Deration Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs . . . 11–43
Multiple Chip Select Configuration Effects . . . 11–43
ISI Effects . . . 11–44
Calibration Effects . . . 11–44
Board Effects . . . 11–45
Timing Deration using the Board Settings . . . 11–45
Slew Rates . . . 11–45
Intersymbol Interference . . . 11–48
Board Skews . . . 11–49
Timing Deration Using the Excel-Based Calculator . . . 11–52
Before You Use the Excel-based Calculator for Timing Deration . . . 11–52
Using the Excel-Based Calculator . . . 11–52
Using the Excel-based Calculator for Timing Deration (Without Board Trace Models) . . . 11–55
Performing I/O Timing Analysis . . . 11–56
Perform I/O Timing Analysis with 3rd Party Simulation Tools . . . 11–56
Perform Advanced I/O Timing Analysis with Board Trace Delay Model . . . 11–56
Document Revision History . . . 11–57

Chapter 12. Debugging Memory IP

Memory IP Debugging Issues . . . 12–1
Resource and Planning Issues . . . 12–2
Resource Issue Evaluation . . . 12–2
Dedicated IOE DQS Group Resources and Pins . . . 12–2
Dedicated DLL Resources . . . 12–3
Specific PLL Resources . . . 12–3
Specific Global, Regional and Dual-Regional Clock Net Resources . . . 12–3
Planning Issue Evaluation . . . 12–4
Interface Configuration Issues . . . 12–4
Performance Issues . . . 12–4
Bottleneck and Efficiency Issues . . . 12–5
Functional Issues . . . 12–6
Functional Issue Evaluation . . . 12–6
Correct Combination of the Quartus II Software and ModelSim-Altera Device Models . . . 12–6
Altera IP Memory Model . . . 12–7
Vendor Memory Model . . . 12–7
Out of PC Memory Issues . . . 12–7
Transcript Window Messages . . . 12–8
Simulation . . . 12–9
Modifying the Example Driver to Replicate the Failure . . . 12–9
Timing Issues . . . 12–10
Timing Issue Characteristics . . . 12–10
Timing Issue Evaluation . . . 12–11
FPGA Timing Issues . . . 12–11
External Memory Interface Timing Issues . . . 12–12
Verifying Memory IP Using the SignalTap II Logic Analyzer . . . 12–13
Monitoring Signals with the SignalTap II Logic Analyzer . . . 12–15
DDR, DDR2, and DDR3 ALTMEMPHY Designs . . . 12–15
UniPHY Designs . . . 12–17
Hardware Debugging Guidelines . . . 12–18
Create a Simplified Design that Demonstrates the Same Issue . . . 12–18
Measure Power Distribution Network . . . 12–18
Measure Signal Integrity and Setup and Hold Margin . . . 12–18
Vary Voltage . . . 12–18
Use Freezer Spray and Heat Gun . . . 12–19
Operate at a Lower Speed . . . 12–19
Find Out if the Issue Exists in Previous Versions of Software . . . 12–19
Find out if the Issue Exists in the Current Version of Software . . . 12–19
Try A Different PCB . . . 12–20
Try Other Configurations . . . 12–20
Debugging Checklist . . . 12–20
Catagorizing Hardware Issues . . . 12–21
Signal Integrity Issues . . . 12–21
Characteristics . . . 12–21
Evaluating SignaI Integrity Issues . . . 12–22
Hardware and Calibration Issues . . . 12–24
Hardware and Calibration Issue Characteristics . . . 12–24
Evaluating Hardware and Calibration Issues . . . 12–24
Intermittent Issues . . . 12–27
Intermittent Issue Evaluation . . . 12–27
Debug Toolkit . . . 12–28
ALTMEMPHY Debug Toolkit Overview and Usage Flow . . . 12–28
UniPHY EMIF Debug Toolkit Overview and Usage Flow . . . 12–29

Document Revision History 12–29

Section II. Miscellaneous Guidelines

Chapter 13. HardCopy Design Migration Guidelines

HardCopy Migration Design Guidelines 13–1
Differences in UniPHY IP Generated with HardCopy Migration Support 13–3
ROM Loader for Designs Using Nios II Sequencer 13–3
Passive Serial (PS) Configuration Scheme 13–5
Active Serial (AS) Configuration Scheme 13–5
Fast Passive Parallel (FPP) Configuration Scheme 13–5
PLL/DLL Run-time Reconfiguration 13–6
Document Revision History 13–8

Chapter 14. Optimizing the Controller

Controller Efficiency 14–1
Factors Affecting Efficiency 14–2
Interface Standard 14–2
Bank Management Efficiency 14–2
Data Transfer 14–5
Ways to Improve Efficiency 14–6
DDR2 SDRAM Controller 14–6
Auto-Precharge Commands 14–6
Additive Latency 14–9
Bank Interleaving 14–10
Command Queue Look-Ahead Depth 14–12
Additive Latency and Bank Interleaving 14–13
User-Controlled Refresh 14–15
Frequency of Operation 14–15
Burst Length 14–16
Series of Reads or Writes 14–16
Reordering 14–17
Data Reordering 14–17
Starvation Control 14–18
Command Reordering 14–18
Bandwidth 14–20
Efficiency Monitor 14–21
Document Revision History 14–22

Chapter 15. PHY Considerations

Core Logic and User Interface Data Rate 15–1
Hard and Soft Memory PHY 15–2
Sequencer 15–3
PLL, DLL and OCT Resource Sharing 15–3
Pin Placement Consideration 15–5
Document Revision History 15–5

Chapter 16. Power Estimation Methods for External Memory Interface Designs

Document Revision History 16–2

Chapter Revision Dates

The chapters in this document, External Memory Interface Handbook, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Selecting Your Memory
Revised: *November 2012*
Part Number: *EMI_DG_001-6.0*

Chapter 2. Selecting Your FPGA Device
Revised: *June 2012*
Part Number: *EMI_DG_002-5.0*

Chapter 3. Planning Pin and FPGA Resources
Revised: *November 2012*
Part Number: *EMI_DG_003-6.0*

Chapter 4. DDR2 and DDR3 SDRAM Board Design Guidelines
Revised: *November 2012*
Part Number: *EMI_DG_004-5.0*

Chapter 5. Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines
Revised: *June 2012*
Part Number: *EMI_DG_005-4.1*

Chapter 6. LPDDR2 SDRAM Board Design Guidelines
Revised: *November 2012*
Part Number: *EMI_DG_016-1.0*

Chapter 7. RLDRAM II and RLDRAM 3 Board Design Guidelines
Revised: *November 2012*
Part Number: *EMI_DG_006-3.2*

Chapter 8. QDR II SRAM Board Design Guidelines
Revised: *November 2012*
Part Number: *EMI_DG_007-4.2*

Chapter 9. Implementing and Parameterizing Memory IP
Revised: *November 2012*
Part Number: *EMI_DG_008-6.0*

Chapter 10. Simulating Memory IP
Revised: *November 2012*
Part Number: *EMI_DG_009-5.1*

Chapter 11. Analyzing Timing of Memory IP
Revised: *November 2012*
Part Number: *EMI_DG_010-4.2*

Chapter 12. Debugging Memory IP
Revised: *November 2012*
Part Number: *EMI_DG_011-4.2*

Chapter 13. HardCopy Design Migration Guidelines
Revised: *November 2012*
Part Number: *EMI_DG_012-2.2*

Chapter 14. Optimizing the Controller
Revised: *November 2012*
Part Number: *EMI_DG_013-3.1*

Chapter 15. PHY Considerations
Revised: *November 2012*
Part Number: *EMI_DG_014-1.2*

Chapter 16. Power Estimation Methods for External Memory Interface Designs
Revised: *November 2012*
Part Number: *EMI_DG_015-2.2*

Section I. Design Flow Guidelines

This section provides guidelines on how to select your memory and FPGA device, pin and resource planning, board design guidelines, and the memory IP design flow.

This section includes the following chapters:

- Chapter 1, Selecting Your Memory
- Chapter 2, Selecting Your FPGA Device
- Chapter 3, Planning Pin and FPGA Resources
- Chapter 4, DDR2 and DDR3 SDRAM Board Design Guidelines
- Chapter 5, Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines
- Chapter 6, LPDDR2 SDRAM Board Design Guidelines
- Chapter 7, RLDRAM II and RLDRAM 3 Board Design Guidelines
- Chapter 8, QDR II SRAM Board Design Guidelines
- Chapter 9, Implementing and Parameterizing Memory IP
- Chapter 10, Simulating Memory IP
- Chapter 11, Analyzing Timing of Memory IP
- Chapter 12, Debugging Memory IP

For information about the revision history for chapters in this section, refer to "Document Revision History" in each individual chapter.

1. Selecting Your Memory

EMI_DG_001-6.0

This chapter discusses memory selection criteria, memory strengths and weaknesses, and memory interfaces with Altera® FPGA devices. This chapter also describes the memory component's capability and provides some typical applications where these memories are used.

The Altera IP may or may not support all of the features supported by the memory.

To compare the performance of the supported external memory interfaces in Altera FPGA devices, refer to the External Memory Interface Spec Estimator page on the Altera website.

Memory Overview

System architects must resolve a number of complex issues in high-performance system applications that range from architecture, algorithms, and features of the available components. Typically, one of the fundamental problems in these applications is memories, as the bottlenecks and challenges of system performance often reside in its memory architecture. As higher speeds become necessary for external memories, signal integrity gets more difficult. Newer devices have added several features to overcome this issue. Altera FPGAs also support these advancements with dedicated I/O circuitry, various I/O standard support, and specialized intellectual property (IP).

When you select an external memory device, consider the following factors:

- Bandwidth and speed
- Cost
- Data storage size and capacity
- Latency
- Power consumption

Because no single memory type can excel in every area, system architects must determine the right balance for their design.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Table 1–1 lists the two common types of high-speed memories and their characteristics.

Table 1–1. Differences between DRAM and SRAM

Memory Type	Description	Bandwidth and Speed	Cost	Data Storage Size and Capacity	Power consumption	Latency
DRAM	A dynamic random access memory (DRAM) cell consisting of a capacitor and a single transistor. DRAM memory must be refreshed periodically to retain the data, resulting in lower overall efficiency and more complex controllers. Generally, designers select DRAM where cost per bit and capacity are important. DRAM is commonly used for main memory.	Lower bandwidth resulting in slower speed	Lower cost	Higher data storage and capacity	Higher-power consumption	Higher latency
SRAM	A static random access memory (SRAM) cell that consists of six transistors. SRAM does not need to be refreshed because the transistors continue to hold the data as long as the power supply is not cut off. Generally, designers select SRAM where speed is more important than capacity. SRAM is commonly used for cache memory.	Higher bandwidth resulting in faster speed	Higher cost	Lower data storage and capacity	Lower-power consumption	Lower latency

DDR, DDR2, and DDR3 SDRAM

This section describes and compares the features of the DDR, DDR2, and DDR2 SDRAM.

DDR SDRAM

Double data rate (DDR) SDRAM is a 2n prefetch architecture with two data transfers per clock cycle. It uses a single-ended strobe, DQS, which is associated with a group of data pins, DQ, for read and write operations. Both DQS and DQ are bidirectional ports. Address ports are shared for read and write operations.

The desktop computing market has positioned DDR SDRAM as a mainstream commodity product, which means this memory is very low-cost. DDR SDRAM is also high-density and low-power. Relative to other high-speed memories, DDR SDRAM has higher latency-they have a multiplexed address bus, which reduces the pin count (minimizing cost) at the expense of a longer and more complex bus cycle.

DDR2 SDRAM

DDR2 SDRAM is the second generation of the DDR SDRAM standard. It is a 4n prefetch architecture (internally the memory operates at half the interface frequency) with two data transfers per clock cycle. DDR2 SDRAM can use a single-ended or differential strobe, DQS or DQSn, which is associated with a group of data pins, and DQ for read and write operations. The DQS, DQSn, and DQ are bidirectional ports. Address ports are shared for read and write operations.

DDR2 SDRAM includes additional features such as increased bandwidth due to higher clock speeds, improved signal integrity on DIMMs with on-die terminations, and lower supply voltages to reduce power.

DDR3 SDRAM

DDR3 SDRAM is the latest generation of SDRAM. DDR3 SDRAM is internally configured as an eight-bank DRAM and it uses an 8n prefetch architecture to achieve high-speed operation. The 8n prefetch architecture is combined with an interface that transfers two data words per clock cycle at the I/O pins. A single read or write operation for DDR3 SDRAM consists of a single 8n-bit wide, four-clock data transfer at the internal DRAM core and two corresponding n-bit wide, one-half clock cycle data transfers at the I/O pins. DDR3 SDRAMs are available as components and modules, such as DIMMs, SODIMMs, RDIMMs, and LRDIMMs.

DDR3 SDRAM is effective at saving system power, increasing system performance, achieving better maximum throughput, and improving signal integrity with fly-by and dynamic on-die terminations.

Read and write operations to the DDR3 SDRAM are burst oriented. Operation begins with the registration of an active command, which is then followed by a read or write command. The address bits registered coincident with the active command select the bank and row to be activated (BA0 to BA2 select the bank; A0 to A15 select the row). The address bits registered coincident with the read or write command select the starting column location for the burst operation, determine if the auto precharge command is to be issued (via A10), and select burst chop (BC) of 4 or burst length (BL) of 8 mode at runtime (via A12), if enabled in the mode register. Before normal operation, the DDR3 SDRAM must be powered up and initialized in a predefined manner.

Differential strobes DQS and DQSn are mandated for DDR3 SDRAM and are associated with a group of data pins, as is DQ for read and write operations. DQS, DQSn, and DQ ports are bidirectional. Address ports are shared for read and write operations.

The DDR3 SDRAM high-performance controller only supports local interfaces running at half the rate of the memory interface.

For more information, refer to the respective DDR, DDR2, and DDR3 SDRAM data sheets.

For more information about parameterizing the DDR2 and DDR3 SDRAM IP, refer to the *Implementing and Parameterizing Memory IP* chapter.

DDR, DDR2, and DDR3 SDRAM Comparison

Table 1–2 compares DDR, DDR2, and DDR3 SDRAM features.

Table 1–2. DDR, DDR2, and DDR3 SDRAM Features

Feature	DDR SDRAM	DDR2 SDRAM	DDR3 SDRAM	DDR3 SDRAM Advantage
Voltage	2.5 V	1.8 V	1.5 V	Reduces memory system power demand from DDR or DDR2 by 17%.
Density	64 MB to 1GB	256 MB to 4 GB	512 MB to 8 GB	High-density components simplify memory subsystem.
Internal banks	4 (fixed number of rows and columns)	4 and 8	8	Has higher page-to-hit ratio and better maximum throughput.
Bank interleaving	—	Allows bank interleaving	Allows bank interleaving	Is extremely effective for concurrent operations and can hide the timing overhead.
Prefetch	2	4	8	Lower memory core speed results in higher operating frequency and lower power operation.
Speed	100 to 200 MHz	200 to 533 MHz	300 to 1,066 MHz	Higher data rate.
Maximum frequency	200 MHz or 400 Mbps per DQ pin	533 MHz or 1,066 Mbps per DQ pin	1,066 MHz or 2,133 Mbps per DQ pin	Higher data rate.
Read latency	2, 2.5, 3 clocks	3, 4, 5 clocks	5, 6, 7, 8, 9, 10, and 11	Eliminating half clock setting allows 8n prefetch architecture.
Additive latency *(1)*	—	0, 1, 2, 3, 4	0, CL1, or CL2	Improves command efficiency.
Write latency	One clock	Read latency – 1	5, 6, 7, or 8	Improves command efficiency.
CAS latency	2, 2.5, 3	2, 3, 4, 5	5, 6, 7, 8, 9, 10	Improves command efficiency.
Burst length	2, 4, 8	4, 8	8	Improves command efficiency.
Termination	PCB, discrete to V_{TT}	Discrete to V_{TT} or ODT	Discrete to V_{TT} or ODT parallel termination. Controlled impedance output.	Improves signaling, eases PCB layout, reduces system cost.
ODT	—	ODT signal options of 50, 75, or 150 Ω on all DQ, DM, and DQS and DQSn signals	Parallel ODT options of RZQ/2, RZQ/4, or RZQ/6 Ω on all DQ, DM, and DQS and DQSn signals	DDR3 supports calibrated parallel ODT through an external resistor RZQ signal termination. DDR3 also supports dynamic ODT.
Data strobes	Single-ended	Differential or single-ended	Differential mandated	Improves timing margin.
Clock, address, and command (CAC) layout	Balanced tree	Balanced tree	Series or daisy chained	The DDR3 SDRAM read and write leveling feature allows for a much simplified PCB and DIMM layout. You can still optionally use the balanced tree topology by using the DDR3 without the leveling option.

Note to Table 1–2:

(1) The Altera DDR and DDR2 SDRAM high-performance controllers do not support additive latency, but the high-performance controller II does.

QDR, QDR II, and QDR II+ SRAM

Quad Data Rate (QDR) SRAM has independent read and write ports that run concurrently at double data rate. QDR SRAM is true dual-port (although the address bus is still shared), which gives this memory a significantly higher bandwidth, allowing back-to-back transactions without the contention issues that can occur when using a single bidirectional data bus. Write and read operations share address ports.

The QDR II SRAM devices are available in ×8, ×9, ×18, and ×36 data bus width configurations. The QDR II+ SRAM devices are available in ×9, ×18, and ×36 data bus width configurations. Write and read operations are burst-oriented. All the data bus width configurations of QDR II SRAM support burst lengths of two and four. QDR II+ SRAM supports only a burst length of four. Burst-of-two and burst-of-four for QDR II and burst-of-four for QDR II+ SRAM devices provide the same overall bandwidth at a given clock speed.

For QDR II SRAM devices, the read latency is 1.5 clock cycles; for QDR II+ SRAM devices, it is 2 or 2.5 clock cycles depending on the memory device. For QDR II+ and burst-of-four QDR II SRAM devices, the write commands and addresses are clocked on the rising edge of the clock, and write latency is one clock cycle. For burst-of-two QDR II SRAM devices, the write command is clocked on the rising edge of the clock, and the write address is clocked on the falling edge of the clock. Therefore, the write latency is zero because the write data is presented at the same time as the write command.

QDR II+ and QDR II SRAM interfaces use a delay-locked loop (DLL) inside the device to edge-align the data with respect to the `K` and `Kn` or `C` and `Cn` pins. You can optionally turn off the DLL, but the performance of the QDR II+ and QDR II SRAM devices is degraded. All timing specifications listed in this document assume that the DLL is on. QDR II+ and QDR II SRAM devices also offer programmable impedance output buffers. You can set the buffers by terminating the `ZQ` pin to VSS through a resistor, RQ. The value of RQ should be five times the desired output impedance. The range for RQ should be between 175 Ω and 350 Ω with a tolerance of 10%.

QDR II/+ SRAM is best suited for applications where the required read/write ratio is near one-to-one. QDR II/+ SRAM includes additional features such as increased bandwidth due to higher clock speeds, lower voltages to reduce power, and on-die termination to improve signal integrity. QDR II+ SDRAM is the latest and fastest generation. For QDR II+ and QDR II SRAM interfaces, Altera supports both 1.5-V and 1.8-V HSTL I/O standards.

For more information, refer to the respective QDRII and QDRII+ data sheets.

For more information about parameterizing the QDRII and QDRII+ SRAM IP, refer to the *Implementing and Parameterizing Memory IP* chapter.

RLDRAM II and RLDRAM 3

Reduced latency DRAM (RLDRAM) provides DRAM-based point-to-point memory devices designed for communications, imaging, server systems, networking, and cache applications requiring high density, high memory bandwidth, and low latency. The fast random access speeds in RLDRAM devices make them a viable alternative to SRAM devices at a lower cost.

The high performance of RLDRAM is achieved by very low random access delay (tRC), low data-bus-turnaround delay, simple command protocol, and a large number of banks. RLDRAM is optimized to meet the needs of high-bandwidth networking applications.

Contrasting with the typical four banks in most memory devices, RLDRAM II is partitioned into eight banks and RLDRAM 3 is partitioned into sixteen banks. Partitioning reduces the parasitic capacitance of the address and data lines, allowing faster accesses and reducing the probability of random access conflicts. Each bank has a fixed number of rows and columns. Only one row per bank is accessed at a time. The memory (instead of the controller) controls the opening and closing of a row, which is similar to an SRAM interface.

Most DRAM memory types need both a row and column phase on a multiplexed address bus to support full random access, while RLDRAM supports a nonmultiplexed address, saving bus cycles at the expense of more pins. RLDRAM II and RLDRAM 3 use the High-Speed Transceiver Logic (HSTL) standard with double data rate (DDR) data transfer to provide a very high throughput.

There are two types of RLDRAM II or RLDRAM 3 devices—common I/O (CIO) and separate I/O (SIO). CIO devices share a single data I/O bus, which is similar to the double data rate (DDR) SDRAM interface. SIO devices, with separate data read and write buses, have an interface similar to SRAM. Altera UniPHY Memory IP only supports CIO RLDRAM.

RLDRAM II and RLDRAM 3 use a DDR scheme, performing two data transfers per clock cycle. RLDRAM II or RLDRAM 3 CIO devices use the bidirectional data pins (DQ) for both read and write data, while RLDRAM II or RLDRAM 3 SIO devices use D pins for write data (input to the memory) and Q pins for read data (output from the memory). Both types use two pairs of unidirectional free-running clocks. The memory uses DK and DK# pins during write operations, and generates QK and QK# pins during read operations. In addition, RLDRAM II and RLDRAM 3 use the system clocks (CK and CK# pins) to sample commands and addresses, and to generate the QK and QK# read clocks. Address ports are shared for write and read operations.

RLDRAM II SIO devices are available in ×9 and ×18 data bus width configurations, while the RLDRAM II CIO devices are available in ×9, ×18, ×36 data bus width configurations. RLDRAM II CIO interfaces may require an extra cycle for bus turnaround time for switching read and write operations. RLDRAM 3 devices are available in x18 and 36 data bus width configurations.

Write and read operations are burst oriented, and all the data bus width configurations of RLDRAM II and RLDRAM 3 support burst lengths of two and four. RLDRAM 3 also supports burst length of eight at bus width x18, and burst lengths of two and four at bus width x36. For detailed comparisons between RLDRAM II and RLDRAM 3 for these features, refer to Table 1–3 on page 1–8.

RLDRAM II and RLDRAM 3 also inherently include the additional memory bits used for parity or error correction code (ECC).

RLDRAM II and RLDRAM 3 also offer programmable impedance output buffers and on-die termination. The programmable impedance output buffers are for impedance matching and are guaranteed to produce 25- to 60-ohm output impedance. The on-die termination is dynamically switched on during read operations and switched off during write operations. Perform an IBIS simulation to observe the effects of this dynamic termination on your system. IBIS simulation can also show the effects of different drive strengths, termination resistors, and capacitive loads on your system.

RLDRAM 3 enables a faster, more efficient transfer of data by doubling performance and reduced latency compared to RLDRAM II. RLDRAM 3 memory is suitable for operation in which high bandwidth and deterministic performance is critical, and is optimized to meet the needs of high-bandwidth networking applications. For detailed comparisons between RLDRAM II and RLDRAM 3, refer to Table 1–3 on page 1–8.

For more information, refer to RLDRAM II and RLDRAM 3 data sheets available from the Micron website (www.micron.com).

For more information about parameterizing the RLDRAM II and RLDRAM 3 IP, refer to the *Implementing and Parameterizing Memory IP* chapter.

LPDDR2

LPDDR2-S is a high-speed SDRAM device internally configured as a 4- or 8-bank memory. All LPDDR2 devices use double data rate architecture on the address and command bus to reduce the number of input pins in the system. The 10-bit address and command bus contains command, address, and bank/row buffer information. Each command uses one clock cycle, during which command information is transferred on both the positive and negative edges of the clock.

LPDDR2-S2 and LPDDR2-S4 devices use double data rate architecture on the `DQ` pins to achieve high speed operation. The double data rate architecture is essentially a 2n/4n prefetch architecture with an interface designed to transfer two data bits per `DQ` every clock cycle at the I/O pins. A single read or write access for the LPDDR2-S2/S4 consists of a single 2n-bit wide /4n-bit wide, one clock cycle data transfer at the internal SDRAM core, and two/four corresponding n-bit wide, with one-half clock cycle data transfers at the I/O pins.

Memory Selection

One of the first considerations in choosing a high-speed memory is data bandwidth. Based on the system requirements, an approximate data rate to the external memory should be determined. You must also consider other memory attributes, including how much memory is required (density), how much latency can be tolerated, what is the power budget, and whether the system is cost sensitive.

Table 1–3 lists memory features and target markets of each technology.

Table 1–3. Memory Selection Overview (Part 1 of 2)

Parameter	LPDDR2	DDR3 SDRAM	DDR2 SDRAM	DDR SDRAM	RLDRAM II	RLDRAM 3	QDR II/+ SRAM
Bandwidth for 32 bits (Gbps) (1)	25.6	95.5	25.6	12.8	25.6	35.8	44.8
Bandwidth at % Efficiency (Gbps) (2)	17.9	23.9	17.9	9	17.9	2.5	38.1
Performance / Clock frequency	100–533 MHz	400–1,066 MHz	200–533 MHz	100–200 MHz	200–533 MHz	200–1,066 MHz	154–350 MHz
Altera-supported data rate	Up to 1,066 Mbps	Up to 2,133 Mbps	Up to 1,066 Mbps	Up to 400 Mbps	Up to 1600 Mbps	Up to 1600 Mbps	Up to 1400 Mbps
Density	64 Mbytes – 8 Gbytes	512 Mbytes–8 Gbytes, 32 Mbytes – 8 Gbytes (DIMM)	256 Mbytes–1 Gbytes, 32 Mbytes – 4 Gbytes (DIMM)	128 Mbytes–1 Gbytes, 32 Mbytes – 2 Gbytes (DIMM)	288 Mbytes, 576 Mbytes	576 Mbytes – 1.1 Gbytes	8–72 Mbytes
I/O standard	HSUL- 12 1.2V	SSTL-15 Class I, II	SSTL-18 Class I, II	SSTL-2 Class I, II	HSTL-1.8V/1.5V	HSTL-1.2V and SSTL-12	HSTL-1.8V/1.5V
Data width (bits)	8, 16, 32	4, 8, 16	4, 8, 16	4, 8, 16, 32	9, 18, 36	18, 36	8, 9, 18, 36
Burst length	4, 8, 16	8	4, 8	2, 4, 8	2, 4, 8	2, 4, 8	2, 4
Number of banks	4, 8	8	8 (>1 GB), 4	4	8	16	—
Row/column access	Row before column	Row before column	Row before column	Row before column	Row and column together or multiplexed option	Row and column together or multiplexed option	—
CAS latency (CL)	—	5, 6, 7, 8, 9, 10	3, 4, 5	2, 2.5, 3	—	—	—
Posted CAS additive latency (AL)	—	0, CL-1, CL-2	0, 1, 2, 3, 4	—	—	—	—
Read latency (RL)	3, 4, 5, 6, 7, 8	RL = CL + AL	RL = CL + AL	RL = CL	3, 4, 5, 6, 7, 8	3-16	1.5, 2, and 2.5 clock cycles
On-die termination	—	Yes	Yes	No	Yes	Yes	Yes
Data strobe	Differential bidirectional	Differential bidirectional strobe only	Differential or single-ended bidirectional strobe	Single-ended bidirectional strobe	Free-running differential read and write clocks	Free-running differential read and write clocks	Free-running read and write clocks
Refresh requirement	Yes	Yes	Yes	Yes	Yes	Yes	No

Table 1–3. Memory Selection Overview (Part 2 of 2)

Parameter	LPDDR2	DDR3 SDRAM	DDR2 SDRAM	DDR SDRAM	RLDRAM II	RLDRAM 3	QDR II/+ SRAM
Relative cost comparison	Higher than DDR SDRAM	Presently lower than DDR2	Less than DDR SDRAM with market acceptance	Low	Higher than DDR SDRAM, less than SRAM	Higher than DDR SDRAM, less than SRAM	Highest
Target market	Mobile devices that target low operating power	Desktops, servers, storage, LCDs, displays, networking, and communication equipment	Desktops, servers, storage, LCDs, displays, networking, and communication equipment	Desktops, servers, storage, LCDs, displays, networking, and communication equipment	Main memory, cache memory, networking, packet processing, and traffic management	Main memory, cache memory, networking, packet processing, and traffic management	Cache memory, routers, ATM switches, packet memories, lookup, and classification memories

Notes to Table 1–3:

(1) 32-bit data bus operating at the maximum supported frequency in a Stratix® V FPGA.

(2) 70% efficiency for DDR memories, which takes into consideration the bus turnaround, refresh, infinite burst length and random access latency and assumes 85% efficiency for QDR memories.

Altera supports the memory interfaces (shown in Table 1–3), provides various IP for the physical interface and the controller, and offers many reference designs (refer to Altera's Memory Solutions Center).

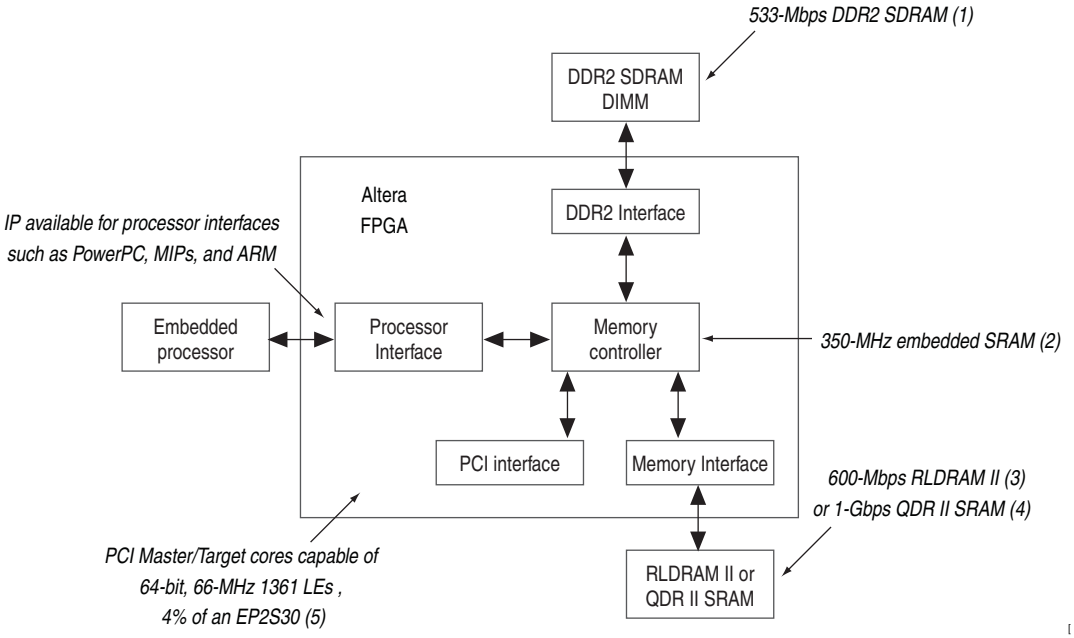
For Altera support and the maximum performance for the various high-speed memory interfaces, refer to the External Memory Interface Spec Estimator page on the Altera website.

High-Speed Memory in Embedded Processor Example

In embedded processor applications—any system that uses processors, excluding desktop processors—due to its very low cost, high density, and low power, DDR SDRAM is typically used for main memory. Next-generation processors invest a large amount of die area to on-chip cache memory to prevent the execution pipelines from sitting idle. Unfortunately, these on-chip caches are limited in size, as a balance of performance, cost, and power must be taken into consideration. In many systems, external memories are used to add another level of cache. In high-performance systems, three levels of cache memory is common: level one (8 Kbytes is common) and level two (512 Kbytes) on chip, and level three off chip (2 Mbytes).

High-end servers, routers, and even video game systems are examples of high-performance embedded products that require memory architectures that are both high speed and low latency. Advanced memory controllers are required to manage transactions between embedded processors and their memories. Altera Arria® series and Stratix series FPGAs optimally implement advanced memory controllers by utilizing their built-in DQS (strobe) phase shift circuitry. Figure 1–1 highlights some of the features available in an Altera FPGA in an embedded application, where DDR2 SDRAM is used as the main memory and QDR II SRAM or RLDRAM II/3 is an external cache level.

Figure 1–1. Memory Controller Example Using FPGA



Notes to Figure 1–1:

(1) 533-Mbps DDR2 SDRAM operation using dedicated DQS circuitry, post-amble circuitry, automatic phase shifting, and six registers in the I/O element: 790 LEs, 3% of an EP2S30, and four clock buffers (for a 72-bit interface).

(2) High-speed memory interfaces such as QDR II SRAM require at least four clock buffers to handle all the different clock phases and data directions.

(3) 600-Mbps RLDRAM II operation: 740 logic elements (LEs), 3% of an EP2S30, and four clock buffers (for a 36-bit wide interface).

(4) Embedded SRAM with features such as true-dual port and 350-MHz operation allows complex “store and forward” memory controller architectures.

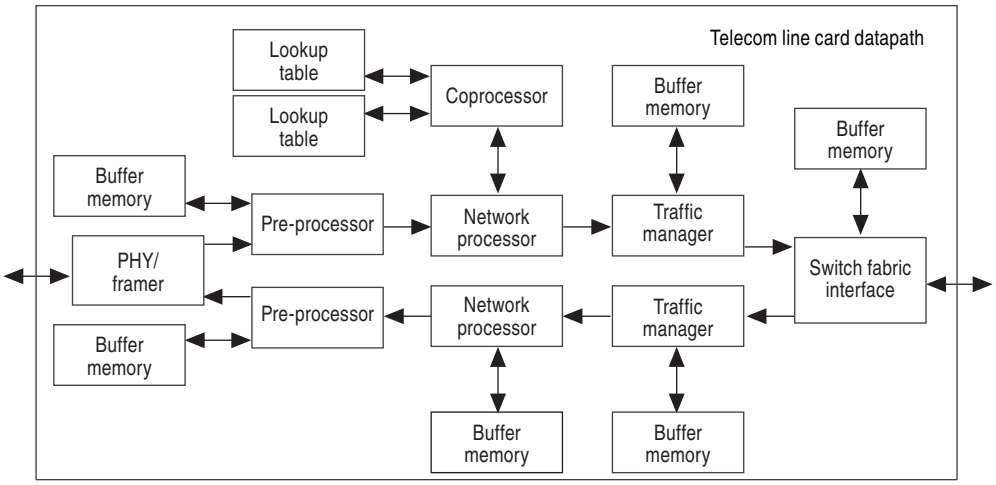
(5) The Quartus® II software reports the number of adaptive look-up tables (ALUTs) that the design uses in the FPGA. The LE count is based on this number of ALUTs.

One of the target markets of RLDRAM II/3 and QDR/QDR II SRAM is external cache memory. RLDRAM II and RLDRAM 3 have a read latency close to synchronous SRAM, but with the density of SDRAM. A sixteen times increase in external cache density is achievable with one RLDRAM II/3 versus that of synchronous static RAM (SSRAM). In contrast, consider QDR and QDR II SRAM for systems that require high bandwidth and minimal latency. Architecturally, the dual-port nature of QDR and QDR II SRAM allows cache controllers to handle read data and instruction fetches completely independent of writes.

High-Speed Memory in Telecom Example

Because telecommunication network architectures are becoming more complex, high-end network systems are running multiple 10-Gbps line cards that connect to multi-shelf switch fabrics scaling to terabits per second. Figure 1–2 shows an example of a typical system line interface card. These line cards offer interfaces ranging from a single-port OC-192 to multi-port Gbps Ethernet, and consist of a number of devices, including a PHY/framer, network processors, traffic managers, fabric interface devices, and high-speed memories.

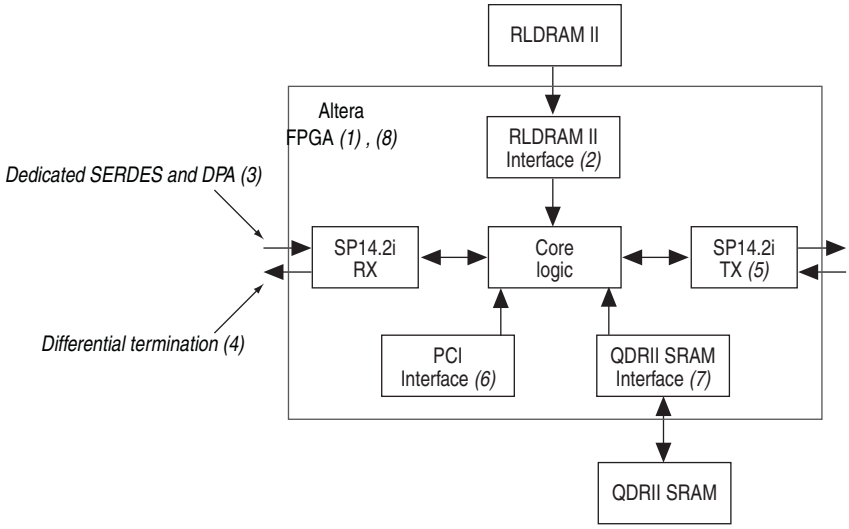
Figure 1–2. Typical Telecom System Line Interface Card



As packets traverse from the PHY/framer device to the switch fabric interface, they are buffered into memories, while the data path devices process headers (determining the destination, classifying packets, and storing statistics for billing) and control the flow of packets into the network to avoid congestion. Typically DDR/DDR2/DDR3 SDRAM and RLDRAM II/3 are used for large buffer memories off network processors, traffic managers, and fabric interfaces, while QDR and QDR II SRAMs are used for look-up tables (LUTs) off preprocessors and coprocessors.

In many designs, FPGAs connect devices together for interoperability and coprocessing, implement features that are not supported by ASIC devices, or implement a device function entirely. Altera Stratix series FPGAs implement traffic management, packet processing, switch fabric interfaces, and coprocessor functions, using features such as 1-Gbps LVDS I/O, high-speed memory interface support, multi-gigabit transceivers, and IP cores. Figure 1–3 highlights some of these features in a packet buffering application where RLDRAM II is used for packet buffer memory and QDR II SRAM is used for control memory.

Figure 1–3. FPGA Example in Packet Buffering Application



Notes to Figure 1–3:

(1) As an example, 85% of the LEs still available in an EP2S90.
(2) 600-Mbps RLDRAM II operation: 740 LEs, 1% of an EP2S90, and four clock buffers (for a 36-bit wide interface).
(3) Dedicated hardware SERDES and DPA circuitry allows clean and reliable implementation of 1-Gbps LVDS.
(4) Differential termination is built in Stratix FPGAs, simplifying board layout and improving signal quality.
(5) SPI 4.2i core capable of 1 Gbps: 5178 LEs per Rx, 6087 LEs per Tx, 12% of an ES2S90, and four clock buffers (for both directions using individual buffer mode, 32-bit data path, and 10 logical ports).
(6) PCI cores capable of 64-bit 66-MHz 656 LEs, 1% of an EP2S90 for a 32-bit target
(7) 1-Gbps QDR II SRAM operation: 100 LEs, 0.1% of an EP2S90, and four clock buffers (for an 18-bit interface).
(8) Note that the Quartus II software reports the number of ALUTs that the design uses in Stratix II devices. The LE count is based on this number of ALUTs.

SDRAM is usually the best choice for buffering at high data rates due to the large amounts of memory required. Some system designers take a hybrid approach to the memory architecture, using SRAM to store the packet headers and DRAM to store the payload. The depth of the memories depends on the architecture and throughput of the system.

The buffer memory for the packet buffering application of an OC-192 line card (approximately 10 Gbps) must be able to sustain a minimum of one write and one read operation, which requires a memory bandwidth of 20 Gbps to operate at full line rate (more bandwidth is required if the headers are modified). The bandwidth requirement for memory is a key factor in memory selection (Table 1–3 on page 1–8). As an example, a simple first-order calculation using RLDRAM II as buffer memory requires a bus width of 48 bits to sustain 20 Gbps (300 MHz × 2 DDR × 0.70 efficiency × 48 bits = 20.1 Gbps), which needs two RLDRAM II parts (one ×18 and one ×36).

RLDRAM II and RLDRAM 3 also inherently include the additional memory bits used for parity or error correction code (ECC). QDR and QDR II SRAM have bandwidth and low random access latency advantages that make them useful for control memory in queue management and traffic management applications. Another typical implementation for this memory is billing and packet statistics, where each packet requires counters to be read from memory, incremented, and then rewritten to memory. The high bandwidth, low latency, and optimal one-to-one read/write ratio make QDR SRAM ideal for this feature.

Document Revision History

Table 1–4 lists the revision history for this document.

Table 1–4. Document Revision History

Date	Version	Changes
November 2012	6.0	Added RLDRAM 3 support.
June 2012	5.0	■ Added LPDDR2 support. ■ Added Feedback icon.
November 2011	4.0	Moved and reorganized “Selecting your Memory” section to Volume 2:Design Guidelines.
June 2011	3.0	Added “Selecting Memory IP” chapter from Volume 2 Section I.
December 2010	2.1	■ Moved protocol-specific feature information to the memory interface user guides in Volume 3. ■ Updated maximum clock rate information for 10.1.
July 2010	2.0	■ Added specifications for DDR2 and DDR3 SDRAM Controllers with UniPHY. ■ Streamlined the specification tables. ■ Added reference to web-based Specification Estimator Tool.
January 2010	1.1	Updated DDR, DDR2, and DDR3 specifications.
November 2009	1.0	First published.

2. Selecting Your FPGA Device

EMI_DG_002-5.0

This chapter discusses the following topics about selecting the right Altera® FPGA device for your external memory interface:

- "Device Family Selection" on page 2–1
- "Device Settings Selection" on page 2–4

Use this document with the *Planning Pin and FPGA Resources* chapter, before you start implementing your external memory interface.

Device Family Selection

Altera external memory interfaces support three FPGA device families—Arria®, Stratix®, and Cyclone® device families. These FPGA device families varies in terms of cost, memory standards, speed grade, and features.

Use the Altera Product Selector to find and compare specifications and features of Altera devices.

The following sections describe the factors that you must consider when selecting an FPGA device family.

Cost

The cost of an FPGA is the main factor in selecting a device family that suites your design. The Stratix FPGA family delivers the industry's highest bandwidth and density. It also has the highest level of system integration with ultimate flexibility at a reduced cost, and the lowest total power for high-end applications. By combining high density, high performance, and a rich feature set, the Stratix series FPGAs allow you to integrate more functions and maximize system bandwidth.

Altera's Arria FPGA series is designed to deliver a balance between cost, power, and performance. This device family targets the cost-and power-sensitive transceiver-based applications. The Arria FPGA series has a rich feature set of memory, logic, and digital signal processing (DSP) blocks combined with superior signal integrity of up to 10G transceivers that allows you to integrate more functions and maximize system bandwidth.

The Cyclone FPGA series is designed for the lowest power consumptions and cost-sensitive design needs. The Cyclone FPGA family provides the market's lowest system cost and lowest power FPGA solution for applications in various fields.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Memory Standards

There are two common types of high-speed memories that are supported by Altera devices—dynamic random access memory (DRAM) and static random access memory (SRAM). The commonly used DRAM devices include DDR, DDR2, DDR3 SDRAM, LPDDR2, and RLDRAM II, while SRAM devices include QDR II and QDR II+ SRAM.

For more information about these memory types, refer to the *Selecting Your Memory* chapter.

Different Altera FPGA devices support different memory types; not all Altera devices support all memory types and configurations. Before you start your design, you must select an Altera device, which supports the memory standard and configurations you plan to use.

In addition, Altera's FPGA devices support various data widths for different memory interfaces. The memory interface support between density and package combinations differs, so you must determine which FPGA device density and package combination suits your application.

For more information about the supported memory types and configurations, refer to the External Memory Interface Spec Estimator page on the Altera website.

I/O Interfaces

Ideally any interface should reside entirely in a single bank. However, interfaces that span across multiple adjacent banks or the entire side of a device are also fully supported. Interfaces that span across sides (top and bottom, or left and right) and wraparound interfaces provide the same level of performance.

Table 2-1 lists the location of interfaces for various device families.

Table 2–1. Location of I/O Interfaces (Part 1 of 2)

Devices	Interface Location	Exceptions
Arria II GX (2)	Top and bottom sides	—
Arria II GZ	Top and bottom sides	—
Arria V (2)	Top and bottom sides	5AGXA1 and 5AGXA3 devices support interfaces on the left side.
Cyclone III	Top and bottom sides	Cyclone III E devices support interfaces support interfaces spanning left and right sides.
Cyclone IV (2)	Top and bottom sides	—
Stratix II (1)	All sides	—
Stratix III (1)	All sides	—

Table 2–1. Location of I/O Interfaces (Part 2 of 2)

Devices	Interface Location	Exceptions
Stratix IV *(1)*	All sides	EP4SGX290 and EP4SGX360 devices does not support interfaces on left and right side. EP4SGX70, EP4SGX110, EP4SGX180, and EP4SGX230 devices does not support interfaces on the right side.
Stratix V *(2)*	Top and bottom sides	—
Cyclone V	Right, top, and bottom sides	5CEA2 and 5CEA4 devices support interfaces on all sides.

Notes to Table 2–1:

(1) Although vertical and horizontal I/O timing parameters are not identical, timing closure can be achieved on all sides of the FPGA for the maximum interface frequency.

(2) There are no user I/O pins on the right and left sides of the device, other than the transceiver pins available in these devices.

For more information about I/O interfaces supported for each device, refer to the respective device handbooks.

Wraparound Interfaces

For maximum performance, Altera recommends that data groups for external memory interfaces should always be within the same side of a device, ideally reside within a single bank. High-speed memory interfaces using top or bottom I/O bank versus left or right IO bank have different timing characteristics, so the timing margins are also different. However, Altera can support interfaces with wraparound data groups that wraparound a corner of the device between vertical and horizontal I/O banks at some speeds. Some devices wraparound interfaces are the same speed as row or column interfaces.

The Arria II GX, Cyclone III and Cyclone IV devices can support wraparound interface across all sides of devices that are not used for transceivers. Other Altera devices only support interfaces with data groups that wraparound a corner of the device.

Read and Write Leveling

The Stratix III, Stratix IV, and Stratix V I/O registers include read and write leveling circuitry to enable skew to be removed or applied to the interface on a DQS group basis. There is one leveling circuit located in each I/O subbank.

UniPHY-based designs do not require read leveling circuitry during read leveling operation.

For more information about read and write leveling, refer to "Leveling Circuitry" section in *Volume 3: Reference Material* of the *External Memory Interface Handbook*.

Dynamic OCT

The Arria II GZ, Arria V, Cyclone V, Stratix III, Stratix IV, and Stratix V devices support dynamic calibrated OCT. This feature allows the specified series termination to be enabled during writes, and parallel termination to be enabled during reads. These I/O features allow you to simplify PCB termination schemes.

Device Settings Selection

After you have selected the appropriate FPGA device family for your memory interface, configure the device settings of your selected FPGA device family to meet your design needs.

Refer to the device ordering code and determine the appropriate device settings for your target device family.

For more information about the ordering code for your target device, refer to the "Ordering Information" section in volume 1 of the respective device handbooks.

The following sections describe the ordering code and how to select the appropriate device settings based on the ordering code to meet the requirements of your external memory interface.

Speed Grade

The device speed grade affects the device timing performance, timing closure, and power utilization. The device with the smallest number is the fastest device and vice-versa. Generally, the faster devices cost more.

Operating Temperature

The operating temperature of the FPGA is divided into the following categories:

- Commercial grade—Used for all device families. The operating temperature range from 0° C to 85° C
- Industrial grade—Used for all device families. The operating temperature range from -40° C to 100° C
- Military grade—Used for Stratix IV device family only. The operating temperature range from -55° C to 125° C
- Automotive grade—Used for Cyclone IV and Cyclone V device families only. The operating temperature range from -40° C to 125° C

Package Size

Each FPGA family has various range of package size. Package size refers to the actual size of an FPGA device and corresponds to the number of pin counts. For example,the package size for the smallest FPGA device in the Stratix IV family is 29 mm x 29 mm, categorized under the F780 package option, where F780 refers to a device with 780 pin counts.

For more information about the package size available for your device, refer to the respective device handbooks.

Device Density and I/O Pin Counts

An FPGA device of the same device family and package size also varies in terms of device density and I/O pin counts. For example, after you have selected the Stratix IV device family with the F780 packaging option, you must determine the type of device models that ranges from EP4GX70 to EP4GX230. Each of these devices have similar speed grades that ranges from grade 2 to grade 4, but are different in density.

Device Density

Device density refers to the number of logic elements (LEs). For example, PLLs, memory blocks, and so on. An FPGA device with higher density contains more logic elements in less area.

I/O Pin Counts

To meet the growing demand for memory bandwidth and memory data rates, memory interface systems use parallel memory channels and multiple controller interfaces. However, the number of memory channels is limited by the package pin count of the Altera devices. Therefore, you must consider device pin count when you select a device; you must select a device with enough I/O pins for your memory interface requirement.

The number of device pins depends on the memory standard, the number of memory interfaces, and the memory data width. For example, a ×72 DDR3 SDRAM single-rank interface requires 125 I/O pins:

- 72 DQ pins (including ECC)
- 9 DM pins
- 9 DQS, DQSn differential pin pairs
- 17 address pins (address and bank address)
- 7 command pins (CAS, RAS, WE, CKE, ODT, reset, and CS)
- 1 CK, CK# differential pin pair

For more information about the number of embedded memory, PLLs and user I/O counts that are available for your device, refer to the respective device handbooks.

For the number of DQS groups available for each FPGA device, refer to the respective device handbooks.

For the maximum number of controllers that is supported by the FPGAs for different memory types, refer to the *Planning Pin and FPGA Resources* chapter.

Altera devices do not limit the interface widths beyond the following requirements:

- The DQS, DQ, clock, and address signals of the entire interface must reside within the same bank or side of the device if possible, to achieve better performance. Although wraparound interfaces are also supported at limited frequencies.

- The maximum possible interface width in any particular device is limited by the number of DQS and DQ groups available within that bank or side.
- Sufficient regional clock networks are available to the interface PLL to allow implementation within the required number of quadrants.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other clock, address, and command pin placement requirements.
- The greater the number of banks, the greater the skew. Altera recommends that you always compile a test project of your desired configuration and confirm that it meets timing requirement.

Your pin count calculation also determines which device side to use (top or bottom, left or right, and wraparound).

There is a constraint in Arria® II GX devices when assigning DQS and DQ pins. You are only allowed to use twelve of the sixteen I/O pins in an I/O module as DQ pins. The remaining four pins can only be used as input pins.

For DQS groups pin-out restriction format, refer to *Arria II GX Pin Connection Guidelines*.

The Arria II GX, Cyclone IV, and Stratix V devices do not support the left interface. There are no user I/O pins, other than the transceiver pins available in these devices.

Document Revision History

Table 2–2 lists the revision history for this document.

Table 2–2. Document Revision History

Date	Version	Changes
June 2012	5.0	■ Added LPDDR2 support. ■ Added Feedback icon.
November 2011	4.0	Moved and reorganized “Selecting your FPGA device” section to Volume 2:Design Guidelines.
June 2011	3.0	Added “Selecting a Device” chapter from Volume 2 Section I.
December 2010	2.1	■ Moved protocol-specific feature information to the memory interface user guides in Volume 3. ■ Updated maximum clock rate information for 10.1.
July 2010	2.0	■ Added specifications for DDR2 and DDR3 SDRAM Controllers with UniPHY. ■ Streamlined the specification tables. ■ Added reference to web-based Specification Estimator Tool.
January 2010	1.1	Updated DDR, DDR2, and DDR3 specifications.
November 2009	1.0	First published.

3. Planning Pin and FPGA Resources

This chapter is for board designers who need to determine the FPGA pin usage, to create the board layout for the system, as the board design process sometimes occurs concurrently with the RTL design process.

Use this document with the *External Memory Interfaces* chapter of the relevant device family handbook.

All external memory interfaces typically require the following FPGA resources:

- Interface pins
- PLL and clock network
- DLL (not applicable in Cyclone® III and Cyclone IV devices)
- Other FPGA resources—for example, core fabric logic, and on-chip termination (OCT) calibration blocks

When you know the requirements for your memory interface, you can then start planning how you can architect your system. The I/O pins and internal memory cannot be shared for other applications or memory interfaces. However, if you do not have enough PLLs, DLLs, or clock networks for your application, you may share these resources among multiple memory interfaces or modules in your system.

Ideally, any interface should reside entirely in a single bank. However, interfaces that span multiple adjacent banks or the entire side of a device are also fully supported. In addition, you may also have wraparound memory interfaces, where the design uses two adjacent sides of the device and the memory interface logic resides in a device quadrant. In some cases, top or bottom bank interfaces have higher supported clock rate than left or right or wraparound interfaces.

Interface Pins

Any I/O banks that do not support transceiver operations in Arria® II, Arria V, Cyclone III, Cyclone IV, Stratix® III, Stratix IV, and Stratix V devices support memory interfaces. However, DQS (data strobe or data clock) and DQ (data) pins are listed in the device pin tables and fixed at specific locations in the device. You must adhere to these pin locations as these locations are optimized in routing to minimize skew and maximize margin. Always check the external memory interfaces chapters from the device handbooks for the number of DQS and DQ groups supported in a particular device and the pin table for the actual locations of the DQS and DQ pins.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

For maximum performance and best skew across the interface, each required memory interface should completely reside within a single I/O bank, or at least one side of the device. Address and command pins can be constrained in a different side of the device if there are not enough pins available. For example, you may have the read and write data pins on the top side of the device, and have the address and command pins on the left side of the device. In memory interfaces with unidirectional data, you may also have all the read data pins on the top side of the device and the write data pin on the left side of the device. However, you should not break a unidirectional pin group across multiple sides of the device. Memory interfaces typically have the following pin groups:

- Write data pin group and read data pin group
- Address and command pin group

Table 3–1 lists a summary of the number of pins required for various example memory interfaces. Table 3–1 uses series OCT with calibration, parallel OCT with calibration, or dynamic calibrated OCT, when applicable, shown by the usage of R_{UP} and R_{DN} pins or R_{ZQ} pin.

Table 3–1. Pin Counts for Various Example Memory Interfaces (1), (2) (Part 1 of 2)

Memory Interface	FPGA DQS Bus Width	Number of DQ Pins	Number of DQS Pins	Number of DM/BWSn Pins	Number of Address Pins (3)	Number of Command Pins	Number of Clock Pins	R_{UP}/R_{DN} Pins (4)	R_{ZQ} Pins (11)	Total Pins with R_{UP}/R_{DN}	Total Pins with R_{ZQ}
LPDDR2	x8	8	2	1	10	2	2	N/A	1	N/A	26
		16	4	2	10	2	2	N/A	1	N/A	37
		72	18	9	10	2	2	N/A	1	N/A	114
DDR3 SDRAM (5), (6)	×4	4	2	0 (7)	14	10	2	2	1	34	33
	×8	8	2	1	14	10	2	2	1	39	38
		16	4	2	14	10	2	2	1	50	49
		72	18	9	14	14	4	2	1	134	133
DDR2 SDRAM (8)	×4	4	1	1 (7)	15	9	2	2	1	34	33
	×8	8	1 (9)	1	15	9	2	2	1	38	37
		16	2 (9)	2	15	9	2	2	1	48	47
		72	9 (9)	9	15	12	6	2	1	125	124
DDR SDRAM (6)	×4	4	1	1 (7)	14	7	2	2	1	29	28
	×8	8	1	1	14	7	2	2	1	33	35
		16	2	2	14	7	2	2	1	43	42
		72	9	9	13	9	6	2	1	118	117
QDR II+ SRAM	×9	18	2	1	19	3 (10)	4	2	1	49	48
	×18	36	2	2	18	3 (10)	4	2	1	67	66
	×36	72	2	4	17	3 (10)	4	2	1	104	103
QDR II SRAM	×9	18	2	1	19	2	4	2	1	48	47
	×18	36	2	2	18	2	4	2	1	66	65
	×36	72	2	4	17	2	4	2	1	103	102

Table 3–1. Pin Counts for Various Example Memory Interfaces (1), (2) (Part 2 of 2)

Memory Interface	FPGA DQS Bus Width	Number of DQ Pins	Number of DQS Pins	Number of DM/BWSn Pins	Number of Address Pins (3)	Number of Command Pins	Number of Clock Pins	R_{UP}/R_{DN} Pins (4)	R_{ZQ} Pins (11)	Total Pins with R_{UP}/R_{DN}	Total Pins with R_{ZQ}
RLDRAMII CIO	×9	9	2	1	22	7 (10)	4	2	1	47	46
		18	2	1	21	7 (10)	6	2	1	57	56
	×18	36	2	1	20	7 (10)	8	2	1	76	75
RLDRAM II SIO	×9	18	2	1	22	7 (10)	4	2	1	56	55
		36	2	1	21	7 (10)	6	2	1	75	74
	×18	72	2	1	20	7 (10)	8	2	1	112	111

Notes to Table 3–1:

(1) These example pin counts are derived from memory vendor data sheets. Check the exact number of addresses and command pins of the memory devices in the configuration that you are using.

(2) PLL and DLL input reference clock pins are not counted in this calculation.

(3) The number of address pins depend on the memory device density.

(4) Some DQS or DQ pins are dual purpose and can also be required as R_{UP}, R_{DN}, or configuration pins. A DQS group is lost if you use these pins for configuration or as R_{UP} or R_{DN} pins for calibrated OCT. Pick R_{UP} and R_{DN} pins in a DQS group that is not used for memory interface purposes. You may need to place the DQS and DQ pins manually if you place the R_{UP} and R_{DN} pins in the same DQS group pins.

(5) The `TDQS` and `TDQS#` pins are not counted in this calculation, as these pins are not used in the memory controller.

(6) Numbers are based on 1-GB memory devices.

(7) Altera® FPGAs do not support DM pins in ×4 mode with differential DQS signaling.

(8) Numbers are based on 2-GB memory devices without using differential DQS, `RDQS`, and `RDQS#` pin support.

(9) Assumes single ended DQS mode. DDR2 SDRAM also supports differential DQS, which makes these DQS and DM numbers identical to DDR3 SDRAM.

(10) The QVLD pin that indicates read data valid from the QDR II+ SRAM or RLDRAM II device, is included in this number.

(11) R_{ZQ} pins are supported by Arria V, Cyclone V, and Stratix V devices.

Maximum interface width varies from device to device depending on the number of I/O pins and DQS or DQ groups available. Achievable interface width also depends on the number of address and command pins that the design requires. To ensure adequate PLL, clock, and device routing resources are available, you should always test fit any IP in the Quartus® II software before PCB sign-off.

Altera devices do not limit the width of external memory interfaces beyond the following requirements:

- Maximum possible interface width in any particular device is limited by the number of DQS groups available.
- Sufficient clock networks are available to the interface PLL as required by the IP.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other address and command, and clock pin placement requirements.
- The greater the number of banks, the greater the skew, hence Altera recommends that you always generate a test project of your desired configuration and confirm that it meets timing.

While you should use the Quartus II software for final pin fitting, you can estimate whether you have enough pins for your memory interface using the following steps:

1. Find out how many read data pins are associated per read data strobe or clock pair, to determine which column of the DQS and DQ group availability (×4, ×8/×9, ×16/×18, or ×32/×36) look at the pin table.
2. Check the device density and package offering information to see if you can implement the interface in one I/O bank or on one side or on two adjacent sides.

   If you target Arria II GX, Cyclone III, or Cyclone IV devices and you do not have enough I/O pins to have the memory interface on one side of the device, you may place them on the other side of the device. These device families allow a memory interface to span across the top and bottom, or left and right sides of the device. For any interface that spans across two different sides, use the wraparound interface performance.

3. Calculate the number of other memory interface pins needed, including any other clocks (write clock or memory system clock), address, command, R_{UP}, R_{DN}, R_{ZQ}, and any other pins to be connected to the memory components. Ensure you have enough pins to implement the interface in one I/O bank or one side or on two adjacent sides.

   The DQS groups in Arria II GX devices reside on I/O modules, each consisting of 16 I/O pins. You can only use a maximum of 12 pins per I/O modules when the pins are used as DQS or DQ pins or HSTL/SSTL output or HSTL/SSTL bidirectional pins. When counting the number of available pins for the rest of your memory interface, ensure you do not count the leftover four pins per I/O modules used for DQS, DQ, address and command pins. The leftover four pins can be used as input pins only.

   Refer to the device pin-out tables and look for the blank space in the relevant DQS group column to identify the four pins that cannot be used in an I/O module for Arria II GX devices.

You should always try the proposed pin-outs with the rest of your design in the Quartus II software (with the correct I/O standard and OCT connections) before finalizing the pin-outs, as there may be some interactions between modules that are illegal in the Quartus II software that you may not find out unless you try compiling a design and use the Quartus II Pin Planner.

The following sections describe the pins for each memory interfaces.

DDR, DDR2, and DDR3 SDRAM

This section provides a description of the clock, command, address, and data signals for DDR, DDR2, and DDR3 SDRAM interfaces.

Clock Signals

DDR, DDR2, and DDR3 SDRAM devices use CK and CK# signals to clock the address and command signals into the memory. Furthermore, the memory uses these clock signals to generate the DQS signal during a read through the DLL inside the memory. The SDRAM data sheet specifies the following timings:

- t_{DQSCK} is the skew between the CK or CK# signals and the SDRAM-generated DQS signal
- t_{DSH} is the DQS falling edge from CK rising edge hold time
- t_{DSS} is the DQS falling edge from CK rising edge setup time
- t_{DQSS} is the positive DQS latching edge to CK rising edge

These SDRAM have a write requirement (t_{DQSS}) that states the positive edge of the DQS signal on writes must be within ± 25% (± 90°) of the positive edge of the SDRAM clock input. Therefore, you should generate the CK and CK# signals using the DDR registers in the IOE to match with the `DQS` signal and reduce any variations across process, voltage, and temperature. The positive edge of the SDRAM clock, CK, is aligned with the DQS write to satisfy t_{DQSS}.

DDR3 SDRAM can use a daisy-chained control address command (CAC) topology, in which the memory clock must arrive at each chip at a different time. To compensate for this flight-time skew between devices across a typical DIMM, write leveling must be employed.

Command and Address Signals

Command and address signals in SDRAM devices are clocked into the memory device using the CK or CK# signal. These pins operate at single data rate (SDR) using only one clock edge. The number of address pins depends on the SDRAM device capacity. The address pins are multiplexed, so two clock cycles are required to send the row, column, and bank address. The CS#, RAS, CAS, WE, CKE, and ODT pins are SDRAM command and control pins. DDR3 SDRAM has an additional pin, `RESET#`, while some DDR3 DIMMs have these additional pins: `RESET#`, `PAR_IN`, and `ERR_OUT#`. The `RESET#` pin uses the 1.5-V LVCMOS I/O standard, and the `PAR_IN` and `ERR_OUT#` pins use the SSTL-15 I/O standard.

The DDR2 SDRAM command and address inputs do not have a symmetrical setup and hold time requirement with respect to the SDRAM clocks, CK, and CK#.

For ALTMEMPHY or Altera SDRAM high-performance controllers in Stratix III and Stratix IV devices, the command and address clock is a dedicated PLL clock output whose phase can be adjusted to meet the setup and hold requirements of the memory clock. The command and address clock is also typically half-rate, although a full-rate implementation can also be created. The command and address pins use the DDIO output circuitry to launch commands from either the rising or falling edges of the clock. The chip select (`mem_cs_n`), clock enable (`mem_cke`), and ODT (`mem_odt`) pins are only enabled for one memory clock cycle and can be launched from either the rising or falling edge of the command and address clock signal. The address and other command pins are enabled for two memory clock cycles and can also be launched from either the rising or falling edge of the command and address clock signal.

In ALTMEMPHY-based designs, the command and address clock `ac_clk_1x` is always half rate. However, because of the output enable assertion, CS#, CKE, and ODT behave like full-rate signals even in a half-rate PHY.

In Arria II GX and Cyclone III devices, the command and address clock is either shared with the `write_clk_2x` or the `mem_clk_2x` clock.

Data, Data Strobes, DM, and Optional ECC Signals

DDR SDRAM uses bidirectional single-ended data strobe (DQS); DDR3 SDRAM uses bidirectional differential data strobes. The DQSn pins in DDR2 SDRAM devices are optional but recommended for DDR2 SDRAM designs operating at more than 333 MHz. Differential DQS operation enables improved system timing due to reduced crosstalk and less simultaneous switching noise on the strobe output drivers. The DQ pins are also bidirectional. Regardless of interface width, DDR SDRAM always operates in ×8 mode DQS groups. DQ pins in DDR2 and DDR3 SDRAM interfaces can operate in either ×4 or ×8 mode DQS groups, depending on your chosen memory device or DIMM, regardless of interface width. The ×4 and ×8 configurations use one pair of bidirectional data strobe signals, DQS and DQSn, to capture input data. However, two pairs of data strobes, UDQS and UDQS# (upper byte) and LDQS and LDQS# (lower byte), are required by the ×16 configuration devices. A group of DQ pins must remain associated with its respective DQS and DQSn pins.

The DQ signals are edge-aligned with the DQS signal during a read from the memory and are center-aligned with the DQS signal during a write to the memory. The memory controller shifts the DQ signals by –90° during a write operation to center align the DQ and DQS signals. The PHY IP delays the DQS signal during a read, so that the DQ and DQS signals are center aligned at the capture register. Altera devices use a phase-locked loop (PLL) to center-align the DQS signal with respect to the DQ signals during writes and Altera devices (except Cyclone III and Cyclone IV devices) use dedicated DQS phase-shift circuitry to shift the incoming DQS signal during reads. Figure 3–1 shows an example where the DQS signal is shifted by 90° for a read from the DDR2 SDRAM.

Figure 3–1. Edge-aligned DQ and DQS Relationship During a DDR2 SDRAM Read in Burst-of-Four Mode

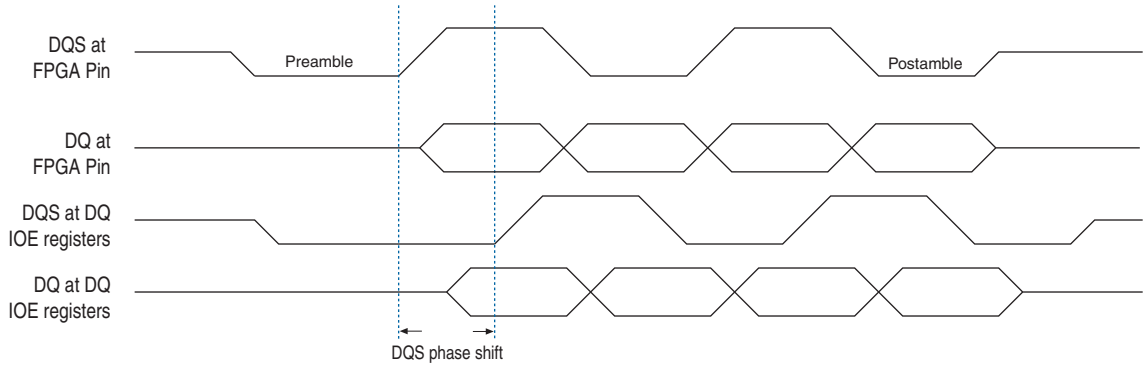
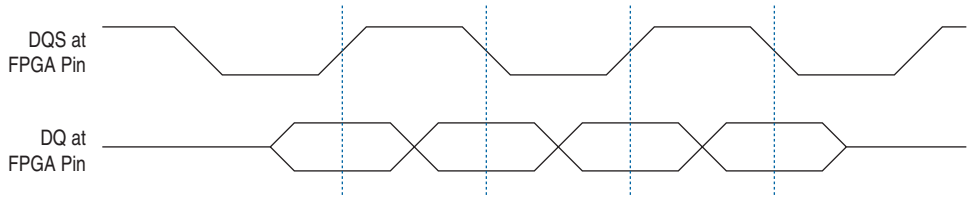


Figure 3–2 shows an example of the relationship between the data and data strobe during a burst-of-four write.

Figure 3–2. DQ and DQS Relationship During a DDR2 SDRAM Write in Burst-of-Four Mode



The memory device's setup (t_{DS}) and hold times (t_{DH}) for the write DQ and DM pins are relative to the edges of DQS write signals and not the CK or CK# clock. Setup and hold requirements are not necessarily balanced inDDR2 and DDR3 SDRAM, unlike in DDR SDRAM devices.

The DQS signal is generated on the positive edge of the system clock to meet the t_{DQSS} requirement. DQ and DM signals use a clock shifted –90° from the system clock, so that the DQS edges are centered on the DQ or DM signals when they arrive at the DDR2 SDRAM. The DQS, DQ, and DM board trace lengths need to be tightly matched (within 20 ps).

The SDRAM uses the DM pins during a write operation. Driving the DM pins low shows that the write is valid. The memory masks the DQ signals if the DM pins are driven high. To generate the DM signal, Altera recommends that you use the spare DQ pin within the same DQS group as the respective data, to minimize skew.

The DM signal's timing requirements at the SDRAM input are identical to those for DQ data. The DDR registers, clocked by the –90° shifted clock, create the DM signals.

Some SDRAM modules support error correction coding (ECC) to allow the controller to detect and automatically correct error in data transmission. The 72-bit SDRAM modules contain eight extra data pins in addition to 64 data pins. The eight extra ECC pins should be connected to a single DQS or DQ group on the FPGA.

DIMM Options

Unbuffered DIMMs (UDIMMs) require one set of chip-select (`CS#`), on-die termination (ODT), clock-enable (`CKE#`), and clock pair (`CK/CKn`) for every physical rank on the DIMM. Registered DIMMs use only one pair of clocks and require a minimum of two chip-select signals, even in the case of a single rank RDIMM.

Compared to the unbuffered DIMMs (UDIMM), registered and load-reduced DIMMs (RDIMMs and LRDIMMs, respectively) use only one pair of clocks and at least two chip-select signals `CS#[1:0]` in DDR3. Both RDIMMs and LRDIMMs require an additional parity signal for address, `RAS`, `CAS`, and `WE` signals.

Unbuffered DIMMs require unique chip-select, `ODT`, `CKE`, and clock pair signals for every rank on the DIMM. RDIMMs differ slightly in that only one clock pair is required for the entire module and a minimum of two chip select signals must be present (even for single-rank DIMMs) for programming of the RDIMM buffer.

LRDIMMs expand on the operation of RDIMMs by buffering the `DQ/DQS` bus. Only one electrical load is presented to the controller regardless of the number of ranks, therefore only one clock enable and ODT signal are required for LRDIMMs, regardless of the number of physical ranks. Because the number of physical ranks may exceed the number of physical chip-select signals, LRDIMMs provide a feature known as rank multiplication, which aggregates two or four physical ranks into one larger logical rank. Refer to LRDIMM buffer documentation for details on rank multiplication.

Both RDIMMs and LRDIMMs require an additoinal parity input for address, `RAS#`, `CAS#`, and `WE#` signals. A parity error signal is asserted by the module whenever a parity error is detected.

Table 3–2 shows UDIMM and RDIMM pin options.

Table 3–2. UDIMM and RDIMM Pin Options

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)
Data	72 bit DQ[71:0] = {CB[7:0], DQ[63:0]}	72 bit DQ[71:0] = {CB[7:0], DQ[63:0]}	72 bit DQ[71:0] = {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}
Data Mask	DM[8:0]	DM[8.0]	DM[8.0]	DM[8.0]
Data Strobe (1)	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]
Address	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]
Clock	CK0/CK0#	CK0/CK0#, CK1/CK1#	CK0/CK0#	CK0/CK0#
Command	ODT, CS#, CKE, RAS#, CAS#, WE#	ODT[1:0], CS#[1:0], CKE[1:0], RAS#, CAS#, WE#	ODT, CS#[1:0], CKE, RAS#, CAS#, WE#	ODT[1:0], CS#[1:0], CKE[1:0], RAS#, CAS#, WE#
Parity	—	—	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT
Other Pins	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#

Note to Table 3–2:

(1) DQS#[8:0] is optional in DDR2 SDRAM and is not supported in DDR SDRAM interfaces.

Table 3–3 shows LRDIMM pin options.

Table 3–3. LRDIMM Pin Options (Part 1 of 2)

Pins	LRDIMM Pins (2R)	LRDIMM (4R, RM=1)	LRDIMM Pins (4R, RM=2)	LRDIMM Pins (8R, RM=2)	LDIMM Pins (8R, RM=4)
Data	72 bit DQ [71:0]={CB [7:0], DQ [63:0]}	72 bit DQ [71:0]={CB [7:0], DQ [63:0]}	72 bit DQ [71:0]={CB [7:0], DQ [63:0]}	72 bit DQ [71:0]={CB [7:0], DQ [63:0]}	72 bit DQ [71:0]={CB [7:0], DQ [63:0]}
Data Mask	—	—	—	—	—
Data Strobe	DQS[17:0] and DQS#[17:0]	DQS[17:0] and DQS#[17:0]	DQS[17:0] and DQS#[17:0]	DQS[17:0] and DQS#[17:0]	DQS[17:0] and DQS#[17:0]
Address	BA[2:0], A[15:0]- 2GB:A[13:0] 4GB:A[14:0] 8GB:A[15:0]	BA[2:0], A[15:0]- 2GB:A[13:0] 4GB:A[14:0] 8GB:A[15:0]	BA[2:0], A[16:0]- 4GB:A[14:0] 8GB:A[15:0] 16GB:A[16:0]	BA[2:0], A[16:0]- 4GB:A[14:0] 8GB:A[15:0] 16GB:A[16:0]	BA[2:0], A[17:0]- 16GB:A[15:0] 32GB:A[16:0] 64GB:A[17:0]
Clock	CK0/CK0#	CK0/CK0#	CK0/CK0#	CK0/CK0#	CK0/CK0#
Command	ODT, CS[1:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[3:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[1:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[3:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[1:0]#, CKE, RAS#, CAS#, WE#
Parity	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT

Table 3–3. LRDIMM Pin Options (Part 2 of 2)

Pins	LRDIMM Pins (2R)	LRDIMM (4R, RM=1)	LRDIMM Pins (4R, RM=2)	LRDIMM Pins (8R, RM=2)	LDIMM Pins (8R, RM=4)
Other Pins	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#

Notes to Table 3–3:

(1) DM pins are not used for LRDIMMs because they are constructed using x4 components for greater memory density.

(2) CS2# is treated as A[16] and CS3# is treated as A[17] for certain rank multiplication configurations. Consult LRDIMM documentation for details.

QDR II+ and QDR II SRAM

This section provides a description of the clock, command, address, and data signals for QDR II and QDR II+ SRAM interfaces.

Clock Signals

QDR II+ and QDR II SRAM devices have three pairs of clocks:

- Input clocks K and K#
- Input clocks C and C#
- Echo clocks CQ and CQ#

The positive input clock, K, is the logical complement of the negative input clock, K#. Similarly, C and CQ are complements of C# and CQ#, respectively. With these complementary clocks, the rising edges of each clock leg latch the DDR data.

The QDR II+ and QDR II SRAM devices use the K and K# clocks for write access and the C and C# clocks for read accesses only when interfacing more than one QDR II+ or QDR II SRAM device. Because the number of loads that the K and K# clocks drive affects the switching times of these outputs when a controller drives a single QDR II+ or QDR II SRAM device, C and C# are unnecessary. This is because the propagation delays from the controller to the QDR II+ or QDR II SRAM device and back are the same. Therefore, to reduce the number of loads on the clock traces, QDR II+ and QDR II SRAM devices have a single clock mode, and the K and K# clocks are used for both reads and writes. In this mode, the C and C# clocks are tied to the supply voltage (V_{DD}).

CQ and CQ# are the source-synchronous output clocks from the QDR II or QDR II+ SRAM device that accompanies the read data.

The Altera device outputs the K and K# clocks, data, address, and command lines to the QDR II+ or QDR II SRAM device. For the controller to operate properly, the write data (D), address (A), and control signal trace lengths (and therefore the propagation times) should be equal to the K and K# clock trace lengths.

You can generate C, C#, K, and K# clocks using any of the PLL registers via the DDR registers. Because of strict skew requirements between K and K# signals, use adjacent pins to generate the clock pair. The propagation delays for K and K# from the FPGA to the QDR II+ or QDR II SRAM device are equal to the delays on the data and address (D, A) signals. Therefore, the signal skew effect on the write and read request operations is minimized by using identical DDR output circuits to generate clock and data inputs to the memory.

Command Signals

QDR II+ and QDR II SRAM devices use the write port select (WPSn) signal to control write operations and the read port select (RPSn) signal to control read operations. The byte write select signal (BWSn) is a third control signal that indicates to the QDR II+ or QDR II SRAM device which byte to write into the QDR II+ or QDR II SRAM device. You can use any of the FPGA's user I/O pins to generate control signals, preferably on the same side and the same bank. Assign the BWSn pin within the same DQS group as the corresponding the write data.

Address Signals

QDR II+ and QDR II SRAM devices use one address bus (A) for both read and write addresses. You can use any of the FPGA's user I/O pins to generate address signals, preferably on the same side and the same banks.

Data and QVLD Signals

QDR II+ and QDR II SRAM devices use two unidirectional data buses: one for writes (D) and one for reads (Q). The read data is edge-aligned with the CQ and CQ# clocks while the write data is center-aligned with the K and K# clocks (see Figure 3–3 and Figure 3–4).

Figure 3–3. Edge-aligned CQ and Q Relationship During QDR II+ SRAM Read

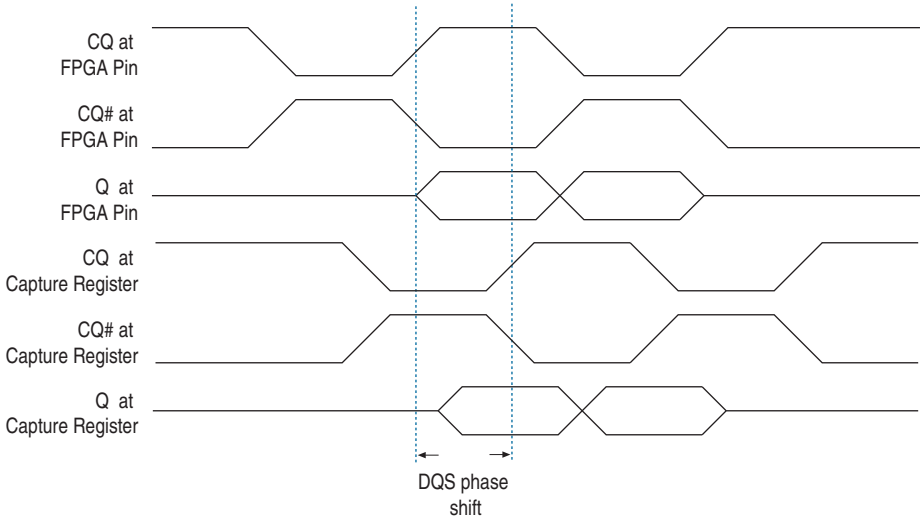
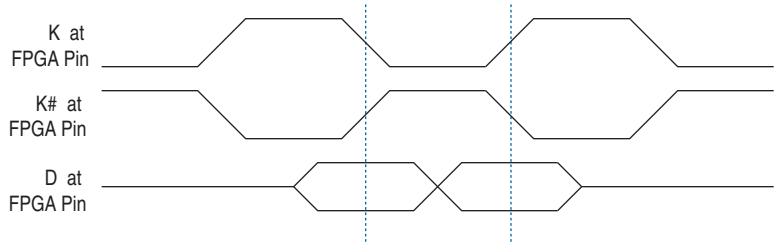


Figure 3–4. Centre-aligned K and D Relationship During QDR II+ SRAM Write



QDR II+ SRAM devices also have a QVLD pin that indicates valid read data. The QVLD signal is edge-aligned with the echo clock and is asserted high for approximately half a clock cycle before data is output from memory.

The Altera QDR II+ SRAM Controller with UniPHY IP does not use the QVLD signal.

RLDRAM II and RLDRAM 3

This section provides a description of the clock, command, address, and data signals for RLDRAM II and RLDRAM 3 interfaces.

Clock Signals

RLDRAM II and RLDRAM 3 devices use CK and CK# signals to clock the command and address bus in single data rate (SDR). There is one pair of CK and CK# pins per RLDRAM II or RLDRAM 3 device.

Instead of a strobe, RLDRAM II and RLDRAM 3 devices use two sets of free-running differential clocks to accompany the data. The DK and DK# clocks are the differential input data clocks used during writes while the QK or QK# clocks are the output data clocks used during reads. Even though QK and QK# signals are not differential signals according to the RLDRAM II and RLDRAM 3 data sheets, Micron treats these signals as such for their testing and characterization. Each pair of DK and DK#, or QK and QK# clocks are associated with either 9 or 18 data bits.

The exact clock-data relationships are as follows:

- RLDRAM II: For ×36 data bus width configuration, there are 18 data bits associated with each pair of write and read clocks. So, there are two pairs of DK and DK# pins and two pairs of QK or QK# pins.
- RLDRAM 3: For ×36 data bus width configuration, there are 18 data bits associated with each pair of write clocks. There are 9 data bits associated with each pair of read clocks. So, there are two pairs of DK and DK# pins and four pairs of QK and QK# pins.
- RLDRAM II: For ×18 data bus width configuration, there are 18 data bits per one pair of write clocks and nine data bits per one pair of read clocks. So, there is one pair of DK and DK# pins, but there are two pairs of QK and QK# pins.
- RLDRAM 3: For ×18 data bus width configuration, there are 9 data bits per one pair of write clocks and nine data bits per one pair of read clocks. So, there are two pairs of DK and DK# pins, and two pairs of QK and QK# pins
- RLDRAM II: For ×9 data bus width configuration, there are nine data bits associated with each pair of write and read clocks. So, there is one pair of DK and DK# pins and one pair of QK and QK# pins each.
- RLDRAM 3: RLDRAM 3 does not have the ×9 data bus width configuration.

There are t_{CKDK} timing requirements for skew between CK and DK or CK# and DK#.

For both RLDRAM II and RLDRAM 3, because of the loads on these I/O pins, the maximum frequency you can achieve depends on the number of memory devices you are connecting to the Altera device. Perform SPICE or IBIS simulations to analyze the loading effects of the pin-pair on multiple RLDRAM II devices.

Commands and Addresses

The CK and CK# signals clock the commands and addresses into the memory devices. These pins operate at single data rate using only one clock edge. RLDRAM II and RLDRAM 3 support both non-multiplexed and multiplexed addressing. Multiplexed addressing allows you to save a few user I/O pins while non-multiplexed addressing allows you to send the address signal within one clock cycle instead of two clock cycles. CS#, REF#, and WE# pins are input commands to the RLDRAM II or RLDRAM 3 device.

The commands and addresses must meet the memory address and command setup (t_{AS}, t_{CS}) and hold (t_{AH}, t_{CH}) time requirements.

UniPHY IP does not support multiplexed addressing.

Data, DM and QVLD Signals

The read data is edge-aligned with the QK or QK# clocks while the write data is center-aligned with the DK and DK# clocks (see Figure 3–5 and Figure 3–6). The memory controller shifts the DK or DK# signal to center align the DQ and DK or DK# signal during a write and to shift the QK signal during a read, so that read data (DQ or Q signals) and QK clock is center-aligned at the capture register. Altera devices use dedicated DQS phase-shift circuitry to shift the incoming QK signal during reads and use a PLL to center-align the DK and DK# signals with respect to the DQ signals during writes.

Figure 3–5. Edge-aligned DQ and QK Relationship During RLDRAM II or RLDRAM 3 Read

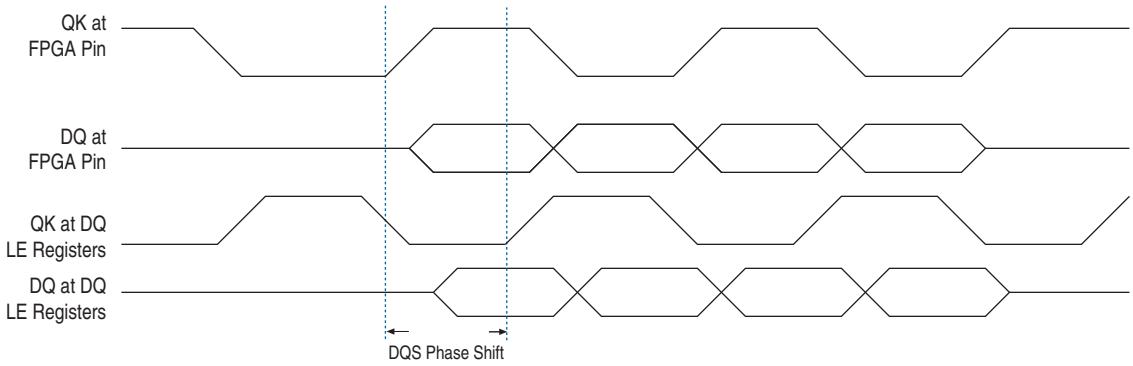
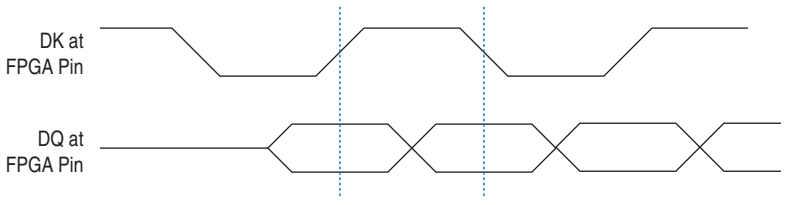


Figure 3–6. Centre-aligned DQ and DK Relationship During RLDRAM II or RLDRAM 3 Write



For RLDRAM II and RLDRAM 3, data mask (DM) pins are only used during a write. The memory controller drives the DM signal low when the write is valid and drives it high to mask the DQ signals. There is one DM pin per memory device.

The DM timing requirements at the input to the memory device are identical to those for DQ data. The DDR registers, clocked by the write clock, create the DM signals. This reduces any skew between the DQ and DM signals.

The RLDRAM II or RLDRAM 3 device's setup time (t_{DS}) and hold (t_{DH}) time for the write DQ and DM pins are relative to the edges of the DK or DK# clocks. The DK and DK# signals are generated on the positive edge of system clock, so that the positive edge of CK or CK# is aligned with the positive edge of DK or DK# respectively to meet the tCKDK requirement. The DQ and DM signals are clocked using a shifted clock so that the edges of DK or DK# are center-aligned with respect to the DQ and DM signals when they arrive at the RLDRAM II or RLDRAM 3 device.

The clocks, data, and DM board trace lengths should be tightly matched to minimize the skew in the arrival time of these signals.

RLDRAM II and RLDRAM 3 devices also have a QVLD pin indicating valid read data. The QVLD signal is edge-aligned with QK or QK# and is high approximately half a clock cycle before data is output from the memory.

The RLDRAM II Controller with UniPHY IP and the RLDRAM 3 PHY-only IP do not use the QVLD signal.

LPDDR2

This section describes the clock, command, address, and data signals for the LPDDR2 interface.

Clock Signal

CK and CKn are differential clock inputs to the LPDDR2 interface. All the double data rate (DDR) inputs are sampled on both the positive and negative edges of the clock. Single data rate (SDR) inputs, CSn and CKE, are sampled at the positive clock edge. The clock is defined as the differential pair which consists of CK and CKn. The positive clock edge is defined by the cross point of a rising CK and a falling CKn. The negative clock edge is defined by the cross point of a falling CK and a rising CKn.

The SDRAM data sheet specifies timing data for the following:

- tDSH is the DQS falling edge hold time from CK.
- tDSS is the DQS falling edge to the CK setup time.
- tDQSS is the Write command to the first DQS latching transition.
- tDQSCK is the DQS output access time from CK_t/CK_c.

Command and Address Signal

All LPDDR2 devices use double data rate architecture on the command/address bus to reduce the number of input pins in the system. The 10-bit command/address bus contains command, address, and bank/row buffer information. Each command uses one clock cycle, during which command information is transferred on both the positive and negative edges of the clock.

Data, Data Strobe, and DM Signals

LPDDR2 devices use bidirectional and differential data strobes. Differential DQS operation enables improved system timing due to reduced crosstalk and less simultaneous switching noise on the strobe output drivers. The DQ pins are also bidirectional. DQS is edge-aligned during the read data and centered during the write data.

DM is the input mask for the write data signal. Input data is masked when DM is sampled high coincident with that input data during a write access.

Maximum Number of Interfaces

Table 3–4 through Table 3–8 list the available device resources for DDR, DDR2, DDR3 SDRAM, RLDRAM II, QDR II and QDR II+ SRAM, and LPDDR2 SDRAM controller interfaces.

Unless otherwise noted, the calculation for the maximum number of interfaces is based on independent interfaces where the address or command pins are not shared. The maximum number of independent interfaces is limited to the number of PLLs each FPGA device has.

Timing closure depends on device resource and routing utilization. For more information about timing closure, refer to the *Area and Timing Optimization Techniques* chapter in the *Quartus II Handbook*.

You need to share DLLs if the total number of interfaces exceeds the number of DLLs available in a specific FPGA device. You may also need to share PLL clock outputs depending on your clock network usage, refer to "PLLs and Clock Networks" on page 3–50.

For information about the number of DQ and DQS in other packages, refer to the DQ and DQS tables in the relevant device handbook.

Table 3–4 describes the maximum number of ×8 DDR SDRAM components fit in the smallest and biggest devices and pin packages assuming the device is blank.

Each interface of size *n*, where *n* is a multiple of 8, consists of:

- *n* DQ pins (including error correction coding (ECC))
- *n*/8 DM pins
- *n*/8 DQS pins
- 18 address pins
- 6 command pins (CAS, RAS, WE, CKE, reset, and CS)
- 1 CK, CK# pin pair for up to every three ×8 DDR SDRAM components

Table 3–4. Maximum Number of DDR SDRAM Interfaces Supported per FPGA (Part 1 of 2)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GX	EP2AGX190 EP2AGX260	1,152	Four ×8 interfaces or one ×72 interface on each side (no DQ pins on left side)
	EP2AGX45 EP2AGX65	358	■ On top side, one ×16 interface ■ On bottom side, one ×16 interface ■ On right side (no DQ pins on left side), one ×8 interface
Arria II GZ	EP2AGZ300 EP2AGZ350 EP2AGZ225	F1,517	Four ×8 interfaces or one ×72 interface on each side
	EP2AGZ300 EP2AGZ350	F780	■ On top side, three ×8 interfaces or one ×64 interface ■ On bottom side, three ×8 interfaces or one ×64 interface ■ No DQ pins on the left and right sides
Cyclone III	EP3C120	780	■ Three ×16 interfaces on both top and bottom sides ■ Two ×16 interfaces on both right and left sides
	EP3C5	256	■ Two ×8 interfaces on both top and bottom sides ■ One ×8 interface on both right and left sides
Cyclone IV E	EP4CE115	780	■ One ×48 interface or two ×8 interfaces on both top and bottom sides ■ Four ×8 interfaces on both right and left sides
	EP4CE10	144	On top side, one ×8 interface with address pins wrapped around the left or right side
Cyclone IV GX	EP4CGX150	896	■ One ×48 interface or four ×8 interfaces on both top and bottom sides ■ On right side, three ×8 interfaces ■ No DQ pins on the left side
	EP4CGX22	324	■ One ×8 interface on both top and bottom sides ■ On right side, one ×8 interface with address pins wrapped around the top or bottom side ■ No DQ pins on the left side
Stratix III	EP3SL340	1,760	■ Two ×72 interfaces on both top and bottom sides ■ One ×72 interface on both right and left sides
	EP3SE50	484	■ Two ×8 interfaces on both top and bottom sides ■ Three ×8 interface on both right and left sides

Table 3–4. Maximum Number of DDR SDRAM Interfaces Supported per FPGA (Part 2 of 2)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Stratix IV	EP4SGX290 EP4SGX360 EP4SGX530	1,932	■ One ×72 interface on each side or ■ One ×72 interface on each side and two additional ×72 wraparound interfaces, only if sharing DLL and PLL resources
	EP4SE530 EP4SE820	1,760	
	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	780	■ Three ×8 interfaces or one ×64 interface on both top and bottom sides ■ On left side, one ×48 interface or two ×8 interfaces ■ No DQ pins on the right side

Table 3–5 lists the maximum number of ×8 DDR2 SDRAM components that can be fitted in the smallest and biggest devices and pin packages assuming the device is blank.

Each interface of size *n*, where *n* is a multiple of 8, consists of:

- *n* DQ pins (including ECC)
- *n*/8 DM pins
- *n*/8 DQS, DQSn pin pairs
- 18 address pins
- 7 command pins (CAS, RAS, WE, CKE, ODT, reset, and CS)
- 1 CK, CK# pin pair up to every three ×8 DDR2 components

Table 3–5. Maximum Number of DDR2 SDRAM Interfaces Supported per FPGA (Part 1 of 3)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GX	EP2AGX190 EP2AGX260	1,152	Four ×8 interfaces or one ×72 interface on each side (no DQ pins on left side)
	EP2AGX45 EP2AGX65	358	■ One ×16 interface on both top and bottom sides ■ On right side (no DQ pins on left side), one ×8 interface
Arria II GZ	EP2AGZ300 EP2AGZ350 EP2AGZ225	F1,517	Four ×8 interfaces or one ×72 interface on each side
	EP2AGZ300 EP2AGZ350	F780	■ Three ×8 interfaces or one ×64 interface on both top and bottom sides ■ No DQ pins on the left and right sides

Table 3–5. Maximum Number of DDR2 SDRAM Interfaces Supported per FPGA (Part 2 of 3)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria V	5AGXB1 5AGXB3 5AGXB5 5AGXB7 5AGTD3 5AGTD7	1,517	■ Two ×72 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5AGXA1 5AGXA3	672	■ One ×56 interface or two x24 interfaces on both top and bottom sides ■ One ×32 interface on the right side ■ No DQ pins on the left side
	5AGXA5 5AGXA7	672	■ One ×56 interface or two x24 interfaces on both top and bottom sides ■ No DQ pins on the left side
Arria V GZ	5AGZE5 5AGZE7	1,517	■ Three ×72 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5AGZE1 5AGZE3	780	■ On top side, two ×8 interfaces ■ On bottom side, four ×8 interfaces or one ×72 interface ■ No DQ pins on left and right sides
Cyclone III	EP3C120	780	■ Three ×16 interfaces on both top and bottom sides ■ Two ×16 interfaces on both left and right sides
	EP3C5	256	■ Two ×8 interfaces on both top and bottom sides ■ One ×8 interface on both right and left sides
Cyclone IV E	EP4CE115	780	■ One ×48 interface or two ×8 interfaces on both top and bottom sides ■ Four ×8 interfaces on both right and left sides
	EP4CE10	144	On top side, one ×8 interface with address pins wrapped around the left or right side
Cyclone IV GX	EP4CGX150	896	■ One ×48 interface or four ×8 interfaces on both top and bottom sides ■ On right side, three ×8 interfaces ■ No DQ pins on the left side
	EP4CGX22	324	■ One ×8 interface on both top and bottom sides ■ On right side, one ×8 interface with address pins wrapped around the top or bottom side ■ No DQ pins on the left side

Table 3–5. Maximum Number of DDR2 SDRAM Interfaces Supported per FPGA (Part 3 of 3)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Cyclone V	5CGTD9 5CEA9 5CGXC9	1,152	■ One ×72 interface or two ×32 interfaces on each of the top, bottom, and right sides ■ No DQ pins on the left side
	5CEA7 5CGTD7 5CGXC7	484	■ One ×48 interface or two ×16 interfaces on both top and bottom sides ■ One x8 interface on the right side ■ No DQ pins on the left side
Stratix III	EP3SL340	1,760	■ Two ×72 interfaces on both top and bottom sides ■ One ×72 interface on both right and left sides
	EP3SE50	484	■ Two ×8 interfaces on both top and bottom sides ■ Three ×8 interfaces on both right and left sides
Stratix IV	EP4SGX290 EP4SGX360 EP4SGX530	1,932	■ One ×72 interface on each side or ■ One ×72 interface on each side and two additional ×72 wraparound interfaces only if sharing DLL and PLL resources
	EP4SE530 EP4SE820	1,760	
	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	780	■ Three ×8 interfaces or one ×64 interface on top and bottom sides ■ On left side, one ×48 interface or two ×8 interfaces ■ No DQ pins on the right side
Stratix V	5SGXA5 5SGXA7	1,932	■ Three ×72 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5SGXA3 5SGXA4	780	■ On top side, two ×8 interfaces ■ On bottom side, four ×8 interfaces or one ×72 interface ■ No DQ pins on left and right sides

Table 3–6 lists the maximum number of ×8 DDR3 SDRAM components that can be fitted in the smallest and biggest devices and pin packages assuming the device is blank.

Each interface of size *n*, where *n* is a multiple of 8, consists of:

- *n* DQ pins (including ECC)
- *n*/8 DM pins
- *n*/8 DQS, DQSn pin pairs
- 17 address pins
- 7 command pins (CAS, RAS, WE, CKE, ODT, reset, and CS)
- 1 CK, CK# pin pair

Table 3–6. Maximum Number of DDR3 SDRAM Interfaces Supported per FPGA (Part 1 of 2)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GX	EP2AGX190 EP2AGX260	1,152	Four ×8 interfaces or one ×72 interface on each side (no DQ pins on left side)
	EP2AGX45 EP2AGX65	358	■ One ×16 interface on both top and bottom sides ■ On right side, one ×8 interface (no DQ pins on left side)
Arria II GZ	EP2AGZ300 EP2AGZ350 EP2AGZ225	F1,517	Four ×8 interfaces on each side
	EP2AGZ300 EP2AGZ350	F780	■ Three ×8 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
Arria V	5AGXB1 5AGXB3 5AGXB5 5AGXB7 5AGTD3 5AGTD7	1,517	■ Two ×72 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5AGXA1 5AGXA3	672	■ One ×56 interface or two ×24 interfaces on top and bottom sides ■ One ×32 interface on the right side ■ No DQ pins on the left side
	5AGXA5 5AGXA7	672	■ One ×56 interface or two ×24 interfaces on both top and bottom sides ■ No DQ pins on the left side
Arria V GZ	5AGZE5 5AGZE7	1,517	■ Two ×72 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5AGZE1 5AGZE3	780	■ On top side, two ×8 interfaces ■ On bottom side, four ×8 interfaces ■ No DQ pins on left and right sides
Cyclone V	5CGTD9 5CEA9 5CGXC9	1,152	■ One ×72 interface or two ×32 interfaces on each of the top, bottom, and right sides ■ No DQ pins on the left side
	5CEA7 5CGTD7 5CGXC7	484	■ One ×48 interface or two ×16 interfaces on both top and bottom sides ■ One x8 interface on the right side ■ No DQ pins on the left side

Table 3–6. Maximum Number of DDR3 SDRAM Interfaces Supported per FPGA (Part 2 of 2)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Stratix III	EP3SL340	1,760	■ Two ×72 interfaces on both top and bottom sides ■ One ×72 interface on both right and left sides
	EP3SE50	484	■ Two ×8 interfaces on both top and bottom sides ■ Three ×8 interfaces on both right and left sides
Stratix IV	EP4SGX290 EP4SGX360 EP4SGX530	1,932	■ One ×72 interface on each side or ■ One ×72 interface on each side and 2 additional ×72 wraparound interfaces only if sharing DLL and PLL resources
	EP4SE530 EP4SE820	1,760	
	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	780	■ Three ×8 interfaces or one ×64 interface on both top and bottom sides ■ On left side, one ×48 interface or two ×8 interfaces (no DQ pins on right side)
Stratix V	5SGXA5 5SGXA7	1,932	■ Two ×72 interfaces (800 MHz) on both top and bottom sides ■ No DQ pins on left and right sides
	5SGXA3 5SGXA4	780	■ On top side, two ×8 interfaces ■ On bottom side, four ×8 interfaces ■ No DQ pins on left and right sides

Table 3–7 on page 3–21 lists the maximum number of independent QDR II+ or QDR II SRAM interfaces that can be fitted in the smallest and biggest devices and pin packages assuming the device is blank.

One interface of ×36 consists of:

- 36 Q pins
- 36 D pins
- 1 K, K# pin pairs
- 1 CQ, CQ# pin pairs
- 19 address pins
- 4 BSWn pins
- WPS, RPS

One interface of ×9 consists of:

- 9 Q pins
- 9 D pins
- 1 K, K# pin pairs
- 1 CQ, CQ# pin pairs

- 21 address pins
- 1 BWSn pin
- WPS, RPS

Table 3–7. Maximum Number of QDR II and QDR II+ SRAM Interfaces Supported per FPGA (Part 1 of 2)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GX	EP2AGX190 EP2AGX260	1,152	One ×36 interface and one ×9 interface one each side
	EP2AGX45 EP2AGX65	358	One ×9 interface on each side (no DQ pins on left side)
Arria II GZ	EP2AGZ300 EP2AGZ350 EP2AGZ225	F1,517	■ Two ×36 interfaces and one ×9 interface on both top and bottom sides ■ Four ×9 interfaces on right and left sides
	EP2AGZ300 EP2AGZ350	F780	■ Three ×9 interfaces on both top and bottom sides ■ No DQ pins on right and left sides
Arria V	5AGXB1 5AGXB3 5AGXB5 5AGXB7 5AGTD3 5AGTD7	1,517	■ Two ×36 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5AGXA1 5AGXA3	672	■ Two ×9 interfaces on both top and bottom sides ■ One ×9 interface on the right side ■ No DQ pins on the left side
	5AGXA5 5AGXA7	672	■ Two ×9 interfaces on both top and bottom sides ■ No DQ pins on the left side
Arria V GZ	5AGZE5 5AGZE7	1,517	■ Two ×36 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5AGZE1 5AGZE3	780	■ On top side, one ×36 interface or three ×9 interfaces ■ On bottom side, two ×9 interfaces ■ No DQ pins on left and right sides
Stratix V	5SGXA5 5SGXA7	1,932	■ Two ×36 interfaces on both top and bottom sides ■ No DQ pins on left and right sides

Table 3–7. Maximum Number of QDR II and QDR II+ SRAM Interfaces Supported per FPGA (Part 2 of 2)

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Stratix III	EP3SL340	1,760	■ Two ×36 interfaces and one ×9 interface on both top and bottom sides ■ Five ×9 interfaces on both right and left sides
	EP3SE50 EP3SL50 EP3SL70	484	■ One ×9 interface on both top and bottom sides ■ Two ×9 interfaces on both right and left sides
Stratix IV	EP4SGX290 EP4SGX360 EP4SGX530	1,932	■ Two ×36 interfaces on both top and bottom sides ■ One ×36 interface on both right and left sides
	EP4SE530 EP4SE820	1,760	
	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	780	Two ×9 interfaces on each side (no DQ pins on right side)
Stratix V	5SGXA5 5SGXA7	1,932	■ Two ×36 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5SGXA3 5SGXA4	780	■ On top side, one ×36 interface or three ×9 interfaces ■ On bottom side, two ×9 interfaces ■ No DQ pins on left and right sides

Table 3–8 on page 3–23 lists the maximum number of independent RLDRAM II interfaces that can be fitted in the smallest and biggest devices and pin packages assuming the device is blank.

One common I/O ×36 interface consists of:

- 36 DQ
- 1 DM pin
- 2 DK, DK# pin pairs
- 2 QK, QK# pin pairs
- 1 CK, CK# pin pair
- 24 address pins
- 1 CS# pin
- 1 REF# pin
- 1 WE# pin
- 1 QVLD pin

One common I/O ×9 interface consists of:

- 9 DQ
- 1 DM pins
- 1 DK, DK# pin pair
- 1 QK, QK# pin pair
- 1 CK, CK# pin pair
- 25 address pins
- 1 CS# pin
- 1 REF# pin
- 1 WE# pin
- 1 QVLD pin

Table 3–8. Maximum Number of RLDRAM II Interfaces Supported per FPGA (Part 1 of 2)

Device	Device Type	Package Pin Count	Maximum Number of RLDRAM II CIO Interfaces
Arria II GZ	EP2AGZ300 EP2AGZ350 EP2AGZ225	F1,517	Two ×36 interfaces on each side
	EP2AGZ300 EP2AGZ350	F780	■ Three ×9 interfaces or one ×36 interface on both top and bottom sides ■ No DQ pins on the left and right sides
Arria V	5AGXB1 5AGXB3 5AGXB5 5AGXB7 5AGTD3 5AGTD7	1,517	■ Two ×36 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5AGXA1 5AGXA3	672	■ One ×36 interface on both top and bottom sides ■ One ×18 interface on the right side ■ No DQ pins on the left side
	5AGXA5 5AGXA7	672	■ One ×36 interface on both top and bottom sides ■ No DQ pins on the left side
Arria V GZ	5ZGZE5 5ZGZE7	1,517	■ Four ×36 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5AGZE1 5AGZE3	780	■ On top side, three ×9 interfaces or two ×36 interfaces ■ On bottom side, two ×9 interfaces or one ×36 interfaces ■ No DQ pins on left and right sides

Table 3–8. Maximum Number of RLDRAM II Interfaces Supported per FPGA (Part 2 of 2)

Device	Device Type	Package Pin Count	Maximum Number of RLDRAM II CIO Interfaces
Stratix III	EP3SL340	1,760	■ Four ×36 components on both top and bottom sides ■ Three ×36 interfaces on both right and left sides
	EP3SE50 EP3SL50 EP3SL70	484	One ×9 interface on both right and left sides
Stratix IV	EP4SGX290 EP4SGX360 EP4SGX530	1,932	■ Three ×36 interfaces on both top and bottom sides ■ Two ×36 interfaces on both right and left sides
	EP4SE530 EP4SE820	1,760	■ Three ×36 interfaces on each side
	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230	780	One ×36 interface on each side (no DQ pins on right side)
Stratix V	5SGXA5 5SGXA7	1,932	■ Four ×36 interfaces on both top and bottom sides ■ No DQ pins on left and right sides
	5SGXA3 5SGXA4	780	■ On top side, two ×9 interfaces or one ×18 interfaces ■ On bottom side, three ×9 interfaces or two ×36 interfaces ■ No DQ pins on left and right sides

Table 3–9 on page 3–25 lists the maximum number of x8 LPDDR2 SDRAM components that can fit in the smallest and largest devices and pin packages, assuming the device is blank.

Each interface of size *n*, where *n* is a multiple of 8, consists of:

- *n* DQ pins (including ECC)
- *n*/8 DM pins
- *n*/8 DQS, DQSn pin pairs
- 10 address pins
- 2 command pins (CKE and CSn)

- 1 CK, CK# pin pair up to every three x8 LPDDR2 components

Table 3–9. Maximum Number of LPDDR2 SDRAM Interfaces Supported per FPGA

Device	Device Type	Package Pin Count	Maximum Number of RLDRAM II CIO Interfaces
Arria V	5AGXB1 5AGXB3 5AGXB5 5AGXB7 5AGTD3 5AGTD7	1,517	■ One ×72 interface on both top and bottom sides ■ No DQ pins on the left and right sides
	5AGXA1 5AGXA3	672	■ One ×64 interface or two ×24 interfaces on both top and bottom sides ■ One ×32 interface on the right side
	5AGXA5 5AGXA7	672	■ One ×64 interface or two ×24 interfaces on both the top and bottom sides ■ No DQ pins on the left side
Cyclone V	5CGTD9 5CEA9 5CGXC9	1,152	■ One ×72 interface or two ×32 interfaces on each of the top, bottom, and right sides ■ No DQ pins on the left side
	5CEA7 5CGTD7 5CGXC7	484	■ One ×48 interface or two ×16 interfaces on both the top and bottom sides ■ One ×8 interface on the right side ■ No DQ pins on the left side

OCT Support for Arria II GX, Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices

This section is not applicable to Cyclone III and Cyclone IV devices because OCT is not used by the Altera IP.

If you use OCT for your memory interfaces with Cyclone III and Cyclone IV devices, refer to the *Device I/O Features* chapter in the *Cyclone III* or *Cyclone IV Device Handbook*.

If the memory interface uses any FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design, you need a calibration block for the OCT circuitry. This calibration block is not required to be within the same bank or side of the device as the memory interface pins. However, the block requires a pair of R_{UP} and R_{DN} or R_{ZQ} pins that must be placed within an I/O bank that has the same V_{CCIO} voltage as the V_{CCIO} voltage of the I/O pins that use the OCT calibration block.

The R_{ZQ} pin in Stratix V, Arria V, and Cyclone V devices is a dual functional pin that can also be used as `DQ` and `DQS` pins when it is not used to support OCT. You can use the `DQS` group in ×4 mode with non-differential `DQS` pins if the R_{ZQ} pin is part of a ×4 `DQS` group.

The R_{UP} and R_{DN} pins in Arria II GX, Arria II GZ, Stratix III, and Stratix IV devices are dual functional pins that can also be used as DQ and DQS pins in when they are not used to support OCT, giving the following impacts on your DQS groups:

- If the R_{UP} and R_{DN} pins are part of a ×4 DQS group, you cannot use that DQS group in ×4 mode.
- If the R_{UP} and R_{DN} pins are part of a ×8 DQS group, you can only use this group in ×8 mode if any of the following conditions apply:
  - You are not using DM or BWSn pins.
  - You are not using a ×8 or ×9 QDR II and QDR II+ SRAM devices, as the R_{UP} and R_{DN} pins may have dual purpose function as the CQn pins. In this case, pick different pin locations for R_{UP} and R_{DN} pins, to avoid conflict with memory interface pin placement. You have the choice of placing the R_{UP} and R_{DN} pins in the same bank as the write data pin group or address and command pin group.
  - You are not using complementary or differential DQS pins.

The QDR II and QDR II+ SRAM controller with UniPHY do not support ×8 QDR II and QDR II+ SRAM devices in the Quartus II software.

A DQS/DQ ×8/×9 group in Arria II GZ, Stratix III, and Stratix IV devices comprises 12 pins. A typical ×8 memory interface consists of one DQS, one DM, and eight DQ pins which add up to 10 pins. If you choose your pin assignment carefully, you can use the two extra pins for R_{UP} and R_{DN}. However, if you are using differential DQS, you do not have enough pins for R_{UP} and R_{DN} as you only have one pin leftover. In this case, as you do not have to put the OCT calibration block with the DQS or DQ pins, you can pick different locations for the R_{UP} and R_{DN} pins. As an example, you can place it in the I/O bank that contains the address and command pins, as this I/O bank has the same V_{CCIO} voltage as the I/O bank containing the DQS and DQ pins.

There is no restriction when using ×16/×18 or ×32/×36 DQS groups that include the ×4 groups when pin members are used as R_{UP} and R_{DN} pins, as there are enough extra pins that can be used as DQS or DQ pins.

You need to pick your DQS and DQ pins manually for the ×8, ×9, ×16 and ×18, or ×32 and ×36 groups, if they are using R_{UP} and R_{DN} pins within the group. The Quartus II software may not place these pins optimally and may be unable to fit the design.

General Pin-out Guidelines

Altera recommends that you place all the pins for one memory interface (attached to one controller) on the same side of the device. For projects where I/O availability is a challenge and therefore it is necessary spread the interface on two sides, for optimal performance, place all the input pins on one side, and the output pins on an adjacent side of the device along with their corresponding source-synchronous clock.

For a unidirectional data bus as in QDR II and QDR II+ SRAM interfaces, do not split a read data pin group or a write data pin group onto two sides. It is also strongly recommended not to split the address and command group onto two sides either, especially when you are interfacing with QDR II and QDR II+ SRAM burst-length-of-two devices, where the address signals are double data rate also. Failure to adhere to these rules may result in timing failure.

In addition, there are some exceptions for the following interfaces:

- ×36 emulated QDR II and QDR II+ SRAM in Arria II, Stratix III, and Stratix IV devices.
- RLDRAM II and RLDRAM 3 CIO devices
- QDR II/+ SDRAM burst-length-of-two devices.

CAUTION

You need to compile the design in Quartus II to ensure that you are not violating signal integrity and Quartus II placement rules, which is critical when you have transceivers in the same design.

The following list gives some general guidelines on how to place pins optimally for your memory interfaces:

1. For Arria II GZ, Arria V, Cyclone V, Stratix III, Stratix IV, and Stratix V designs, if you are using OCT, the R_{UP} and R_{DN}, or R_{ZQ} pins need to be in any bank with the same I/O voltage as your memory interface signals and often use two DQS and DQ pins from a group. If you decide to place the R_{UP} and R_{DN}, or R_{ZQ} pins in a bank where the DQS and DQ groups are used, place these pins first and then see how many DQ pins you have left after, to find out if your data pins can fit in the remaining pins. Refer to "OCT Support for Arria II GX, Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices" on page 3–25.
2. Use the PLL that is on the same side of the memory interface. If the interface is spread out on two adjacent sides, you may use the PLL that is located on either adjacent side. You must use the dedicated input clock pin to that particular PLL as the reference clock for the PLL as the input of the memory interface PLL cannot come from the FPGA clock network.
3. The Altera IP uses the output of the memory interface PLL for the DLL input reference clock. Therefore, ensure you pick a PLL that can directly feed a suitable DLL.

   Alternatively, you can use an external pin to feed into the DLL input reference clock. The available pins are also listed in the *External Memory Interfaces* chapter of the relevant device family handbook. You can also activate an unused PLL clock outputs, set it at the desired DLL frequency, and route it to a PLL dedicated output pin. Connect a trace on the PCB from this output pin to the DLL reference clock pin, but be sure to include any signal integrity requirements such as terminations.

4. Read data pins require the usage of DQS and DQ group pins to have access to the DLL control signals.

☞ In addition, QVLD pins in RLDRAM II and RLDRAM 3, and QDR II+ SRAM must use DQS group pins, when the design uses the QVLD signal. None of the Altera IP uses QVLD pins as part of read capture, so theoretically you do not need to connect the QVLD pins if you are using the Altera solution. It is good to connect it anyway in case the Altera solution gets updated to use QVLD pins.

5. In differential clocking (DDR3/DDR2 SDRAM, RLDRAM II, and RLDRAM 3 interfaces), connect the positive leg of the read strobe or clock to a DQS pin, and the negative leg of the read strobe or clock to a DQSn pin. For QDR II or QDR II+ SRAM devices with 2.5 or 1.5 cycles of read latency, connect the CQ pin to a DQS pin, and the CQn pin to a CQn pin (and not the DQSn pin). For QDR II or QDR II+ SRAM devices with 2.0 cycles of read latency, connect the CQ pin to a CQn pin, and the CQn pin to a DQS pin.
6. Write data (if unidirectional) and data mask pins (DM or BWSn) pins must use DQS groups. While the DLL phase shift is not used, using DQS groups for write data minimizes skew, and must use the SW and TCCS timing analysis methodology.
7. Assign the write data strobe or write data clock (if unidirectional) in the corresponding DQS/DQSn pin with the write data groups that place in DQ pins (except in RLDRAM II and RLDRAM 3 CIO devices, refer to "Pin-out Rule Exceptions" on page 3–29)

☞ When interfacing with a DDR, or DDR2, or DDR3 SDRAM without leveling, put the three CK and CK# pairs in a single ×4 DQS group to minimize skew between clocks and maximize margin for the t_{DQSS}, t_{DSS}, and t_{DSH} specifications from the memory devices.

8. Assign any address pins to any user I/O pin. To minimize skew within the address pin group, you should assign the address and command pins in the same bank or side of the device.
9. Assign the command pins to any I/O pins and assign the pins in the same bank or device side as the other memory interface pins, especially address and memory clock pins. The memory device usually uses the same clock to register address and command signals.

☞ In QDR II and QDR II+ SRAM interfaces where the memory clock also registers the write data, assign the address and command pins in the same I/O bank or same side as the write data pins, to minimize skew.

☞ For more information about assigning memory clock pins for different device families and memory standards, refer to "Pin Connection Guidelines Tables" on page 3–37.

Pin-out Rule Exceptions

The following sub sections described exceptions to the rule described in the "General Pin-out Guidelines" on page 3–26.

Exceptions for ×36 Emulated QDR II and QDR II+ SRAM Interfaces in Arria II, Stratix III and Stratix IV Devices

A few packages in the Arria II, Arria v GZ, Stratix III, Stratix IV, and Stratix V device families do not offer any ×32/×36 DQS groups where one read clock or strobe is associated with 32 or 36 read data pins. This limitation exists in the following I/O banks:

- All I/O banks in U358- and F572-pin packages for all Arria II GX devices
- All I/O banks in F484-pin packages for all Stratix III devices
- All I/O banks in F780-pin packages for all Arria II GZ, Stratix III, and Stratix IV devices; top and side I/O banks in F780-pin packages for all Stratix V and Arria V GZ devices
- All I/O banks in F1152-pin packages for all Arria II GZ, Stratix III, and Stratix IV devices, except EP4SGX290, EP4SGX360, EP4SGX530, EPAGZ300, and EPAGZ350 devices
- Side I/O banks in F1517- and F1760-pin packages for all Stratix III devices
- All I/O banks in F1517-pin for EP4SGX180, EP4SGX230, EP4S40G2, EP4S40G5, EP4S100G2, EP4S100G5, and EPAGZ225 devices
- Side I/O banks in F1517-, F1760-, and F1932-pin packages for all Arria II GZ and Stratix IV devices

This limitation limits support for ×36 QDR II and QDR II+ SRAM devices. To support these memory devices, this following section describes how you can emulate the ×32/×36 DQS groups for these devices.

The maximum frequency supported in ×36 QDR II and QDR II+ SRAM interfaces using ×36 emulation is lower than the maximum frequency when using a native ×36 DQS group.

The F484-pin package in Stratix III devices cannot support ×32/×36 DQS group emulation, as it does not support ×16/×18 DQS groups.

To emulate a ×32/×36 DQS group, combine two ×16/×18 DQS groups together. For ×36 QDR II and QDR II+ SRAM interfaces, the 36-bit wide read data bus uses two ×16/×18 groups; the 36-bit wide write data uses another two ×16/×18 groups or four ×8/×9 groups. The CQ and CQn signals from the QDR II and QDR II+ SRAM device traces are then split on the board to connect to two pairs of CQ/CQn pins in the FPGA. You may then need to split the QVLD pins also (if you are connecting them). These connections are the only connections on the board that you need to change for this implementation. There is still only one pair of K and Kn# connections on the board from the FPGA to the memory (see Figure 3–7). Use an external termination for the CQ/CQn signals at the FPGA end. You can use the FPGA OCT features on the other QDR II interface signals with ×36 emulation. In addition, there may be extra assignments to be added with ×36 emulation.

☞ Other QDR II and QDR II+ SRAM interface rules also apply for this implementation.

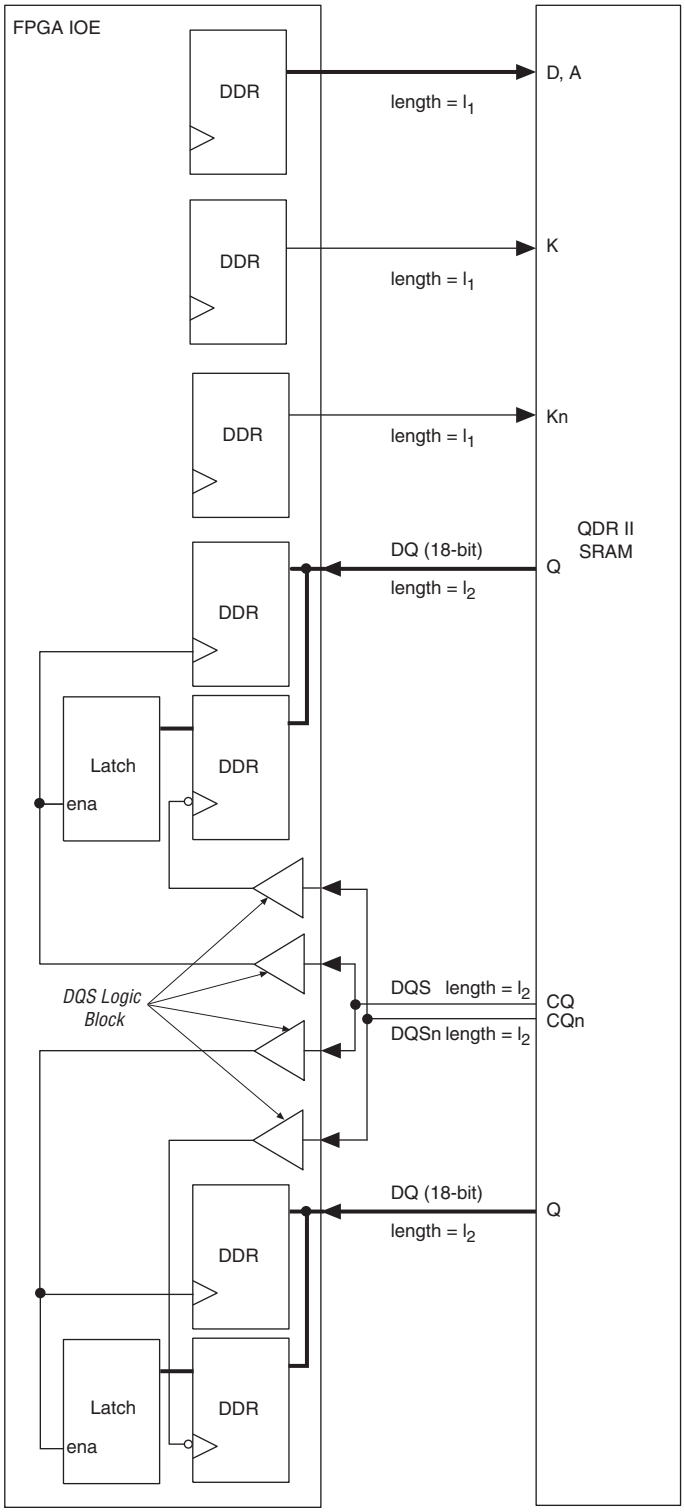
You may also combine four ×9 DQS groups (or two ×9 DQS groups and one ×18 group) on the same side of the device, if not the same I/O bank, to emulate a x36 write data group, if you need to fit the QDR II interface in a particular side of the device that does not have enough ×18 DQS groups available for write data pins. Altera does not recommend using ×4 groups as the skew may be too large, as you need eight ×4 groups to emulate the ×36 write data bits.

You cannot combine four ×9 groups to create a ×36 read data group as the loading on the CQ pin is too large and hence the signal is degraded too much.

When splitting the CQ and CQn signals, the two trace lengths that go to the FPGA pins must be as short as possible to reduce reflection. These traces must also have the same trace delay from the FPGA pin to the Y or T junction on the board. The total trace delay from the memory device to each pin on the FPGA should match the Q trace delay (I_2).

☞ You must match the trace delays. However, matching trace length is only an approximation to matching actual delay.

Figure 3–7. Board Trace Connection for Emulated x36 QDR II and QDR II+ SRAM Interface



Timing Impact on x36 Emulation

With ×36 emulation, the CQ/CQn signals are split on the board, so these signals see two loads (to the two FPGA pins)—the DQ signals still only have one load. The difference in loading gives some slew rate degradation, and a later CQ/CQn arrival time at the FPGA pin.

The slew rate degradation factor is taken into account during timing analysis when you indicate in the UniPHY Preset Editor that you are using ×36 emulation mode. However, you must determine the difference in CQ/CQn arrival time as it is highly dependent on your board topology.

The slew rate degradation factor for ×36 emulation assumes that CQ/CQn has a slower slew rate than a regular ×36 interface. The slew rate degradation is assumed not to be more than 500 ps (from 10% to 90% V_{CCIO} swing). You may also modify your board termination resistor to improve the slew rate of the ×36-emulated CQ/CQn signals. If your modified board does not have any slew rate degradation, you do not need to enable the ×36 emulation timing in the UniPHY-based controller MegaWizard™ interface.

For more information about how to determine the CQ/CQn arrival time skew, refer to "Determining the CQ/CQn Arrival Time Skew" on page 3–33.

Because of this effect, the maximum frequency supported using x36 emulation is lower than the maximum frequency supported using a native x36 DQS group.

Rules to Combine Groups

For devices that do not have four ×16/×18 groups in a single side of the device to form two ×36 groups for read and write data, you can form one ×36 group on one side of the device, and another ×36 group on the other side of the device. All the read groups have to be on the same edge (column I/O or row I/O) and all write groups have to be on the same type of edge (column I/O or row I/O), so you can have an interface with the read group in column I/O and the write group in row I/O. The only restriction is that you cannot combine an ×18 group from column I/O with an ×18 group from row IO to form a x36-emulated group.

For vertical migration with the ×36 emulation implementation, check if migration is possible and enable device migration in the Quartus II software.

I/O bank 1C in both Stratix III and Stratix IV devices has dual-function configuration pins. Some of the DQS pins may not be available for memory interfaces if these are used for device configuration purposes.

Each side of the device in these packages has four remaining ×8/×9 groups. You can combine four of the remaining for the write side (only) if you want to keep the ×36 QDR II and QDR II+ SRAM interface on one side of the device, by changing the **Memory Interface Data Group** default assignment, from the default **18** to **9**.

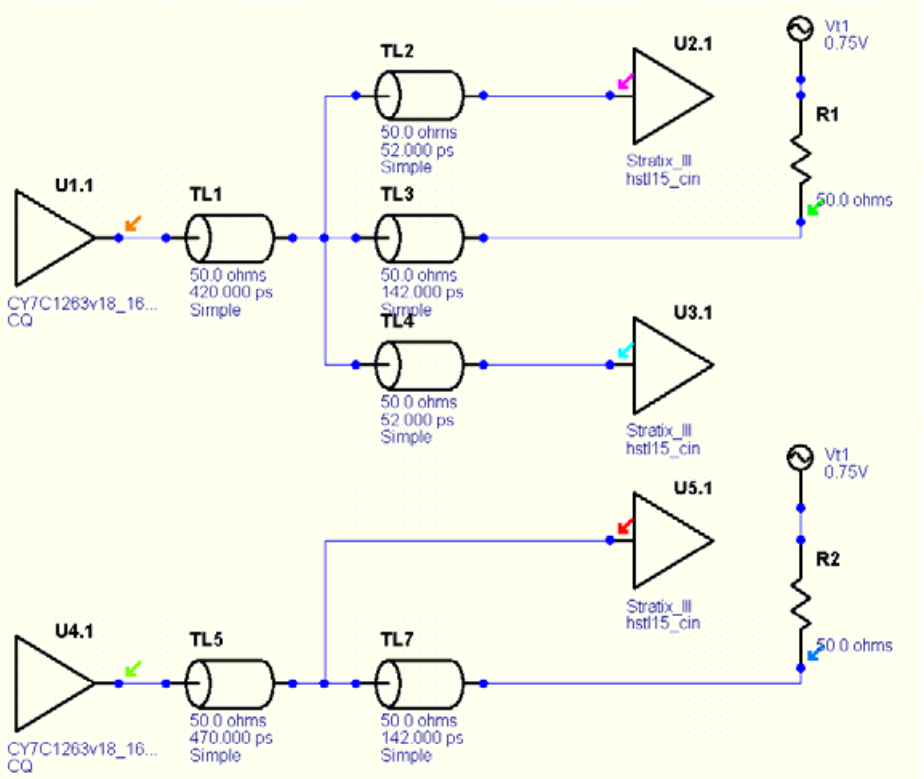
The ALTMEMPHY megafunction does not support ×36 mode emulation wraparound interface, where the ×36 group consists of a ×18 group from the top/bottom I/O bank and a ×18 group from the side I/O banks.

For more information about rules to combine groups for your target device, refer to the External Memory Interfaces chapter in the respective device handbooks.

Determining the CQ/CQn Arrival Time Skew

Before compiling a design in Quartus II, you need to determine the CQ/CQn arrival time skew based on your board simulation. You then need to apply this skew in the **report_timing.tcl** file of your QDR II and QDR II+ SRAM interface in the Quartus II software. Figure 3–8 shows an example of a board topology comparing an emulated case where CQ is double-loaded and a non-emulated case where CQ only has a single load.

Figure 3–8. Board Simulation Topology Example

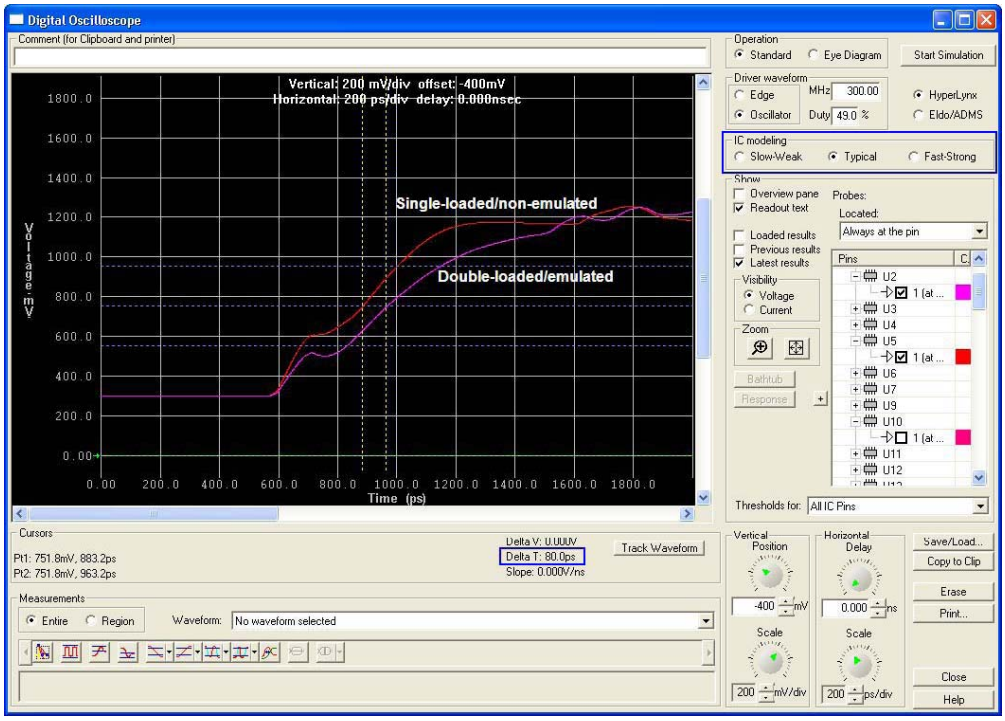


Run the simulation and look at the signal at the FPGA pin. Figure 3–9 shows an example of the simulation results from Figure 3–8. As expected, the double-loaded emulated signal, in pink, arrives at the FPGA pin later than the single-loaded signal, in red. You then need to calculate the difference of this arrival time at V_{REF} level (0.75 V in this case). Record the skew and rerun the simulation in the other two cases (slow-weak and fast-strong). To pick the largest and smallest skew to be included in Quartus II timing analysis, follow these steps:

1. Open the <*variation_name*>**_report_timing.tcl** and search for `tmin_additional_dqs_variation`.
2. Set the minimum skew value from your board simulation to `tmin_additional_dqs_variation`.

3. Set the maximum skew value from your board simulation to `tmax_additional_dqs_variation`.
4. Save the **.tcl** file.

Figure 3–9. Board Simulation Results



Exceptions for RLDRAM II and RLDRAM 3 Interfaces

RLDRAM II and RLDRAM 3 CIO devices have one bidirectional bus for the data, but there are two different sets of clocks: one for read and one for write. As the QK and QK# already occupies the DQS and DQSn pins needed for read, placement of DK and DK# pins are restricted due to the limited number of pins in the FPGA. This limitations causes the exceptions to the previous rules, which are discussed in the following sections.

The address or command pins of RLDRAM II must be placed in a DQ-group because these pins are driven by the PHY clock. Half-rate RLDRAM II interfaces and full-rate RLDRAM 3 interfaces use the PHY clock for both the DQ pins and the address or command pins.

Interfacing with ×9 RLDRAM II CIO Devices

RLDRAM II devices have the following pins:

- 2 pins for QK and QK# signals
- 9 DQ pins (in a ×8/×9 DQS group)
- 2 pins for DK and DK# signals
- 1 DM pin

- 14 pins total (15 if you have a QVLD)

In the FPGA, the ×8/×9 DQS group consists of 12 pins: 2 for the read clocks and 10 for the data. In this case, move the QVLD (if you want to keep this connected even though this is not used in the Altera memory interface solution) and the DK and DK# pins to the adjacent DQS group. If that group is in use, move to any available user I/O pins in the same I/O bank.

RLDRAM 3 devices do not have the x9 configuration.

Interfacing with ×18 RLDRAM II and RLDRAM 3 CIO Devices

RLDRAM II devices have the following pins:

- 4 pins for QK/QK# signals
- 18 DQ pins (in ×8/×9 DQS group)
- 2 pins for DK/DK# signals
- 1 DM pin
- 25 pins total (26 if you have a QVLD)

In the FPGA, you use two ×8/×9 DQS group totaling 24 pins: 4 for the read clocks and 18 for the read data.

Each ×8/×9 group has one DQ pin left over that can either use QVLD or DM, so one ×8/×9 group has the DM pin associated with that group and one ×8/×9 group has the QVLD pin associated with that group.

RLDRAM 3 devices have the following pins:

- 4 pins for QK/QK# signals
- 18 DQ pins (in ×8/×9 DQS group)
- 4 pins for DK/DK# signals
- 2 DM pins
- 28 pins total (29 if you have a QVLD)

In the FPGA, you use two ×8/×9 DQS group totaling 24 pins: 4 for the read clocks and 18 for the read data.

Each ×8/×9 group has one DQ pin left over that can either use QVLD or DM, so one ×8/×9 group has the DM pin associated with that group and one ×8/×9 group has the QVLD pin associated with that group.

Interfacing with RLDRAM II and RLDRAM 3 ×36 CIO Devices

RLDRAM II devices have the following pins:

- 4 pins for QK/QK# signals
- 36 DQ pins (in x16/x18 DQS group)
- 4 pins for DK/DK# signals
- 1 DM pins
- 46 pins total (47 if you have a QVLD)

In the FPGA, you use two ×16/×18 DQS groups totaling 48 pins: 4 for the read clocks and 36 for the read data. Configure each ×16/×18 DQS group to have:

- Two QK/QK# pins occupying the DQS/DQSn pins
- Pick two DQ pins in the ×16/×18 DQS groups that are DQS and DQSn pins in the ×4 or ×8/×9 DQS groups for the DK and DK# pins
- 18 DQ pins occupying the DQ pins
- There are two DQ pins leftover that you can use for QVLD or DM pins. Put the DM pin in the group associated with `DK[1]` and the QVLD pin in the group associated with `DK[0]`.

Check that DM is associated with `DK[1]` for your chosen memory component.

RLDRAM 3 devices have the following pins:

- 8 pins for QK/QK# signals
- 36 DQ pins (in x8/x9 DQS group)
- 4 pins for DK/DK# signals
- 2 DM pins
- 48 pins total (49 if you have a QVLD)

In the FPGA, you use four ×8/×9 DQS groups.

In addition, observe the following placement rules for RLDRAM 3 interfaces:

For ×18 devices:

- Use two ×8/×9 DQS groups. Assign the QK/QK# pins and the DQ pins of the same read group to the same DQS group.
- DQ, DM, and DK/DK# pins belonging to the same write group should be assigned to the same I/O sub-bank, for timing closure.
- Whenever possible, assign CK/CK# pins to the same I/O sub-bank as the DK/DK# pins, to improve tCKDK timing.

For ×36 devices:

- Use four ×8/×9 DQS groups. Assign the QK/QK# pins and the DQ pins of the same read group to the same DQS group.
- DQ, DM, and DK/DK# pins belonging to the same write group should be assigned to the same I/O sub-bank, for timing closure.

- Whenever possible, assign CK/CK# pins to the same I/O sub-bank as the DK/DK# pins, to improve tCKDK timing

Exceptions for QDR II and QDR II+ SRAM Burst-length-of-two Interfaces

If you are using the QDR II and QDR II+ SRAM burst-length-of-two devices, you may want to place the address pins in a DQS group to minimize skew, because these pins are now double data rate too. The address pins typically do not exceed 22 bits, so you may use one ×18 DQS groups or two ×9 DQS groups on the same side of the device, if not the same I/O bank. In Arria V GZ, Stratix III, Stratix IV, and Stratix V devices, one ×18 group typically has 22 DQ bits and 2 pins for DQS/DQSn pins, while one ×9 group typically has 10 DQ bits with 2 pins for DQS/DQSn pins. Using ×4 DQS groups should be a last resort.

Pin Connection Guidelines Tables

Table 3–10 on page 3–38 lists the FPGA pin utilization for DDR, DDR2, and DDR3 SDRAM without leveling interfaces.

Table 3–10. FPGA Pin Utilization for DDR, DDR2, and DDR3 SDRAM without Leveling Interfaces (Part 1 of 3)

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization			
		Arria II GX	Cyclone III and Cyclone IV	Arria II GZ, Stratix III, and Stratix IV	Arria V, Cyclone V, and Stratix V
Memory System Clock	CK and CK# (1), (2)	If you are using single-ended DQS signaling, place any unused DQ or DQS pins with DIFFOUT capability located in the same bank or on the same side as the data pins. If you are using differential DQS signaling in ALTMEMPHY IP, the first CK/CK# pair must use any unused DQ or DQS pins with DIFFIO_RX or DIFFIN capability in the same bank or on the same side as the data pins. You can use either side of the device for wraparound interfaces. If there are other CK/CK# pairs, place them on DIFFOUT in the same single DQ group of adequate width to minimize skew. For example, DIMMs requiring three memory clock pin-pairs must use a ×4 DQS group, where the mem_clk[0] and mem_clk n[0] use the DIFF_RX or DIFFIN pins in the group, while mem_clk[2:1] and mem_clk n[2:1] pins use DIFFOUT pins in that DQS group.	Place any differential I/O pin pair (DIFFIO) in the same bank or on the same side as the data pins. You can use either side of the device for wraparound interfaces. The first CK/CK# pair cannot be placed in the same row or column pad group as any of the DQ pins (Figure 3–10 and Figure 3–11).	If you are using single-ended DQS signaling, place any DIFFOUT pins in the same bank or on the same side as the data pins If you are using differential DQS DQS signaling in ALTMEMPHY IP, the first CK/CK# pair must use any unused DIFFIO_RX pins in the same bank or on the same side as the data pins. You can use either side of the device for wraparound interfaces. If there are other CK/CK# pairs, place them on DIFFOUT in the same single DQ group of adequate width to minimize skew. For example, DIMMs requiring three memory clock pin-pairs must use a ×4 DQS group, where mem_clk[0] and mem_clk_n[0] pins use the DIFFIO_RX or DIFFIN pins in that group, while, mem_clk[2:1] and mem_clk_n[2:1] pins use DIFFOUT pins in that DQS group.	If you are using single-ended DQS signaling, place any unused DQ or DQS pins with DIFFOUT capability in the same bank or on the same side as the data pins. If you are using differential DQS signaling, place any unused DQ or DQS pins with DIFFOUT capability for the mem_clk[n:0] and mem_clk_n[n:0] signals (where n>=0). CK pins must use the differential pairs where both pins of the pair are DQ or DQS pins. Do not place CK and CK# pins in the same group as any other DQ or DQS pins.

Table 3–10. FPGA Pin Utilization for DDR, DDR2, and DDR3 SDRAM without Leveling Interfaces (Part 2 of 3)

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization			
		Arria II GX	**Cyclone III and Cyclone IV**	**Arria II GZ, Stratix III, and Stratix IV**	**Arria V, Cyclone V, and Stratix V**
		If you are using differential DQS signaling in UniPHY IP, place on `DIFFOUT` in the same single DQ group of adequate width to minimize skew.		If you are using differential DQS signaling in UniPHY IP, place any `DIFFOUT` pins in the same bank or on the same side as the data pins. If there are multiple CK/CK# pairs, place them on `DIFFOUT` in the same single DQ group of adequate width. For example, DIMMs requiring three memory clock pin-pairs must use a ×4 DQS group.	If there are multiple CK and CK# pin pairs, place them on `DIFFOUT` in the same single DQ group of adequate width.
Clock Source	—	Dedicated PLL clock input pin with direct connection to the PLL (not using the global clock network). For Arria II GX, Arria II GZ, Arria V GZ, Stratix III, Stratix IV and Stratix V Devices, also ensure that the PLL can supply the input reference clock to the DLL. Otherwise, refer to alternative DLL input reference clocks ("General Pin-out Guidelines" on page 3–26).			
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal.			
Data	DQ	DQ in the pin table, marked as Q in the Quartus II Pin Planner. Each DQ group has a common background color for all of the DQ and DM pins, associated with DQS (and DQSn) pins.			
Data mask	DM				
Data strobe	DQS or DQS and DQSn (DDR2 and DDR2 SDRAM only)	DQS (S in the Quartus II Pin Planner) for single-ended DQS signaling or DQS and DQSn (S and Sbar in the Quartus II Pin Planner) for differential DQS signaling. DDR2 supports either single-ended or differential DQS signaling. However, Cyclone III and Cyclone IV devices do not support differential DQS signaling. DDR3 SDRAM mandates differential DQS signaling.			

Table 3–10. FPGA Pin Utilization for DDR, DDR2, and DDR3 SDRAM without Leveling Interfaces (Part 3 of 3)

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization			
		Arria II GX	Cyclone III and Cyclone IV	Arria II GZ, Stratix III, and Stratix IV	Arria V, Cyclone V, and Stratix V
Address and command	A[], BA[], CAS#, CKE, CS#, ODT, RAS#, WE#, RESET#	Any user I/O pin. To minimize skew, you must place the address and command pins in the same bank or side of the device as the CK/CK# pins, DQ, DQS, or DM pins. The reset# signal is only available in DDR3 SDRAM interfaces. Altera devices use the SSTL-15 I/O standard on the RESET# signal to meet the voltage requirements of 1.5 V CMOS at the memory device. Altera recommends that you do not terminate the RESET# signal to VTT.			

Notes to Table 3–10:

(1) The first CK/CK# pair refers to `mem_clk[0]` or `mem_clk_n[0]` in the IP core.

(2) The restriction on the placement for the first CK/CK# pair is required because this placement allows the mimic path that the IP VT tracking uses to go through differential I/O buffers to mimic the differential DQS signals.

Additional Placement Rules for Cyclone III and Cyclone IV Devices

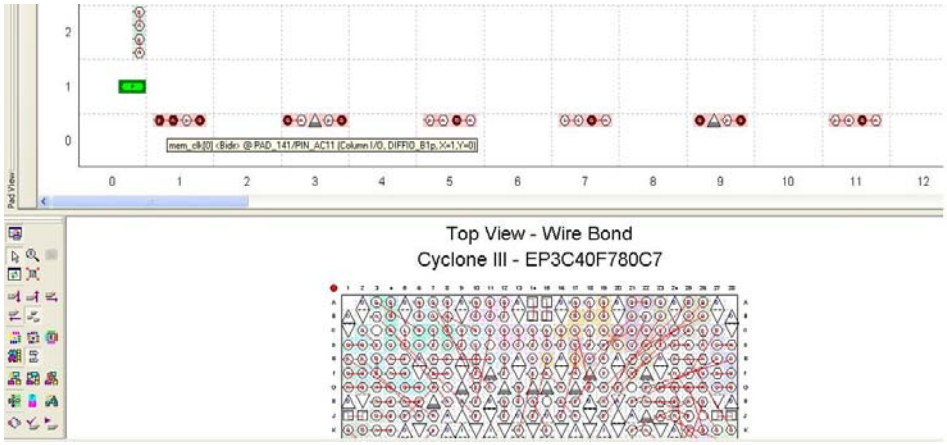
Assigning the `mem_clk[0]` pin on the same row or column pad group as the DQ pin pins results in the failure to constrain the DDIO input nodes correctly and close timing. Hence, the Read Capture and Write timing margins computed by TimeQuest may not be valid due to the violation of assumptions made by the timing scripts.

Figure 3–10 shows an example of assigning `mem_clk[0]` and `mem_clk_n[0]` incorrectly.

As you can see, `mem_clk[0]` pin is assigned at the same column pad group as `mem_dq` pin (in column X = 1). This assignment results in the Quartus II software showing the following critical warning:

```
Register <name> fed by pin mem_clk[0] must be placed in adjacent LAB X:1
Y:0 instead of X:2 Y:0
```

Figure 3–10. Incorrect Placement of mem_clk[0] and mem_clk_n[0] in Cyclone III and Cyclone IV Devices.



To eliminate this critical warning, assign the `mem_clk[0]` pin at different column or row from the data pin (Figure 3–11).

Figure 3–11. Correct Placement of mem_clk[0] and mem_clk_n[0] in Cyclone III and Cyclone IV Devices.

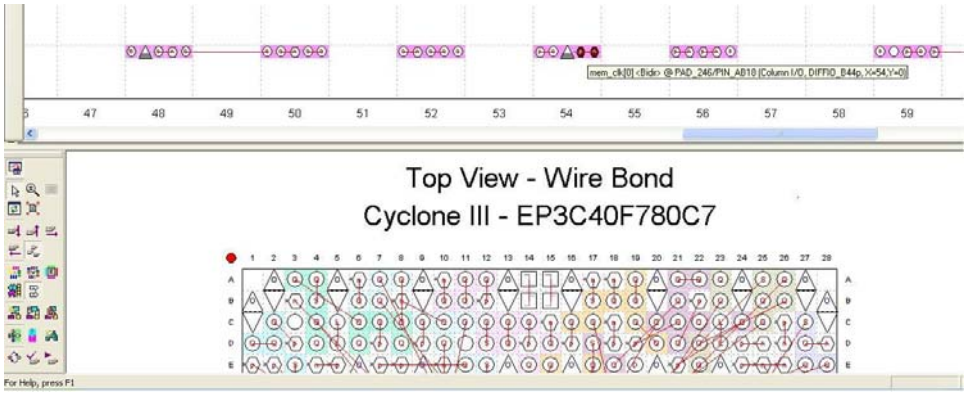


Table 3–11 lists the FPGA pin utilization for DDR3 SDRAM with leveling interfaces.

Table 3–11. DDR3 SDRAM With Leveling Interface Pin Utilization Applicable for Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices (Part 1 of 2)

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Data	DQ	DQ in the pin table, marked as Q in the Quartus II Pin Planner. Each DQ group has a common background color for all of the DQ and DM pins, associated with DQS (and DQSn) pins. The ×4 DIMM has the following mapping between DQS and DQ pins: ■ DQS[0] maps to DQ[3:0] ■ DQS[9] maps to DQ[7:4] ■ DQS[1] maps to DQ[11:8] ■ DQS[10] maps to DQ[15:12] The DQS pin index in other DIMM configurations typically increases sequentially with the DQ pin index (DQS[0]: DQ[3:0]; DQS[1]: DQ[7:4]; DQS[2]: DQ[11:8]). In this DIMM configuration, the DQS pins are indicted this way to ensure pin out is compatible with both ×4 and ×8 DIMMs.
Data Mask	DM	
Data Strobe	DQS and DQSn	DQS and DQSn (S and Sbar in the Quartus II Pin Planner)
Address and Command	A[], BA[], CAS#, CKE, CS#, ODT, RAS#, WE#,	Any user I/O pin. To minimize skew, you should place address and command pins in the same bank or side of the device as the following pins: CK/CK# pins, DQ, DQS, or DM pins.
	RESET#	ALTMEMPHY uses the SSTL-15 I/O standard and UniPHY uses the 1.5 V CMOS I/O standard on the RESET# signal. Both standards are valid. However, Altera recommends that you use the 1.5V CMOS I/O standard. If your board is already using the SSTL-15 I/O standard, you do not terminate the RESET# signal to V_{TT}.
Memory system clock	CK and CK#	For controllers with ALTMEMPHY IP, the first CK/CK# pin pairs (namely mem_clk[0] or mem_clk_n[0] in the IP) must use any unused DQ or DQS pins with DIFFIO_RX capability pins in the same bank or on the same side as the data pins. You can use either side of the device for wraparound interfaces. This placement is to allow the mimic path used in the IP VT tracking to go through differential I/O buffers to mimic the differential DQS signals. Any other CK/CK# pin pairs (mem_clk[n:1] and mem_clk_n [n:1]) can use any unused DQ or DQS pins in the same bank or on the same side as the data pins. For controllers with UniPHY IP, you can assign the memory clock to any unused DIFF_OUT pins in the same bank or on the same side as the data pins. However, for Arria V GZ and Stratix V devices, place the memory clock pins to any unused DQ or DQS pins. Do not place the memory clock pins in the same DQ group as any other DQ or DQS pins. If there are multiple CK/CK# pin pairs using Arria V GZ or Stratix V devices, you must place them on DIFFOUT in the same single DQ groups of adequate width. For example, DIMMs requiring three memory clock pin-pairs must use a ×4 DQS group. Placing the multiple CK/CK# pin pairs on DIFFOUT in the same single DQ groups for Stratix III and Stratix IV devices improves timing.

Table 3–11. DDR3 SDRAM With Leveling Interface Pin Utilization Applicable for Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices (Part 2 of 2)

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Clock Source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal.

Table 3–12 lists the FPGA pin utilization for QDR II and QDR II+ SRAM interfaces.

Table 3–12. QDR II and QDR II+ SRAM Pin Utilization for Arria II, Arria V, Stratix III, Stratix IV, and Stratix V Devices

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Read Clock	CQ and CQ# *(1)*	For QDR II SRAM devices with 1.5 or 2.5 cycles of read latency or QDR II+ SRAM devices with 2.5 cycles of read latency, connect CQ to DQS pin (S in the Quartus II Pin Planner), and CQn to CQn pin (Qbar in the Quartus II Pin Planner). For QDR II or QDR II+ SRAM devices with 2.0 cycles of read latency, connect CQ to CQn pin (Qbar in the Quartus II Pin Planner), and CQn to DQS pin (S in the Quartus II Pin Planner).
Read Data	Q	DQ pins (Q in the Quartus II Pin Planner). Ensure that you are using the DQ pins associated with the chosen read clock pins (DQS and CQn pins). QVLD pins are only available for QDR II+ SRAM devices and note that Altera IP does not use the QVLD pin.
Data Valid	QVLD	
Memory and Write Data Clock	K and K#	Differential or pseudo-differential DQ, DQS, or DQSn pins in or near the write data group.
Write Data	D	DQ pins. Ensure that you are using the DQ pins associated with the chosen memory and write data clock pins (DQS and DQS pins).
Byte Write Select	BWS#, NWS#	
Address and Control Signals	A, WPS#, RPS#	Any user I/O pin. To minimize skew, you should place address and command pins in the same bank or side of the device as the following pins: K and K# pins, DQ, DQS, BWS#, and NWS# pins. If you are using burst-length-of-two devices, place the address signals in a DQS group pin as these signals are now double data rate.
Clock source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal

Note to Table 3–12:

(1) For designs with integer latency, connect the CQ# signal to the CQ/CQ# pins from the pin table and ignore the polarity in the Pin Planner. For designs with fractional latency, connect the CQ signal to the CQ/CQ# pins from the pin table.

Table 3–13 lists the FPGA pin utilization for RLDRAM II CIO interfaces.

Table 3–13. RLDRAM II CIO Pin Utilization for Arria II GZ, Arria V, Stratix III, Stratix IV, and Stratix V Devices

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Read Clock	QK and QK# (1)	DQS and DQSn pins (S and Sbar in the Quartus II Pin Planner)
Data	Q	DQ pins (Q in the Quartus II Pin Planner). Ensure that you are using the DQ pins associated with the chosen read clock pins (DQS and DQSn pins). Altera IP does not use the QVLD pin. You may leave this pin unconnected on your board. You may not be able to fit these pins in a DQS group. For more information about how to place these pins, refer to "Exceptions for RLDRAM II and RLDRAM 3 Interfaces" on page 3–34.
Data Valid	QVLD	
Data Mask	DM	
Write Data Clock	DK and DK#	DQ pins in the same DQS group as the read data (Q) pins or in adjacent DQS group or in the same bank as the address and command pins. For more information, refer to "Exceptions for RLDRAM II and RLDRAM 3 Interfaces" on page 3–34. DK/DK# must use differential output-capable pins. For Nios-based configuration, the DK pins must be in a DQ group but the DK pins do not have to be in the same group as the data or QK pins.
Memory Clock	CK and CK#	Any differential output-capable pins. For Arria V GZ and Stratix V devices, place any unused DQ or DQS pins with DIFFOUT capability. Place the memory clock pins either in the same bank as the DK or DK# pins to improve DK versus CK timing, or in the same bank as the address and command pins to improve address command timing. Do not place CK and CK# pins in the same DQ group as any other DQ or DQS pins.
Address and Control Signals	A, BA, CS#, REF#, WE#	Any user I/O pins. To minimize skew, you should place address and command pins in the same bank or side of the device as the following pins: CK/CK# pins, DQ, DQS, and DM pins.
Clock source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal

Note to Table 3–13:

(1) For Arria V devices, refer to the pin table for the QK and QK# pins. Connect QK and QK# signals to the QK and QK# pins from the pin table and ignore the polarity in the Pin Planner.

Table 3–14 lists the FPGA pin utilization for RLDRAM II SIO interfaces.

Table 3–14. RLDRAM II SIO Pin Utilization Applicable for Arria II GZ, Arria V, Stratix III, Stratix IV, and Stratix V Devices

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Read Clock	QK and QK#	DQS and DQSn pins (S and Sbar in the Quartus II Pin Planner) in the same DQS group as the respective read data (Q) pins.
Read Data	Q	DQ pins (Q in the Quartus II Pin Planner). Ensure that you are using the DQ pins associated with the chosen read clock (DQS and DQSn) pins. Altera does not use the QVLD pin. You may leave this pin unconnected on your board.
Data valid	QVLD	
Memory and Write Data Clock	DK and DK#	DQS and DQSn pins (S and Sbar in the Quartus II Pin Planner) in the same DQS group as the respective write data (D) pins. For Nios-based configuration, the DK pins must be in a DQ group but the DK pins do not have to be in the same group as the data or QK pins.
Write Data	D	DQ pins. Ensure that you are using the DQ pins associated with the chosen write data clock (DQS and DQSn) pins.
Data Mask	DM	
Memory Clock	CK and CK#	Any differential output-capable pins. For Arria V GZ and Stratix V devices, place any unused DQ or DQS pins with DIFFOUT capability. Place the memory clock pins either in the same bank as the DK or DK# pins to improve DK versus CK timing, or in the same bank as the address and command pins to improve address command timing. Do not place CK and CK# pins in the same DQ group as any other DQ or DQS pins.
Address and Control Signals	A, BA, CS#, REF#, WE#	Any user I/O pin. To minimize skew, you should place address and command pins in the same bank or side of the device as the following pins: CK/CK# pins, DQ, DQS, or DM pins.
Clock source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal

Note to Table 3–14:

(1) For Arria V devices, refer to the pin table for the QK and QK# pins. Connect QK and QK# signals to the QK and QK# pins from the pin table and ignore the polarity in the Pin Planner.

Table 3–15 on page 3–46 lists the FPGA pin utilization for LPDDR2 SDRAM.

Table 3–15. LPDDR2 Pin Utilization for Arria V, Cyclone V, and Stratix V Devices

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Memory Clock	CK, CKn	Differential clock inputs. All double data rate (DDR) inputs are sampled on both positive and negative edges of the CK signal. Single data rate (SDR) inputs are sampled at the positive clock edge. Place any unused DQ or DQS pins with DIFFOUT capability for the mem_clk[n:0] and mem_clk_n[n:0] signals (where n>=0). Do not place CK and CK# pins in the same group as any other DQ or DQS pins. If there are multiple CK and CK# pin pairs, place them on DIFFOUT in the same single DQ group of adequate width.
Address and Command	CA0-CA9 CSn CKE	Unidirectional DDR command and address bus inputs. Chip Select: CSn is considered to be part of the command code. Clock Enable: CKE HIGH activates and CKE LOW deactivates internal clock signals and therefore device input buffers and output drivers. Any user I/O pin. To minimize skew, Altera recommends using address and command pins in the same bank or side of the device as the CK/CK#, DQ. DQS, or DM pins.
Data	DQ0-DQ7 (×8) DQ0-DQ15 (×16) DQ0-DQ31 (×32)	Bidirectional data bus. Pins are used as data inputs and outputs. DQ in the pin table is marked as Q in the Pin Planner. Each DQ group has a common background color for all of the DQ and DM pins associated with DQS (and DQSn) pins. Place on DQ group pin marked Q in the Pin Planner.
Data Strobe	DQS, DQSn	Data Strobe. The data strobe is bidirectional (used for read and write data) and differential (DQS and DQSn). It is output with read data and input with write data. Place on DQS and DQSn (S and Sbar in the Pin Planner) for differential DQS signaling.
Data Mask	DM0 (×8) DM0-DM1 (×16) DM0-DM3 (×32)	Input Data Mask. DM is the input mask signal for write data. Input data is masked when DM is sampled HIGH coincident with that input data during a write access. DM is sampled on both edges of DQS. DQ in the pin table is marked as Q in the Pin Planner. Each DQ group has a common background color for all of the DQ and DM pins, associated with DQS (and DQSn) pins. Place on DQ group pin marked Q in the Pin Planner.
Clock Source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal.

Additional Guidelines for Arria V GZ and Stratix V Devices

This section provides guidelines on how to improve timing for Arria V GZ and Stratix V devices and the rules that you must follow to overcome timing failures.

Performing Manual Pin Placement

Table 3–16 lists rules that you can follow to perform proper manual pin placement and avoid timing failures.

The rules are categorized as follows:

- **Mandatory**—This rule is mandatory and cannot be violated as it would result in a no-fit error.

- **Recommended**—This rule is recommended and if violated the implementation is legal but the timing is degraded.
- **Highly Recommended**—This rule is not mandatory but is highly recommended because disregarding this rule might result in timing violations.

Table 3–16. Manual Pin Placement Rules (Part 1 of 2)

Rules	Frequency	Device	Reason
Mandatory			
Must place all CK, CK#, address, control, and command pins of an interface in the same I/O sub-bank.	> 800 MHz	All	For optimum timing, clock and data output paths must share as much hardware as possible. For write data pins (for example, DQ/DQS), the best timing is achieved through the DQS Groups.
Must not split interface between top and bottom sides	Any	All	Because PLLs and DLLs on the top edge cannot access the bottom edge of a device and vice-versa.
Must not place pins from separate interfaces in the same I/O sub-banks unless the interfaces share PLL or DLL resources.	Any	All	All pins require access to the same leveling block.
Must not share the same PLL input reference clock unless the interfaces share PLL or DLL resources.	Any	All	Because sharing the same PLL input reference clock forces the same ff-PLL to be used. Each ff-PLL can drive only one PHY clock tree and interfaces not sharing a PLL cannot share a PHY clock tree.
Recommended			
Place all CK, CK#, address, control, and command pins of an interface in the same I/O sub-bank.	<800 MHz	All	Place all CK/CK#, address, control, and command pins in the same I/O sub-bank when address and command timing is critical. For optimum timing, clock and data output paths should share as much hardware as possible. For write data pins (for example, DQ/DQS), the best timing is achieved through the DQS Groups.
Avoid using I/Os at the device corners (for example, sub-bank "A").	Any	A7 *(1)*	The delay from the FPGA core fabric to the I/O periphery is higher toward the sub-banks in the corners. By not using I/Os at the device corners, you can improve core timing closure.
	>=800 MHz	All	Corner I/O pins use longer delays, therefore avoiding corner I/O pins is recommended for better memory clock performance.

Table 3–16. Manual Pin Placement Rules (Part 2 of 2)

Rules	Frequency	Device	Reason
Avoid straddling an interface across the center PLL.	Any	All	Straddling the center PLL causes timing degradation, because it increases the length of the PHY clock tree and increases jitter. By not straddling the center PLL, you can improve core timing closure.
Use the center PLL(f-PLL1) for a wide interface that must straddle across center PLL.	>= 800 MHz	All	Using a non-center PLL results in driving a sub-bank in the opposite quadrant due to long PHY clock tree delay.
Place the DQS/DQS# pins such that all DQ groups of the same interface are next to each other and do not span across the center PLL.	Any	All	To ease core timing closure. If the pins are too far apart then the core logic is also placed apart which results in difficult timing closure.
Place CK, CK#, address, control, and command pins in the same quadrant as DQ groups for improved timing in general.	Any	All	
Highly Recommended			
Place all CK, CK#, address, control, and command pins of an interface in the same I/O sub-bank.	800 MHz	All	For optimum timing, clock and data output paths should share as much hardware as possible. For write data pins (for example, DQ/DQS), the best timing is achieved through the DQS Groups.
Use center PLL and ensure that the PLL input reference clock pin is placed at a location that can drive the center PLL.	>= 800 MHz	All	Using a non-center PLL results in driving a sub-bank in the opposite quadrant due to long PHY clock tree delay.
If center PLL is not accessible, place pins in the same quadrant as the PLL.	>= 800 MHz	All	

Note to Table 3–16:

(1) This rule is currently applicable to A7 devices only. This rule might be applied to other devices in the future if they show the same failure.

Additional Guidelines for Arria V (Except Arria V GZ) Devices

This section provides guidelines on how to improve timing for Arria V devices and the rules that you must follow to overcome timing failures.

Performing Manual Pin Placement

Table 3–17 lists a set of rules you can follow to perform proper manual pin placement and avoid timing failures.

The rules are categorized as follows:

- **Mandatory**—This rule is mandatory and cannot be violated as it would result in a no-fit error.

- **Recommended**—This rule is recommended and if violated the implementation is legal but the timing is degraded.

Table 3–17. Manual Pin Placement Rules for Arria V (Except Arria V GZ) Devices

Rules	Frequency	Device	Reason
Mandatory			
Must place all CK, CK#, address, control, and command pins of an interface on the same device edge as the DQ groups.	All	All	For optimum timing, clock and data output ports must share as much hardware as possible.
Must not place pins from separate interfaces in the same I/O sub-banks unless the interfaces share PLL or DLL resources. To share resources, the interfaces must use the same memory protocol, frequency, controller rate, and phase requirements.	All	All	All pins require access to the same PLL/DLL block.
Must not split interface between top, bottom, and right sides.	All	All	PHYCLK network support interfaces at the same side of the I/O banks only. PHYCLK networks do not support split interface.
Recommended			
Place the DQS/DQS# pins such that all DQ groups of the same interface are next to each other and do not span across the center PLL.	All	All	To ease core timing closure. If the pins are too far apart then the core logic is also placed apart which results in difficult timing closure.
Place all pins for a memory interface in an I/O bank and use the nearest PLL to that I/O bank for the memory interface.	All	All	Improve timing performance by reducing the PHY clock tree delay.

Additional Guidelines for Cyclone V Devices

This section provides guidelines on how to improve performance for Cyclone V devices.

I/O Pins Connect to Ground for Hard Memory Interface Operation

According to the Cyclone V pin-out file, there are some general I/O pins that are connected to ground for hard memory interface operation. These I/O pins should be grounded to reduce crosstalk from neighboring I/O pins and to ensure the performance of the hard memory interface.

The grounded user I/O pins can also be used as regular I/O pins if you run short of available I/O pins; however, the hard memory interface performance will be reduced if these pins are not connected to ground.

PLLs and Clock Networks

The exact number of clocks and PLLs required in your design depends greatly on the memory interface frequency, and the IP that your design uses.

For example, you can build simple DDR slow-speed interfaces that typically require only two clocks: system and write. You can then use the rising and falling edges of these two clocks to derive four phases (0°, 90°, 180°, and 270°). However, as clock speeds increase, the timing margin decreases and additional clocks are required, to optimize setup and hold and meet timing. Typically, at higher clock speeds, you need to have dedicated clocks for resynchronization, and address and command paths.

In addition, ALTMEMPHY-based interfaces, use a voltage and temperature (VT) tracking clock to measure and compensate for VT changes and their effects.

Altera memory interface IP uses one PLL, which generates the various clocks needed in the memory interface data path and controller, and provides the required phase shifts for the write clock and address and command clock. The PLL is instantiated when you generate the Altera memory IPs.

By default, the memory interface IP uses the PLL to generate the input reference clock for the DLL, available in all device families except for the Cyclone III and Cyclone IV devices. This method eliminates the need of an extra pin for the DLL input reference clock.

The input reference clock to the DLL can come from certain input clock pins or clock output from certain PLLs.

Altera recommends using integer PLLs for memory interfaces; handbook specifications are based on integer PLL implementations.

For the actual pins and PLLs connected to the DLLs, refer to the *External Memory Interfaces* chapter of the relevant device family handbook.

You must use the PLL located in the same device quadrant or side as the memory interface and the corresponding dedicated clock input pin for that PLL, to ensure optimal performance and accurate timing results from the Quartus II software.

The input clock to the PLL can fan out to logic other than the PHY, so long as the clock input pin to the PLL is a dedicated input clock path, and you ensure that the clock domain transfer between UniPHY and the core logic is clocked by the reference clock going into a global clock.

Table 3–18 and Table 3–19 list a comparison of the number of PLLs and dedicated clock outputs available respectively in Arria II, Arria V, Cyclone III, Cyclone IV, Stratix III, Stratix IV, and Stratix V devices.

Table 3–18. Number of PLLs Available in Altera Device Families *(1)*

Device Family	Enhanced PLLs Available
Arria II GX	4-6
Arria II GZ	3-8
Arria V	16-24
Arria V GZ (fPLL)	22-28

Table 3–18. Number of PLLs Available in Altera Device Families (1)

Device Family	Enhanced PLLs Available
Cyclone III and Cyclone IV	2-4
Cyclone V	4-8
Stratix III	4-12
Stratix IV	3-12
Stratix V (fPLL)	22-28

Note to Table 3–18:

(1) For more details, refer to the *Clock Networks and PLL* chapter of the respective device family handbook.

Table 3–19. Number of Enhanced PLL Clock Outputs and Dedicated Clock Outputs Available in Altera Device Families (1)

Device Family	Number of Enhanced PLL Clock Outputs	Number Dedicated Clock Outputs
Arria II GX (2)	7 clock outputs each	1 single-ended or 1 differential pair 3 single-ended or 3 differential pair total (3)
Arria V	18 clock outputs each	4 single-ended or 2 single-ended and 1 differential pair
Cyclone III and Cyclone IV	5 clock outputs each	1 single-ended or 1 differential pair total (not for memory interface use)
Stratix III	Left/right: 7 clock outputs Top/bottom: 10 clock outputs	Left/right: 2 single-ended or 1 differential pair Top/bottom: 6 single-ended or 4 single-ended and 1 differential pair
Arria II GZ and Stratix IV	Left/right: 7 clock outputs Top/bottom: 10 clock outputs	Left/right: 2 single-ended or 1 differential pair Top/bottom: 6 single-ended or 4 single-ended and 1 differential pair
Arria V GZ and Stratix V	18 clock outputs each	4 single-ended or 2 single-ended and 1 differential pair

Notes to Table 3–19:

(1) For more details, refer to the *Clock Networks and PLL* chapter of the respective device family handbook.

(2) PLL_5 and PLL_6 of Arria II GX devices do not have dedicated clock outputs.

(3) The same PLL clock outputs drives three single-ended or three differential I/O pairs, which are only supported in PLL_1 and PLL_3 of the EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices.

Table 3–20 lists the number of clock networks available in the Altera device families.

Table 3–20. Number of Clock Networks Available in Altera Device Families (1) **(Part 1 of 2)**

Device Family	Global Clock Network	Regional Clock Network
Arria II GX	16	48
Arria II GZ	16	64–88
Arria V	16	88
Arria V GZ	16	92

Table 3–20. Number of Clock Networks Available in Altera Device Families (1) (Part 2 of 2)

Device Family	Global Clock Network	Regional Clock Network
Cyclone III and Cyclone IV	10-20	N/A
Cyclone V	16	N/A
Stratix III	16	64–88
Stratix IV	16	64–88
Stratix V	16	92

Note to Table 3–20:

(1) For more information on the number of available clock network resources per device quadrant to better understand the number of clock networks available for your interface, refer to the *Clock Networks and PLL* chapter of the respective device family handbook.

You must decide whether you need to share clock networks, PLL clock outputs, or PLLs if you are implementing multiple memory interfaces.

Table 3–21 through Table 3–23 list the number of PLL outputs and clock networks required for the memory standards using Altera IP. Table 3–24 lists the names and frequency of the clocks used.

Table 3–21. Clock Network Usage in ALTMEMPHY-based Memory Standards

Device	DDR3 SDRAM		DDR2/DDR SDRAM			
	Half-Rate		Half-Rate		Full-Rate	
	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock	Number of half-rate clock
Arria II GX	4 global	2 global	4 global	2 global	5 global	1 global
Cyclone III and Cyclone IV	—		4 global	1 global	5 global	—
Stratix III and Stratix IV	1 global 2 regional	2 regional	1 regional 2 dual-regional	1 global 2 dual-regional	1 global 2 dual-regional	2 dual-regional

Table 3–22. Clock Network Usage in UniPHY-based Memory Interfaces—DDR2 and DDR3 SDRAM (1) (1)

Device	DDR3 SDRAM		DDR2 SDRAM	
	Half-Rate		Half-Rate	
	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock	Number of half-rate clock
Stratix III	3 global	1 global 1 regional	1 global 2 global	1 global 1 regional
Arria II GZ and Stratix IV	3 global	1 global 1 regional	1 regional 2 regional	1 global 1 regional

Table 3–22. Clock Network Usage in UniPHY-based Memory Interfaces—DDR2 and DDR3 SDRAM *(1) (1)*

Device	DDR3 SDRAM		DDR2 SDRAM	
	Half-Rate		Half-Rate	
	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock	Number of half-rate clock
Arria V GZ and Stratix V	1 global 2 regional	2 global	1 regional 2 regional	2 global

Note to Table 3–22:

(1) There are two additional regional clocks, `pll_avl_clk` and `pll_config_clk` for DDR2 and DDR3 SDRAM with UniPHY memory interfaces.

(2) In multiple interface designs with other IP, the clock network might need to be modified to get a design to fit. For more information, refer to *Clock Networks and PLLs* chapter in the respective device handbooks.

Table 3–23. Clock Network Usage in UniPHY-based Memory Interfaces—RLDRAM II, and QDR II and QDR II+ SRAM

Device	RLDRAM II			QDR II/QDR II+ SRAM		
	Half-Rate		Full-Rate	Half-Rate		Full-Rate
	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock
Arria II GX	—	—	—	2 global	2 global	4 global
Stratix III	2 regional	1 global 1 regional	1 global 2 regional	1 global 1 regional	2 regional	1 global 2 regional
Arria II GZ and Stratix IV	2 regional	1 global 1 regional	1 global 2 regional	1 global 1 regional	2 regional	1 global 2 regional

For more information about the clocks used in UniPHY-based memory standards, refer to the *Functional Description—UniPHY* chapter in volume 3 of the *External Memory Interface Handbook*.

Table 3–24. Clocks Used in the ALTMEMPHY Megafunction *(1)*

Clock Name	Usage Description
`phy_clk_1x`	Static system clock for the half-rate data path and controller.
`mem_clk_2x`	Static DQS output clock that generates DQS, CK/CK# signals, the input reference clock to the DLL, and the system clock for the full-rate datapath and controller.
`mem_clk_1x`	This clock drives the `aux_clk` output or clocking DQS and as a reference clock for the memory devices.
`write_clk_2x`	Static DQ output clock used to generate DQ signals at 90º earlier than DQS signals. Also may generate the address and command signals.
`mem_clk_ext_2x`	This clock is only used if the memory clock generation uses dedicated output pins. Applicable only in HardCopy® II or Stratix II prototyping for HardCopy II designs.
`resync_clk_2x`	Dynamic-phase clock used for resynchronization and postamble paths. Currently, this clock cannot be shared by multiple interfaces.
`measure_clk_2x`/ `measure_clk_1x` *(2)*	Dynamic-phase clock used for VT tracking purposes. Currently, this clock cannot be shared by multiple interfaces.

Table 3–24. Clocks Used in the ALTMEMPHY Megafunction *(1)*

Clock Name	Usage Description
ac_clk_2x ac_clk_1x	Dedicated static clock for address and command signals.
scan_clk	Static clock to reconfigure the PLL
seq_clk	Static clock for the sequencer logic

Notes to Table 3–24:

(1) For more information about the clocks used in the ALTMEMPHY megafunction, refer to the *Clock Networks and PLL* chapter of the respective device family handbook for more details.

(2) This clock should be of the same clock network clock as the resync_clk_2x clock.

In every ALTMEMPHY solution, the measure_clk and resync_clk_2x clocks (Table 3–24) are calibrated and hence may not be shared or used for other modules in your system. You may be able to share the other statically phase-shifted clocks with other modules in your system provided that you do not change the clock network used.

Changing the clock network that the ALTMEMPHY solution uses may affect the output jitter, especially if the clock is used to generate the memory interface output pins. Always check the clock network output jitter specification in the *DC and Switching Characteristics* chapter of the device handbook, before changing the ALTMEMPHY clock network, to ensure that it meets the memory standard jitter specifications, which includes period jitter, cycle-to-cycle jitter and half duty cycle jitter.

If you need to change the resync_clk_2x clock network, you have to change the measure_clk_1x clock network also to ensure accurate VT tracking of the memory interface.

For more information about sharing clocks in multiple controllers, refer to the design tutorials on the List of designs using Altera External Memory IP page of the Altera Wiki website.

In addition, you should not change the PLL clock numbers as the wizard-generated Synopsis Design Constraints File (**.sdc**) assumes certain counter outputs from the PLL (Table 3–25 through Table 3–26).

Table 3–25. PLL Usage for DDR, DDR2, and DDR3 SDRAM Without Leveling Interfaces

Clock	Arria II GX Devices	Cyclone III and Cyclone IV Devices	Stratix III and Stratix IV Devices
C0	■ phy_clk_1x in half-rate designs ■ aux_half_rate_clk ■ PLL scan_clk	■ phy_clk_1x in half-rate designs ■ aux_half_rate_clk	■ phy_clk_1x in half-rate designs ■ aux_half_rate_clk ■ PLL scan_clk
C1	■ phy_clk_1x in full-rate designs ■ aux_full_rate_clk ■ mem_clk_2x to generate DQS and CK/CK# signals ■ ac_clk_2x ■ cs_n_clk_2x	■ phy_clk_1x in full-rate designs ■ aux_full_rate_clk ■ mem_clk_2x to generate DQS and CK/CK# signals ■ ac_clk_2x ■ cs_n_clk_2x	■ mem_clk_2x
C2	■ Unused	■ write_clk_2x (for DQ) ■ ac_clk_2x ■ cs_n_clk_2x	■ phy_clk_1x in full-rate designs ■ aux_full_rate_clk
C3	■ write_clk_2x (for DQ) ■ ac_clk_2x ■ cs_n_clk_2x	■ resync_clk_2x	■ write_clk_2x
C4	■ resync_clk_2x	■ measure_clk_2x	■ resync_clk_2x
C5	■ measure_clk_2x	—	■ measure_clk_1x
C6	—	—	■ ac_clk_1x

Table 3–26. PLL Usage for DDR3 SDRAM With Leveling Interfaces

Clock	Stratix III and Stratix IV Devices
C0	■ phy_clk_1x in half-rate designs ■ aux_half_rate_clk ■ PLL scan_clk
C1	■ mem_clk_2x
C2	■ aux_full_rate_clk
C3	■ write_clk_2x
C4	■ resync_clk_2x
C5	■ measure_clk_1x
C6	■ ac_clk_1x

Using PLL Guidelines

When using PLL for external memory interfaces, you must consider the following guidelines:

- For the clock source, use the clock input pin specifically dedicated to the PLL that you want to use with your external memory interface. The input and output pins are only fully compensated when you use the dedicated PLL clock input pin. If the clock source for the PLL is not a dedicated clock input pin for the dedicated PLL, you would need an additional clock network to connect the clock source to the PLL block. Using additional clock network may increase clock jitter and degrade the timing margin.
- Pick a PLL and PLL input clock pin that are located on the same side of the device as the memory interface pins.
- Share the DLL and PLL static clocks for multiple memory interfaces provided the controllers are on the same or adjacent side of the device and run at the same memory clock frequency.
- If you are using Cyclone III or Cyclone IV devices, you need not set the PLL mode to **No Compensation** in the Quartus II software. The PLL for these devices in **Normal** mode has low jitter. Changing the compensation mode may result in inaccurate timing results.
- If your design uses a dedicated PLL to only generate a DLL input reference clock (not available for Cyclone III or Cyclone IV device), you must set the PLL mode to **No Compensation** in the Quartus II software to minimize the jitter, or the software forces this setting automatically. The PLL does not generate other output, so it does not need to compensate for any clock path.
- If your design cascades PLL, the source (upstream) PLL must have a low-bandwidth setting, while the destination (downstream) PLL must have a high-bandwidth setting to minimize jitter. Altera does not recommend using cascaded PLLs for external memory interfaces because your design gets accumulated jitters. The memory output clock may violate the memory device jitter specification.

  Use cascading PLLs at your own risk. For more information, refer to "PLL Cascading" on page 3–57.

- If you are using Arria II GX devices, for a single memory instance that spans two right-side quadrants, use a middle-side PLL as the source for that interface.
- If you are using Arria II GZ, Arria V GZ, Stratix III, Stratix IV, or Stratix V devices, for a single memory instance that spans two top or bottom quadrants, use a middle top or bottom PLL as the source for that interface. The ten dual regional clocks that the single interface requires must not block the design using the adjacent PLL (if available) for a second interface.

PLL Cascading

Arria II GZ PLLs, Stratix III PLLs, Stratix IV PLLs, Stratix V and Arria V GZ fractional PLLs (fPLLs), and the two middle PLLs in Arria II GX EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices can be cascaded using either the global or regional clock trees, or the cascade path between two adjacent PLLs.

Use cascading PLLs at your own risk. You should use faster memory devices to maximize timing margins.

Cyclone III and Cyclone IV devices do not support PLL cascading for external memory interfaces.

The UniPHY IP supports PLL cascading using the cascade path without any additional timing derating when the bandwidth and compensation rules are followed. The timing constraints and analysis assume that there is no additional jitter due to PLL cascading when the upstream PLL uses no compensation and low bandwidth, and the downstream PLL uses no compensation and high bandwidth.

The UniPHY IP does not support PLL cascading using the global and regional clock networks. You can implement PLL cascading at your own risk without any additional guidance and specifications from Altera. The Quartus II software does issue a critical warning suggesting use of the cascade path to minimize jitter, but does not explicitly state that Altera does not support cascading using global and regional clock networks.

The Quartus II software does not issue a critical warning stating that Cyclone III and Cyclone IV ALTMEMPHY designs do not support PLL cascading; it issues the Stratix III warning message requiring use of cascade path.

Some Arria II GX devices (EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260) have direct cascade path for two middle right PLLs. Arria II GX PLLs have the same bandwidth options as Stratix IV GX left and right PLLs.

DLL

The Altera memory interface IP uses one DLL (except in Cyclone III and Cyclone IV devices, where this resource is not available). The DLL is located at the corner of the device and can send the control signals to shift the DQS pins on its adjacent sides for Stratix-series devices, or DQS pins in any I/O banks in Arria II GX devices.

For example, the top-left DLL can shift DQS pins on the top side and left side of the device. The DLL generates the same phase shift resolution for both sides, but can generate different phase offset to the two different sides, if needed. Each DQS pin can be configured to use or ignore the phase offset generated by the DLL.

The DLL cannot generate two different phase offsets to the same side of the device. However, you can use two different DLLs to for this functionality.

DLL reference clocks must come from either dedicated clock input pins located on either side of the DLL or from specific PLL output clocks. Any clock running at the memory frequency is valid for the DLLs.

To minimize the number of clocks routed directly on the PCB, typically this reference clock is sourced from the memory controllers PLL. In general, DLLs can use the PLLs directly adjacent to them (corner PLLs when available) or the closest PLL located in the two sides adjacent to its location.

By default, the DLL reference clock in Altera external memory IP is from a PLL output.

When designing for 780-pin packages with EP3SE80, EP3SE110, EP3SL150, EP4SE230, EP4SE360, EP4SGX180, and EP4SGX230 devices, the PLL to DLL reference clock connection is limited. DLL2 is isolated from a direct PLL connection and can only receive a reference clock externally from pins `CLK[11:4]p` in EP3SE80, EP3SE110, EP3SL150, EP4SE230, and EP4SE360 devices. In EP4SGX180 and EP4SGX230 devices, DLL2 and DLL3 are not directly connected to PLL. DLL2 and DLL3 receive a reference clock externally from pins `CLK[7:4]p` and `CLK[15:12]p` respectively.

For more DLL information, refer to the respective device handbooks.

The DLL reference clock should be the same frequency as the memory interface, but the phase is not important.

The required DQS capture phase is optimally chosen based on operating frequency and external memory interface type (DDR, DDR2, DDR3 SDRAM, and QDR II SRAM, or RLDRAM II). As each DLL supports two possible phase offsets, two different memory interface types operating at the same frequency can easily share a single DLL. More may be possible, depending on the phase shift required.

Altera memory IP always specifies a default optimal phase setting, to override this setting, refer to the *Implementing and Parameterizing Memory IP* chapter.

When sharing DLLs, your memory interfaces must be of the same frequency. If the required phase shift is different amongst the multiple memory interfaces, you can use a different delay chain in the DQS logic block or use the DLL phase offset feature.

To simplify the interface to IP connections, multiple memory interfaces operating at the same frequency usually share the same system and static clocks as each other where possible. This sharing minimizes the number of dedicated clock nets required and reduces the number of different clock domains found within the same design.

As each DLL can directly drive four banks, but each PLL only has complete C (output) counter coverage of two banks (using dual regional networks), situations can occur where a second PLL operating at the same frequency is required. As cascaded PLLs increase jitter and reduce timing margin, you are advised to first ascertain if an alternative second DLL and PLL combination is not available and more optimal.

Select a DLL that is available for the side of the device where the memory interface resides. If you select a PLL or a PLL input clock reference pin that can also serve as the DLL input reference clock, you do not need an extra input pin for the DLL input reference clock.

Other FPGA Resources

The Altera memory interface IP uses FPGA fabric, including registers and the Memory Block to implement the memory interface.

For resource utilization examples to ensure that you can fit your other modules in the device, refer to the "Resource Utilization" section in the *Introduction to UniPHY IP* and the *Introduction to ALTMEMPHY IP* chapters of the *External Memory Interface Handbook*.

In addition, one OCT calibration block is used if you are using the FPGA OCT feature in the memory interface.The OCT calibration block uses two pins (R_{UP} and R_{DN}), or single pin (R_{ZQ}) ("OCT Support for Arria II GX, Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices" on page 3–25). You can select any of the available OCT calibration block as you do not need to place this block in the same bank or device side of your memory interface. The only requirement is that the I/O bank where you place the OCT calibration block uses the same V_{CCIO} voltage as the memory interface. You can share multiple memory interfaces with the same OCT calibration block if the V_{CCIO} voltage is the same.

Even though Cyclone III and Cyclone IV devices support OCT, this feature is not turned on by default in the Altera IP solution.

Document Revision History

Table 3–27 lists the revision history for this document.

Table 3–27. Document Revision History

Date	Version	Changes
November 2012	6.0	■ Added Arria V GZ information. ■ Added RLDRAM 3 information. ■ Added LRDIMM information.
June 2012	5.0	■ Added LPDDR2 information. ■ Added Cyclone V information. ■ Added Feedback icon.
November 2011	4.0	■ Moved and reorganized "Planning Pin and Resource" section to Volume 2:Design Guidelines. ■ Added Additional Guidelines for Arria V GZ and Stratix V Devices section. ■ Added Arria V and Cyclone V information.
June 2011	3.0	■ Moved *Select a Device* and *Memory IP Planning* chapters to Volume 1. ■ Added information about interface pins. ■ Added guidelines for using PLL.
December 2010	2.1	■ Added a new section on controller efficiency. ■ Added Arria II GX and Stratix V information.
July 2010	2.0	Updated information about UniPHY-based interfaces and Stratix V devices.
April 2010	1.0	Initial release.

4. DDR2 and DDR3 SDRAM Board Design Guidelines

EMI_DG_004-5.0

This chapter provides guidelines on how to improve the signal integrity of your system and layout guidelines to help you successfully implement a DDR2 or DDR3 SDRAM interface on your system.

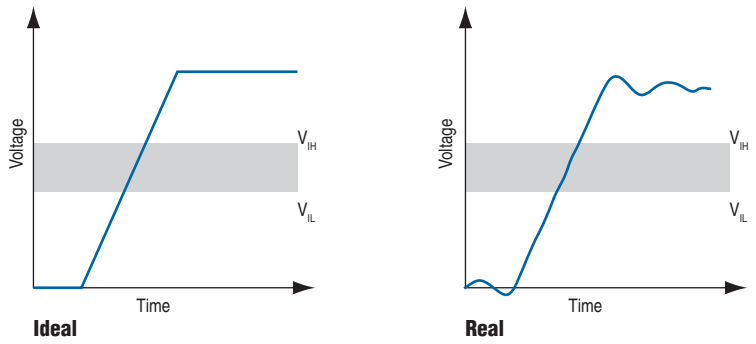
DDR3 SDRAM is the third generation of the DDR SDRAM family, and offers improved power, higher data bandwidth, and enhanced signal quality with multiple on-die termination (ODT) selection and output driver impedance control while maintaining partial backward compatibility with the existing DDR2 SDRAM standard.

This chapter focuses on the following key factors that affect signal quality at the receiver:

- Leveling and dynamic ODT
- Proper use of termination
- Output driver drive strength setting
- Loading at the receiver
- Layout guidelines

As memory interface performance increases, board designers must pay closer attention to the quality of the signal seen at the receiver because poorly transmitted signals can dramatically reduce the overall data-valid margin at the receiver. Figure 4–1 shows the differences between an ideal and real signal seen by the receiver.

Figure 4–1. Ideal and Real Signal at the Receiver



© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

In addition, this chapter compares various types of termination schemes, and their effects on the signal quality on the receiver. It also discusses the proper drive strength setting on the FPGA to optimize the signal integrity at the receiver, and the effects of different loading types, such as components versus DIMM configuration, on signal quality. The objective of this chapter is to understand the trade-offs between different types of termination schemes, the effects of output drive strengths, and different loading types, so you can swiftly navigate through the multiple combinations and choose the best possible settings for your designs.

Leveling and Dynamic ODT

DDR3 SDRAM DIMMs, as specified by JEDEC, always use a fly-by topology for the address, command, and clock signals. This standard DDR3 SDRAM topology requires the use of Altera® DDR3 SDRAM Controller with UniPHY or ALTMEMPHY with read and write leveling.

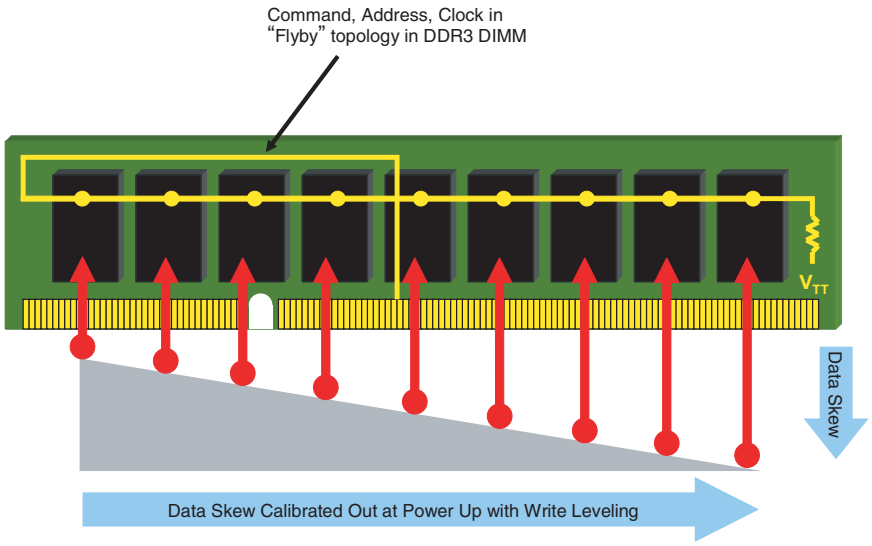
Altera recommends that for full DDR3 SDRAM compatibility when using discrete DDR3 SDRAM components, you should mimic the JEDEC DDR3 UDIMM fly-by topology on your custom printed circuit boards (PCB).

Arria® II, Arria V, and Cyclone® V devices do not support DDR3 SDRAM with read or write leveling, so these devices do not support standard DDR3 SDRAM DIMMs or DDR3 SDRAM components using the standard DDR3 SDRAM fly-by address, command, and clock layout topology.

Read and Write Leveling

One major difference between DDR2 and DDR3 SDRAM is the use of leveling. To improve signal integrity and support higher frequency operations, the JEDEC committee defined a fly-by termination scheme used with clocks, and command and address bus signals. Fly-by topology reduces simultaneous switching noise (SSN) by deliberately causing flight-time skew between the data and strobes at every DRAM as the clock, address, and command signals traverse the DIMM (Figure 4–2).

Figure 4–2. DDR3 DIMM Fly-By Topology Requiring Write Leveling



The flight-time skew caused by the fly-by topology led the JEDEC committee to introduce the write leveling feature on the DDR3 SDRAMs; thus requiring controllers to compensate for this skew by adjusting the timing per byte lane.

During a write, DQS groups launch at separate times to coincide with a clock arriving at components on the DIMM, and must meet the timing parameter between the memory clock and DQS defined as t_{DQSS} of ± 0.25 t_{CK}.

During the read operation, the memory controller must compensate for the delays introduced by the fly-by topology. The Stratix® III, Stratix IV, and Stratix V FPGAs have alignment and synchronization registers built in the I/O element (IOE) to properly capture the data.

In DDR2 SDRAM, there are only two drive strength settings, full or reduced, which correspond to the output impedance of 18 Ω and 40 Ω, respectively. These output drive strength settings are static settings and are not calibrated; as a result, the output impedance varies as the voltage and temperature drifts.

The DDR3 SDRAM uses a programmable impedance output buffer. Currently, there are two drive strength settings, 34 Ω and 40 Ω. The 40-Ω drive strength setting is currently a reserved specification defined by JEDEC, but available on the DDR3 SDRAM, as offered by some memory vendors. Refer to the datasheet of the respective memory vendors for more information about the output impedance setting. You select the drive strength settings by programming the memory mode register defined by mode register 1 (MR1). To calibrate output driver impedance, an external precision resistor, RZQ, connects the ZQ pin and VSSQ. The value of this resistor must be 240 Ω ± 1%.

If you are using a DDR3 SDRAM DIMM, RZQ is soldered on the DIMM so you do not need to layout your board to account for it. Output impedance is set during initialization. To calibrate output driver impedance after power-up, the DDR3 SDRAM needs a calibration command that is part of the initialization and reset procedure and is updated periodically when the controller issues a calibration command.

In addition to calibrated output impedance, the DDR3 SDRAM also supports calibrated parallel ODT through the same external precision resistor, RZQ, which is possible by using a merged output driver structure in the DDR3 SDRAM, which also helps to improve pin capacitance in the DQ and DQS pins. The ODT values supported in DDR3 SDRAM are 20 Ω, 30 Ω, 40 Ω, 60 Ω, and 120 Ω, assuming that RZQ is 240 Ω.

In DDR3 SDRAM, there are two commands related to the calibration of the output driver impedance and ODT. The controller often uses the first calibration command, ZQ CALIBRATION LONG (ZQCL), at initial power-up or when the DDR3 SDRAM is in a reset condition. This command calibrates the output driver impedance and ODT to the initial temperature and voltage condition, and compensates for any process variation due to manufacturing. If the controller issues the ZQCL command at initialization or reset, it takes 512 memory clock cycles to complete; otherwise, it requires 256 memory clock cycles to complete. The controller uses the second calibration command, ZQ CALIBRATION SHORT (ZQCS) during regular operation to track any variation in temperature or voltage. The ZQCS command takes 64 memory clock cycles to complete. Use the ZQCL command any time there is more impedance error than can be corrected with a ZQCS command.

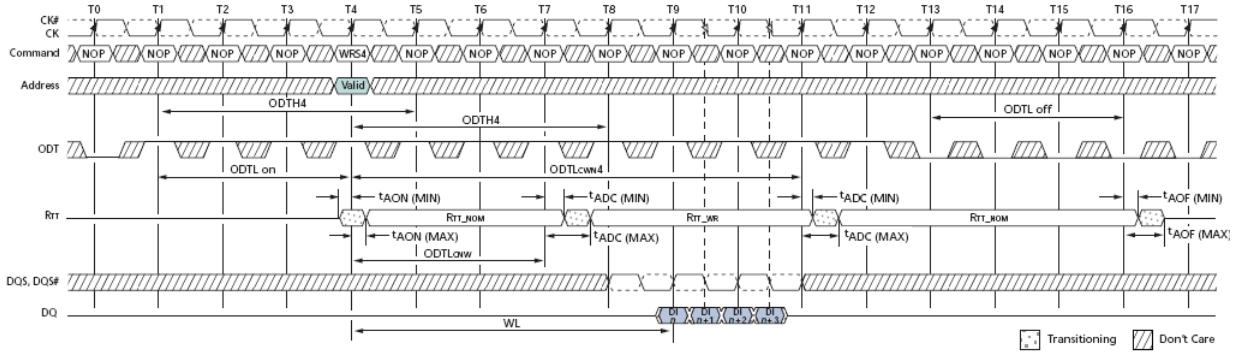
For more information about using ZQ Calibration in DDR3 SDRAM, refer to the application note by Micron, *TN-41-02 DDR3 ZQ Calibration*.

Dynamic ODT

Dynamic ODT is a new feature in DDR3 SDRAM, and not available in DDR2 SDRAM. Dynamic ODT can change the ODT setting without issuing a mode register set (MRS) command. When you enable dynamic ODT, and there is no write operation, the DDR3 SDRAM terminates to a termination setting of `RTT_NORM`; when there is a write operation, the DDR3 SDRAM terminates to a setting of `RTT_WR`. You can preset the values of `RTT_NORM` and `RTT_WR` by programming the mode registers, MR1 and MR2.

Figure 4–3 shows the behavior of ODT when you enable dynamic ODT.

Figure 4–3. Dynamic ODT: Behavior with ODT Asserted Before and After the Write *(1)*



Note to Figure 4–3:

(1) Source: *TN-41-04 DDR3 Dynamic On-Die Termination*, Micron.

In the two-DIMM DDR3 SDRAM configuration, dynamic ODT helps reduce the jitter at the module being accessed, and minimizes reflections from any secondary modules.

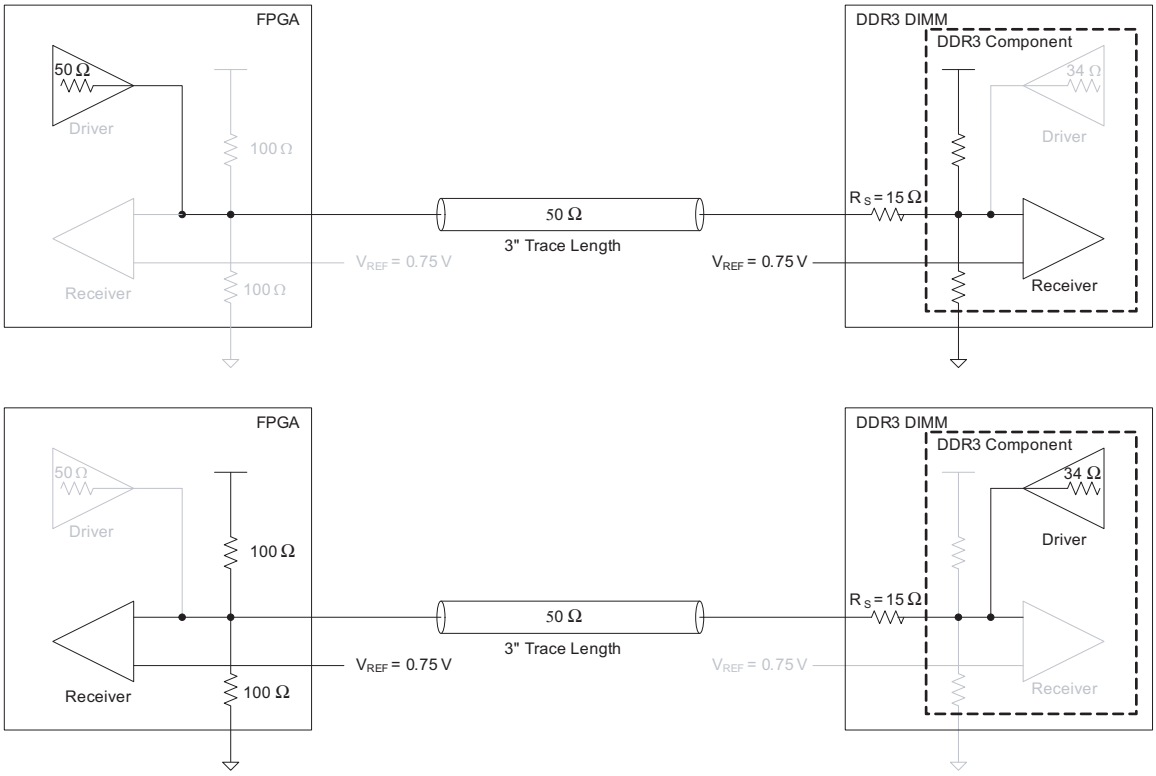
For more information about using the dynamic ODT on DDR3 SDRAM, refer to the application note by Micron, *TN-41-04 DDR3 Dynamic On-Die Termination*.

Dynamic OCT in Stratix III and Stratix IV Devices

Stratix III and Stratix IV devices support on-off dynamic series and parallel termination for a bidirectional I/O in all I/O banks. Dynamic OCT is a new feature in Stratix III and Stratix IV FPGA devices. You enable dynamic parallel termination only when the bidirectional I/O acts as a receiver and disable it when the bidirectional I/O acts as a driver. Similarly, you enable dynamic series termination only when the bidirectional I/O acts as a driver and is disable it when the bidirectional I/O acts as a receiver. The default setting for dynamic OCT is series termination, to save power when the interface is idle—no active reads or writes.

Additionally, the dynamic control operation of the OCT is separate to the output enable signal for the buffer. Hence, UniPHY IP can only enable parallel OCT during read cycles, saving power when the interface is idle.

Figure 4–4. Dynamic OCT Between Stratix III and Stratix IV FPGA Devices



This feature is useful for terminating any high-performance bidirectional path because signal integrity is optimized depending on the direction of the data. In addition, dynamic OCT also eliminates the need for external termination resistors when used with memory devices that support ODT (such as DDR3 SDRAM), thus reducing cost and easing board layout.

However, dynamic OCT in Stratix III and Stratix IV FPGA devices is different from dynamic ODT in DDR3 SDRAM mentioned in previous sections and these features should not be assumed to be identical.

For detailed information about the dynamic OCT feature in the Stratix III FPGA, refer to the *Stratix III Device I/O Features* chapter in volume 1 of the *Stratix III Device Handbook*.

For detailed information about the dynamic OCT feature in the Stratix IV FPGA, refer to the *I/O Features in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

Dynamic OCT in Stratix V Devices

Stratix V devices also support dynamic OCT feature and provide more flexibility. Stratix V OCT calibration uses one RZQ pin that exists in every OCT block. You can use any one of the following as a reference resistor on the RZQ pin to implement different OCT values:

- 240-Ω reference resistor—to implement R_S OCT of 34 Ω, 40 Ω, 48 Ω, 60 Ω, and 80 Ω; and R_T OCT resistance of 20 Ω, 30 Ω, 40 Ω, and 120 Ω
- 100 Ω reference resistor—to implement R_S OCT of 25 Ω and 50 Ω; and R_T OCT resistance of 50 Ω

For detailed information about the dynamic OCT feature in the Stratix V FPGA, refer to the *I/O Features in Stratix V Devices* chapter in volume 1 of the *Stratix V Device Handbook*.

Board Termination for DDR2 SDRAM

DDR2 adheres to the JEDEC standard of governing Stub-Series Terminated Logic (SSTL), JESD8-15a, which includes four different termination schemes.

Two commonly used termination schemes of SSTL are:

- Single parallel terminated output load with or without series resistors (Class I, as stated in JESD8-15a)
- Double parallel terminated output load with or without series resistors (Class II, as stated in JESD8-15a)

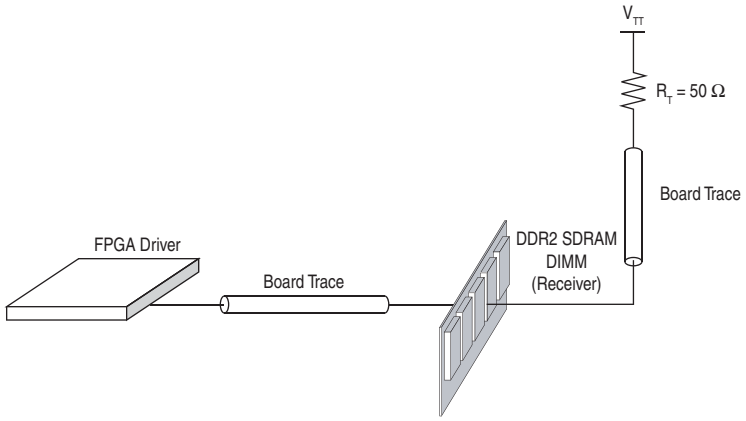
Depending on the type of signals you choose, you can use either termination scheme. Also, depending on your design's FPGA and SDRAM memory devices, you may choose external or internal termination schemes.

With the ever-increasing requirements to reduce system cost and simplify printed circuit board (PCB) layout design, you may choose not to have any parallel termination on the transmission line, and use point-to-point connections between the memory interface and the memory. In this case, you may take advantage of internal termination schemes such as on-chip termination (OCT) on the FPGA side and on-die termination (ODT) on the SDRAM side when it is offered on your chosen device.

External Parallel Termination

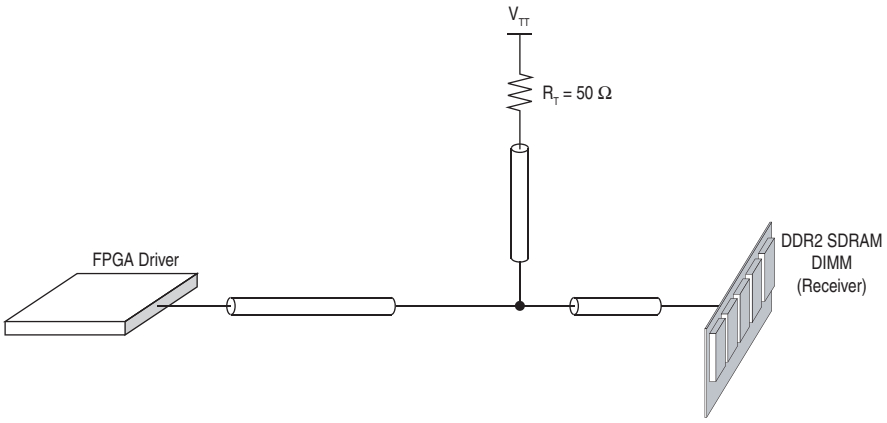
If you use external termination, you must study the locations of the termination resistors to determine which topology works best for your design. Figure 4–5 and Figure 4–6 illustrate the two most commonly used termination topologies: fly-by topology and non-fly-by topology, respectively.

Figure 4–5. Fly-By Placement of a Parallel Resistor



With fly-by topology (Figure 4–5), you place the parallel termination resistor after the receiver. This termination placement resolves the undesirable unterminated stub found in the non-fly-by topology. However, using this topology can be costly and complicate routing. The Stratix II Memory Board 2 uses the fly-by topology for the parallel terminating resistors placement. The Stratix II Memory Board 2 is a memory test board available only within Altera for the purpose of testing and validating Altera's memory interface.

Figure 4–6. Non-Fly-By Placement of a Parallel Resistor



With non-fly-by topology (Figure 4–6), the parallel termination resistor is placed between the driver and receiver (closest to the receiver). This termination placement is easier for board layout, but results in a short stub, which causes an unterminated transmission line between the terminating resistor and the receiver. The unterminated transmission line results in ringing and reflection at the receiver.

If you do not use external termination, DDR2 offers ODT and Altera FPGAs have varying levels of OCT support. You should explore using ODT and OCT to decrease the board power consumption and reduce the required board real estate.

On-Chip Termination

OCT technology is offered on Arria II GX, Arria II GZ, Arria V, Cyclone III, Cyclone IV, Cyclone V, Stratix III, Stratix IV, and Stratix V devices. Table 4–1 summarizes the extent of OCT support for each device. This table provides information about SSTL-18 standards because SSTL-18 is the supported standard for DDR2 memory interface by Altera FPGAs.

On-chip series (R_S) termination is supported only on output and bidirectional buffers. The value of R_S with calibration is calibrated against a 25-Ω resistor for class II and 50-Ω resistor for class I connected to R_{UP} and R_{DN} pins and adjusted to ± 1% of 25 Ω or 50 Ω. On-chip parallel (R_T) termination is supported only on inputs and bidirectional buffers. The value of R_T is calibrated against 100 Ω connected to the R_{UP} and R_{DN} pins. Calibration occurs at the end of device configuration. Dynamic OCT is supported only on bidirectional I/O buffers.

Table 4–1. On-Chip Termination Schemes

Termination Scheme	SSTL-18	FPGA Device						
		Arria II GX	Arria II GZ	Arria V	Cyclone III and Cyclone IV	Cyclone V	Stratix III and Stratix IV	Stratix V (1)
		Column and Row I/O	Column and Row I/O	Column and Row I/O	Column and Row I/O	Column and Row I/O	Column and Row I/O	Column I/O
On-Chip Series Termination without Calibration	Class I	50	50	50	50	50	50	50
	Class II	25	25	25	25	25	25	25
On-Chip Series Termination with Calibration	Class I	50	50	50	50	50	50	50
	Class II	25	25	25	25	25	25	25
On-Chip Parallel Termination with Calibration	Class I and Class II	—	50	50	—	50	50	50

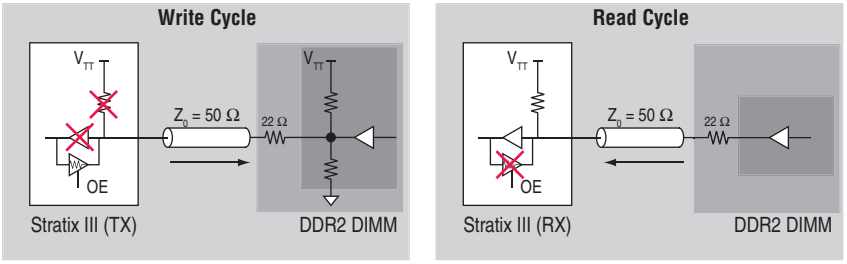
Note to Table 4–1:

(1) Row I/O is not available for external memory interfaces in Stratix V devices.

The dynamic OCT scheme is only available in Stratix III, Stratix IV, and Stratix V FPGAs. The dynamic OCT scheme enables series termination (R_S) and parallel termination (R_T) to be dynamically turned on and off during the data transfer.

The series and parallel terminations are turned on or off depending on the read and write cycle of the interface. During the write cycle, the R_S is turned on and the R_T is turned off to match the line impedance. During the read cycle, the R_S is turned off and the R_T is turned on as the Stratix III FPGA implements the far-end termination of the bus (Figure 4–7).

Figure 4–7. Dynamic OCT for Memory Interfaces



Recommended Termination Schemes

Table 4–2 provides the recommended termination schemes for major DDR2 memory interface signals. Signals include data (DQ), data strobe (DQS/DQSn), data mask (DM), clocks (`mem_clk/mem_clk_n`), and address and command signals.

When interfacing with multiple DDR2 SDRAM components where the address, command, and memory clock pins are connected to more than one load, follow these steps:

1. Simulate the system to get the new slew-rate for these signals.
2. Use the derated tIS and tIH specifications from the DDR2 SDRAM datasheet based on the simulation results.
3. If timing deration causes your interface to fail timing requirements, consider signal duplication of these signals to lower their loading, and hence improve timing.

Altera uses Class I and Class II termination in this table to refer to drive strength, and not physical termination.

You must simulate your design for your system to ensure correct functionality.

Table 4–2. Termination Recommendations (Part 1 of 3) (1)

Device Family	Signal Type	SSTL 18 IO Standard (2), (3), (4), (5), (6)	FPGA-End Discrete Termination	Memory-End Termination 1 (Rank/DIMM)	Memory I/O Standard
Arria II GX					
DDR2 component	DQ	Class I R50 CAL	50 Ω Parallel to V_{TT} discrete	ODT75 (7)	HALF (8)
	DQS DIFF (13)	DIFF Class R50 CAL	50 Ω Parallel to V_{TT} discrete	ODT75 (7)	HALF (8)
	DQS SE (12)	Class I R50 CAL	50 Ω Parallel to V_{TT} discrete	ODT75 (7)	HALF (8)
	DM	Class I R50 CAL	N/A	ODT75 (7)	N/A
	Address and command	Class I MAX	N/A	56 Ω parallel to V_{TT} discrete	N/A
	Clock	DIFF Class I R50 CAL	N/A	×1 = 100 Ω differential (10) ×2 = 200 Ω differential (11)	N/A
DDR2 DIMM	DQ	Class I R50 CAL	50 Ω Parallel to V_{TT} discrete	ODT75 (7)	FULL (9)
	DQS DIFF (13)	DIFF Class I R50 CAL	50 Ω Parallel to V_{TT} discrete	ODT75 (7)	FULL (9)
	DQS SE (12)	Class I R50 CAL	50 Ω Parallel to V_{TT} discrete	ODT75 (7)	FULL (9)
	DM	Class I R50 CAL	N/A	ODT75 (7)	N/A
	Address and command	Class I MAX	N/A	56 Ω parallel to V_{TT} discrete	N/A
	Clock	DIFF Class I R50 CAL	N/A	N/A = on DIMM	N/A
Arria V and Cyclone V					
DDR2 component	DQ	Class I R50/P50 DYN CAL	N/A	ODT75 (7)	HALF (8)
	DQS DIFF (13)	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 (7)	HALF (8)
	DQS SE (12)	Class I R50/P50 DYN CAL	N/A	ODT75 (7)	HALF (8)
	DM	Class I R50 CAL	N/A	ODT75 (7)	N/A
	Address and command	Class I MAX	N/A	56 Ω parallel to V_{TT} discrete	N/A
	Clock	DIFF Class I R50 NO CAL	N/A	×1 = 100 Ω differential (10) ×2 = 200 Ω differential (11)	N/A

Table 4–2. Termination Recommendations (Part 2 of 3) (1)

Device Family	Signal Type	SSTL 18 IO Standard (2), (3), (4), (5), (6)	FPGA-End Discrete Termination	Memory-End Termination 1 (Rank/DIMM)	Memory I/O Standard
DDR2 DIMM	DQ	Class I R50/P50 DYN CAL	N/A	ODT75 (7)	FULL (9)
	DQS DIFF (13)	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 (7)	FULL (9)
	DQS SE (12)	Class I R50/P50 DYN CAL	N/A	ODT75 (7)	FULL (9)
	DM	Class I R50 CAL	N/A	ODT75 (7)	N/A
	Address and command	Class I MAX	N/A	56 Ω parallel to V_{TT} discrete	N/A
	Clock	DIFF Class I R50 NO CAL	N/A	N/A = on DIMM	N/A
Cyclone III and Cyclone IV					
DDR2 component	DQ/DQS	Class I 12 mA	50 Ω Parallel to V_{TT} discrete	ODT75 (7)	HALF (8)
	DM	Class I 12 mA	N/A	56 Ω parallel to V_{TT} discrete	N/A
	Address and command	Class I MAX	N/A		N/A
	Clock	Class I 12 mA	N/A	×1 = 100 Ω differential (10) ×2 = 200 Ω differential (11)	N/A
DDR2 DIMM	DQ/DQS	Class I 12 mA	50 Ω Parallel to V_{TT} discrete	ODT75 (7)	FULL (9)
	DM	Class I12 mA	N/A	56 Ω parallel to V_{TT} discrete	N/A
	Address and command	Class I MAX	N/A		N/A
	Clock	Class I 12 mA	N/A	N/A = on DIMM	N/A
Arria II GZ, Stratix III, Stratix IV, and Stratix V					
DDR2 component	DQ	Class I R50/P50 DYN CAL	N/A	ODT75 (7)	HALF (8)
	DQS DIFF (13)	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 (7)	HALF (8)
	DQS SE (12)	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 (7)	HALF (8)
	DM	Class I R50 CAL	N/A	ODT75 (7)	N/A
	Address and command	Class I MAX	N/A	56 Ω Parallel to V_{TT} discrete	N/A
	Clock	DIFF Class I R50 NO CAL	N/A	x1 = 100 Ω differential (10) x2 = 200 Ω differential (11)	N/A

Table 4–2. Termination Recommendations (Part 3 of 3) *(1)*

Device Family	Signal Type	SSTL 18 IO Standard *(2)*, *(3)*, *(4)*, *(5)*, *(6)*	FPGA-End Discrete Termination	Memory-End Termination 1 (Rank/DIMM)	Memory I/O Standard
DDR2 DIMM	DQ	Class I R50/P50 DYN CAL	N/A	ODT75 *(7)*	FULL *(9)*
	DQS DIFF *(13)*	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 *(7)*	FULL *(9)*
	DQS SE *(12)*	Class I R50/P50 DYN CAL	N/A	ODT75 *(7)*	FULL *(9)*
	DM	Class I R50 CAL	N/A	ODT75 *(7)*	N/A
	Address and command	Class I MAX	N/A	56 Ω Parallel to V_{TT} discrete	N/A
	Clock	DIFF Class I R50 NO CAL	N/A	N/A = on DIMM	N/A

Notes to Table 4–2:

(1) N/A is not available.
(2) R is series resistor.
(3) P is parallel resistor.
(4) DYN is dynamic OCT.
(5) NO CAL is OCT without calibration.
(6) CAL is OCT with calibration.
(7) ODT75 vs. ODT50 on the memory has the effect of opening the eye more, with a limited increase in overshoot/undershoot.
(8) HALF is reduced drive strength.
(9) FULL is full drive strength.
(10) x1 is a single-device load.
(11) x2 is two-device load. For example, you can feed two out of nine devices on a single rank DIMM with a single clock pair.
(12) DQS SE is single-ended DQS.
(13) DQS DIFF is differential DQS

Dynamic On-Chip Termination

The termination schemes are described in JEDEC standard JESD8-15a for SSTL 18 I/O. Dynamic OCT is available in Stratix III and Stratix IV. When the Stratix III FPGA (driver) is writing to the DDR2 SDRAM DIMM (receiver), series OCT is enabled dynamically to match the impedance of the transmission line. As a result, reflections are significantly reduced. Similarly, when the FPGA is reading from the DDR2 SDRAM DIMM, the parallel OCT is dynamically enabled.

For information about setting the proper value for termination resistors, refer to the *Stratix III Device I/O Features* chapter in the *Stratix III Device Handbook* and the *I/O Features in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*.

FPGA Writing to Memory

Figure 4–8 shows dynamic series OCT scheme when the FPGA is writing to the memory. The benefit of using dynamic series OCT is that when driver is driving the transmission line, it "sees" a matched transmission line with no external resistor termination.

Figure 4–8. Dynamic Series OCT Scheme with ODT on the Memory

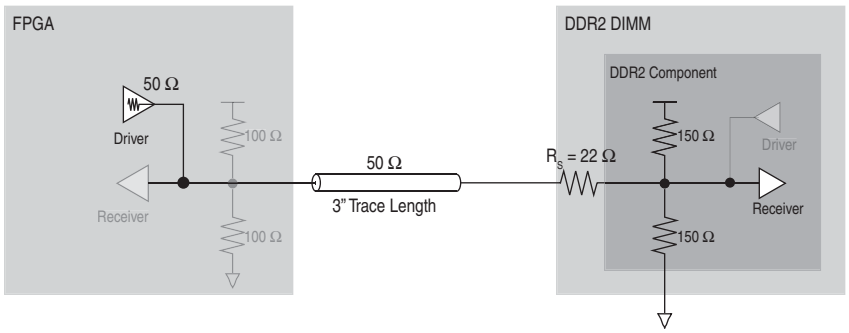


Figure 4–9 and Figure 4–10 show the simulation and measurement results of a write to the DDR2 SDRAM DIMM. The system uses Class I termination with a 50-Ω series OCT measured at the FPGA with a full drive strength and a 75 Ω ODT at the DIMM. Both simulation and bench measurements are in 200 pS/div and 200 mV/div.

Figure 4–9. HyperLynx Simulation FPGA Writing to Memory

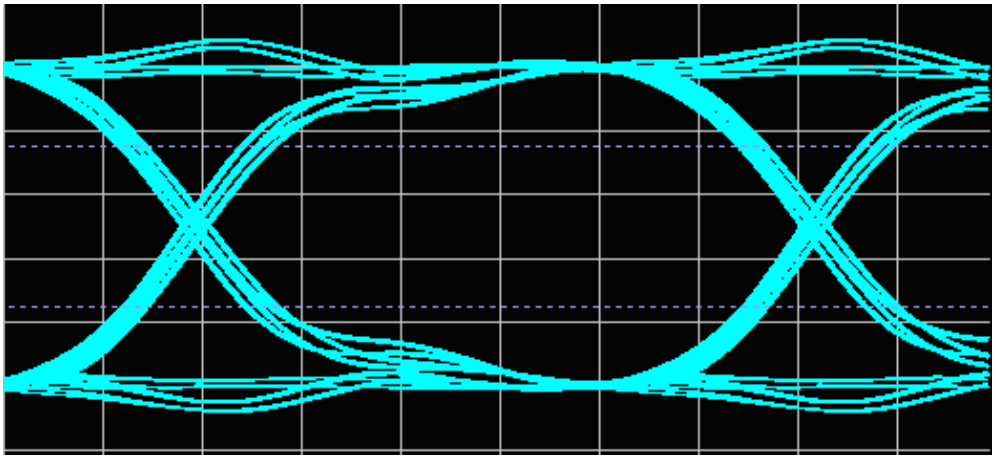


Figure 4–10. Board Measurement, FPGA Writing to Memory

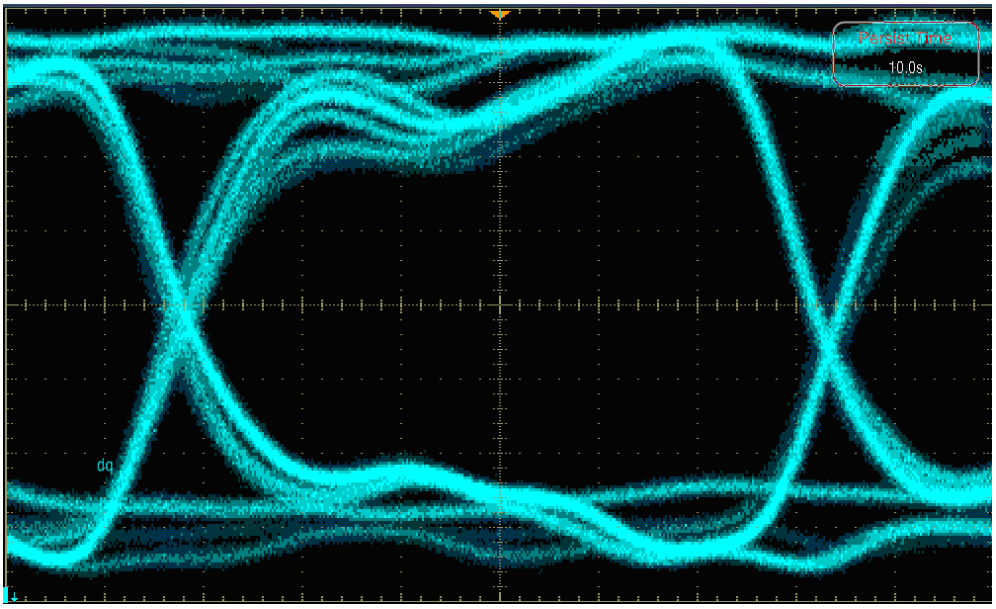


Table 4–3 lists the comparison between the simulation and the board measurement of the signal seen at the DDR2 SDRAM DIMM.

Table 4–3. Signal Comparison When the FPGA is Writing to the Memory *(1)*

	Eye Width (ns) *(2)*	Eye Height (V)	Overshoot (V)	Undershoot (V)
Simulation	1.194	0.740	N/A	N/A
Board Measurement	1.08	0.7	N/A	N/A

Notes to Table 4–3:

(1) N/A is not applicable.

(2) The eye width is measured from V_{IH}/V_{IL}(ac) = VREF ±250 mV to V_{IH}/V_{IL}(dc) = VREF ±125 mV, where V_{IH} and V_{IL} are determined per the JEDEC specification for SSTL-18.

The data in Table 4–3 and Figure 4–9 and Figure 4–10 suggest that when the FPGA is writing to the memory, the bench measurements are closely matched with simulation measurements. They indicate that using the series dynamic on-chip termination scheme for your bidirectional I/Os maintains the integrity of the signal, while it removes the need for external termination.

Depending on the I/O standard, you should consider the four parameters listed in Table 4–3 when designing a memory interface. Although the simulation and board measurement appear to be similar, there are some discrepancies when the key parameters are measured. Although simulation does not fully model the duty cycle distortion of the I/O, crosstalk, or board power plane degradation, it provides a good indication on the performance of the board.

For memory interfaces, the eye width is important when determining if there is a sufficient window to correctly capture the data. Regarding the eye height, even though most memory interfaces use voltage-referenced I/O standards (in this case, SSTL-18), as long as there is sufficient eye opening below and above VIL and VIH, there should be enough margin to correctly capture the data. However, because effects such as crosstalk are not taken into account, it is critical to design a system to achieve the optimum eye height, because it impacts the overall margin of a system with a memory interface.

Refer to the memory vendors when determining the over- and undershoot. They typically specify a maximum limit on the input voltage to prevent reliability issues.

FPGA Reading from Memory

Figure 4–11 shows the dynamic parallel termination scheme when the FPGA is reading from memory. When the DDR2 SDRAM DIMM is driving the transmission line, the ringing and reflection is minimal because the FPGA-side termination 50-Ω pull-up resistor is matched with the transmission line. Figure 4–12 shows the simulation and measurement results of a read from DDR2 SDRAM DIMM. The system uses Class I termination with a 50-Ω calibrated parallel OCT measured at the FPGA end with a full drive strength and a 75-Ω ODT at the memory. Both simulation and bench measurements are in 200 pS/div and 200 mV/div.

Figure 4–11. Dynamic Parallel OCT Scheme with Memory-Side Series Resistor

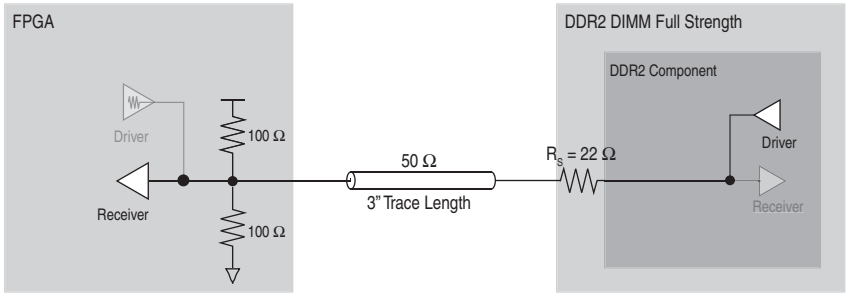


Figure 4–12. Hyperlynx Simulation and Board Measurement, FPGA Reading from Memory

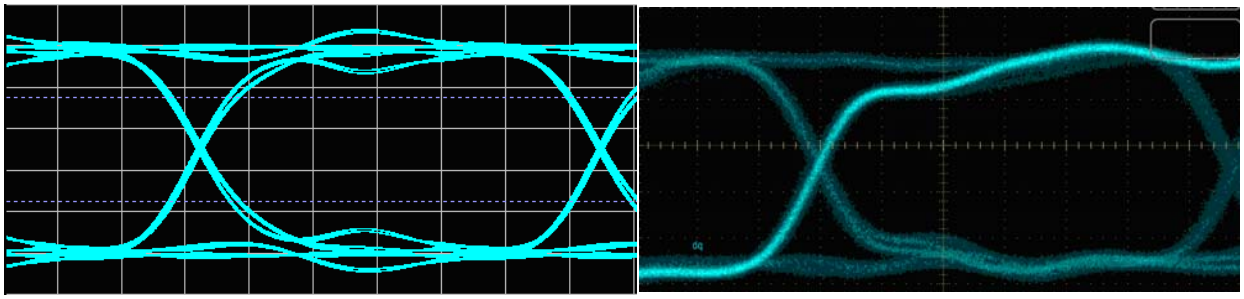


Table 4–4 lists the comparison between the simulation and the board measurement of the signal seen at the FPGA end.

Table 4–4. Signal Comparison When the FPGA is Reading from the Memory (1), (2)

	Eye Width (ns) (3)	Eye Height (V)	Overshoot (V)	Undershoot (V)
Simulation	1.206	0.740	N/A	N/A
Board Measurement	1.140	0.680	N/A	N/A

Notes to Table 4–4:

(1) The drive strength on the memory DIMM is set to Full.

(2) N/A is not applicable.

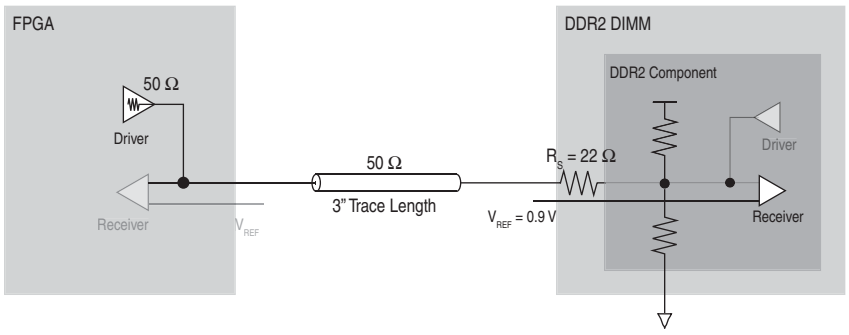
(3) The eye width is measured from V_{IH}/V_{IL}(ac) = VREF ±250 mV to V_{IH}/V_{IL}(dc) = VREF ±125 mV, in which V_{IH} and V_{IL} are determined per the JEDEC specification for SSTL-18.

The data in Table 4–4 and Figure 4–13 suggest that bench measurements are closely matched with simulation measurements when the FPGA is reading from the memory. They indicate that using the parallel dynamic on-chip termination scheme in bidirectional I/Os maintains the integrity of the signal, while it removes the need for external termination.

On-Chip Termination (Non-Dynamic)

When you use the 50-ΩOCT feature in a Class I termination scheme using ODT with a memory-side series resistor, the output driver is tuned to 50 Ω, which matches the characteristic impedance of the transmission line. Figure 4–13 shows the Class I termination scheme using ODT when the 50-ΩOCT on the FPGA is turned on.

Figure 4–13. Class I Termination Using ODT with 50-ΩOCT



The resulting signal quality has a similar eye opening to the 8 mA drive strength setting (refer to "Drive Strength" on page 4–50) without any over- or undershoot. Figure 4–14 shows the simulation and measurement of the signal at the memory side (DDR2 SDRAM DIMM) with the drive strength setting of 50-Ω OCT in the FPGA.

Figure 4–14. HyperLynx Simulation and Measurement, FPGA Writing to Memory

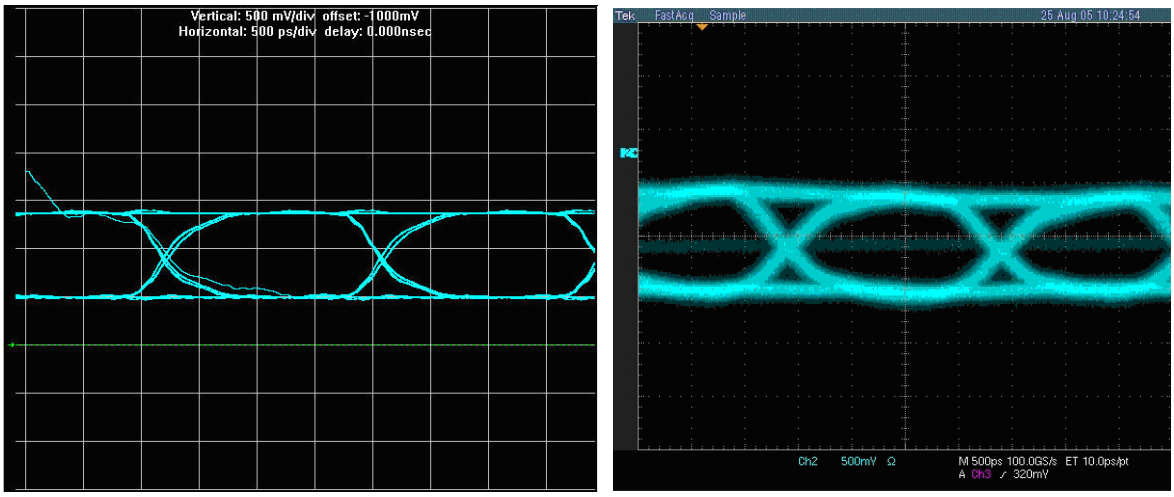


Table 4–5 lists data for the signal at the DDR2 SDRAM DIMM of a Class I scheme termination using ODT with a memory-side series resistor. The FPGA is writing to the memory with 50-Ω OCT.

Table 4–5. Simulation and Board Measurement Results for 50-Ω OCT and 8-mA Drive Strength Settings *(1)*

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
50-Ω OCT Drive Strength Setting				
Simulation	1.68	0.82	N/A	N/A
Board Measurement	1.30	0.70	N/A	N/A

Note to Table 4–5:

(1) N/A is not applicable.

When you use the 50-Ω OCT setting on the FPGA, the signal quality for the Class I termination using ODT with a memory-side series resistor is further improved with lower over- and undershoot.

In addition to the 50-Ω OCT setting, Stratix II devices have a 25-ΩOCT setting that you can use to improve the signal quality in a Class II terminated transmission line. Figure 4–15 shows the Class II termination scheme using ODT when the 25-ΩOCT on the FPGA is turned on.

Figure 4–15. Class II Termination Using ODT with 25-ΩOCT

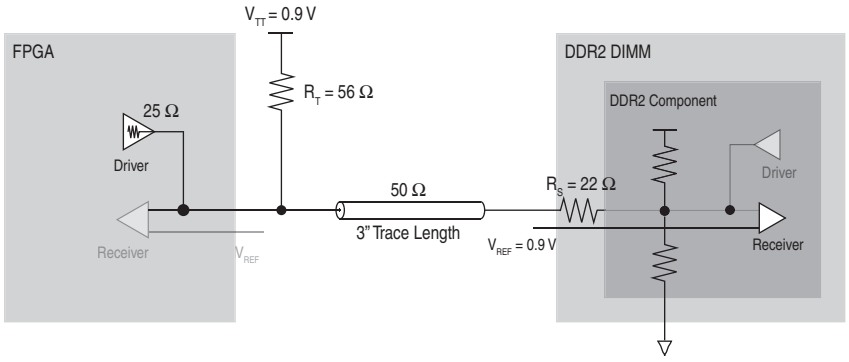


Figure 4–16 shows the simulation and measurement of the signal at the DDR2 SDRAM DIMM (receiver) with a drive strength setting of 25-ΩOCT in the FPGA.

Figure 4–16. HyperLynx Simulation and Measurement, FPGA Writing to Memory

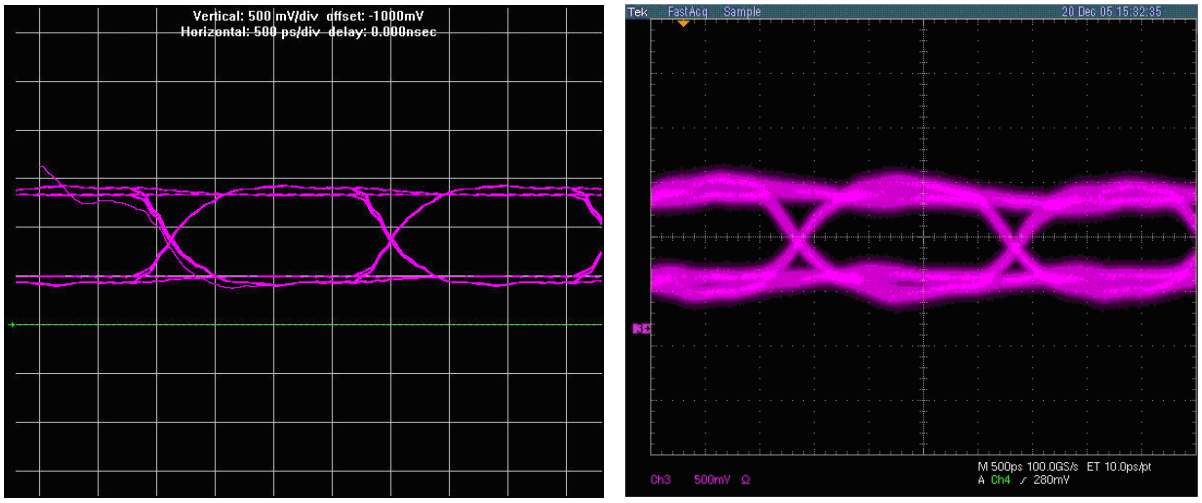


Table 4–6 lists the data for the signal at the DDR2 SDRAM DIMM of a Class II termination with a memory-side series resistor. The FPGA is writing to the memory with 25-Ω OCT.

Table 4–6. Simulation and Board Measurement Results for 25-Ω OCT and 16-mA Drive Strength Settings *(1)*

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
25-Ω OCT Drive Strength Setting				
Simulation	1.70	0.81	N/A	N/A
Board Measurement	1.47	0.51	N/A	N/A

Note to Table 4–6:

(1) N/A is not applicable.

This type of termination scheme is only used for bidirectional signals, such as data (DQ), data strobe (DQS), data mask (DM), and memory clocks (CK) found in DRAMs.

Class II External Parallel Termination

The double parallel (Class II) termination scheme is described in JEDEC standards JESD8-6 for HSTL I/O, JESD8-9b for SSTL-2 I/O, and JESD8-15a for SSTL-18 I/O. When the FPGA (driver) is writing to the DDR2 SDRAM DIMM (receiver), the transmission line is terminated at the DDR2 SDRAM DIMM. Similarly, when the FPGA is reading from the DDR2 SDRAM DIMM, the DDR2 SDRAM DIMM is now the driver and the transmission line is terminated at the FPGA (receiver). This type of termination scheme is typically used for bidirectional signals, such as data (DQ) and data strobe (DQS) signal found in DRAMs.

FPGA Writing to Memory

Figure 4–17 shows the Class II termination scheme when the FPGA is writing to the memory. The benefit of using Class II termination is that when either driver is driving the transmission line, it sees a matched transmission line because of the termination resistor at the receiver-end, thereby reducing ringing and reflection.

Figure 4–17. Class-II Termination Scheme with Memory-Side Series Resistor

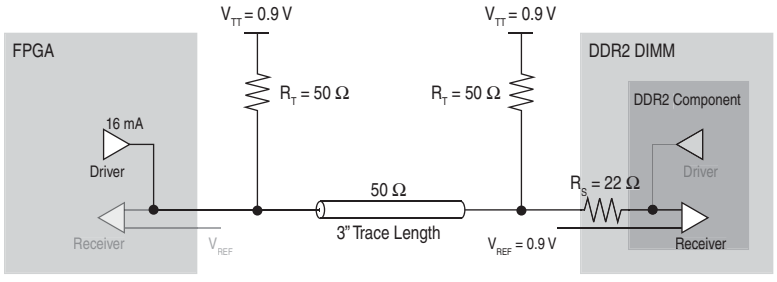
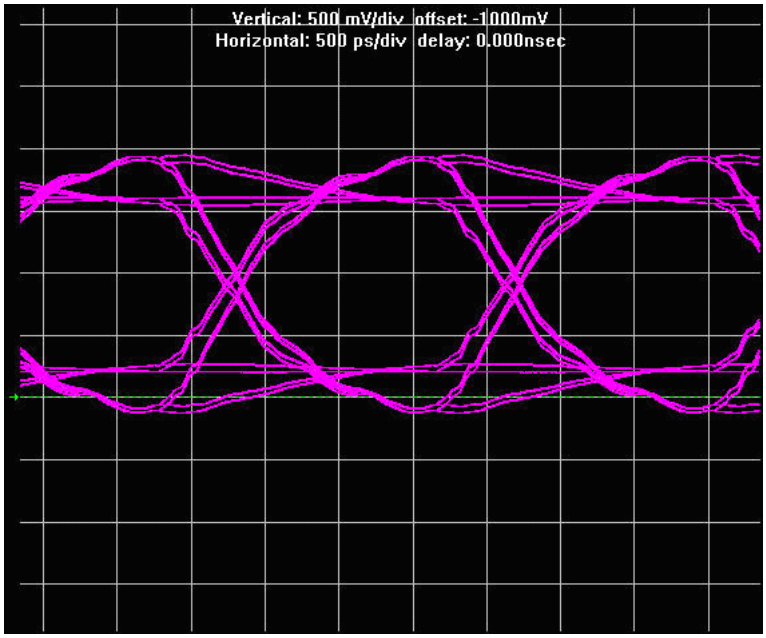


Figure 4–18 and Figure 4–19 show the simulation and measurement result of a write to the DDR2 SDRAM DIMM. The system uses Class II termination with a source-series resistor measured at the DIMM with a drive strength setting of 16 mA.

Figure 4–18. HyperLynx Simulation, FPGA Writing to Memory



The simulation shows a clean signal with a good eye opening, but there is slight over- and undershoot of the 1.8-V signal specified by DDR2 SDRAM. The over- and undershoot can be attributed to either overdriving the transmission line using a higher than required drive strength setting on the driver or the over-termination on the receiver side by using an external resistor value that is higher than the characteristic impedance of the transmission line. As long as the over- and undershoot do not exceed the absolute maximum rating specification listed in the memory vendor's DDR2 SDRAM data sheet, it does not result in any reliability issues. The simulation results are then correlated with actual board level measurements.

Figure 4–19 shows the measurement obtained from the Stratix II Memory Board 2. The FPGA is using a 16 mA drive strength to drive the DDR2 SDRAM DIMM on a Class II termination transmission line.

Figure 4–19. Board Measurement, FPGA Writing to Memory

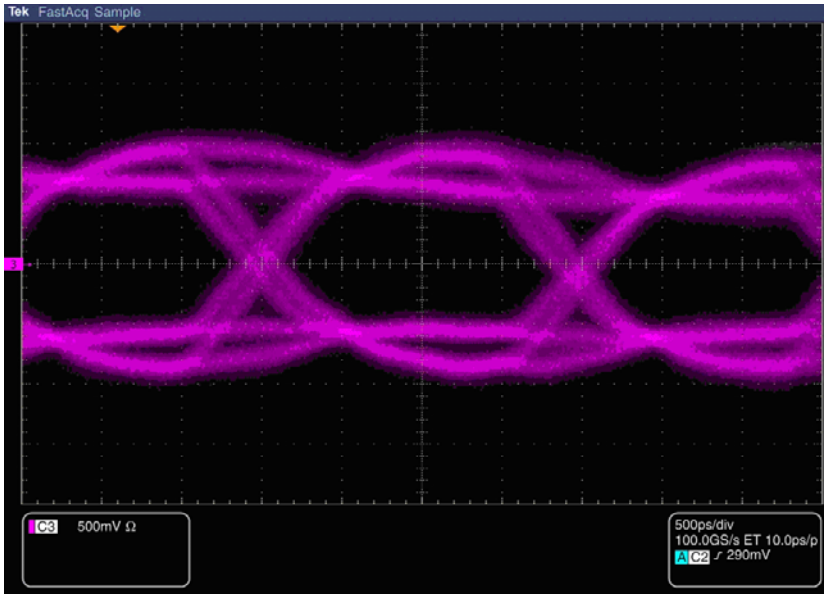


Table 4–7 lists the comparison between the simulation and the board measurement of the signal seen at the DDR2 SDRAM DIMM.

Table 4–7. Signal Comparison When the FPGA is Writing to the Memory (1)

	Eye Width (ns) (2)	Eye Height (V)	Overshoot (V)	Undershoot (V)
Simulation	1.65	1.28	0.16	0.14
Board Measurement	1.35	0.83	0.16	0.18

Notes to Table 4–7:

(1) The drive strength on the FPGA is set to 16 mA.

(2) The eye width is measured from V_{REF} ± 125 mV where V_{IH} and V_{IL} are determined per the JEDEC specification for SSTL-18.

A closer inspection of the simulation shows an ideal duty cycle of 50%–50%, while the board measurement shows that the duty cycle is non-ideal, around 53%–47%, resulting in the difference between the simulation and measured eye width. In addition, the board measurement is conducted on a 72-bit memory interface, but the simulation is performed on a single I/O.

FPGA Reading from Memory

Figure 4–20 shows the Class II termination scheme when the FPGA is reading from memory. When the DDR2 SDRAM DIMM is driving the transmission line, the ringing and reflection is minimal because of the matched FPGA-side termination pull-up resistor with the transmission line.

Figure 4–20. Class II Termination Scheme with Memory-Side Series Resistor

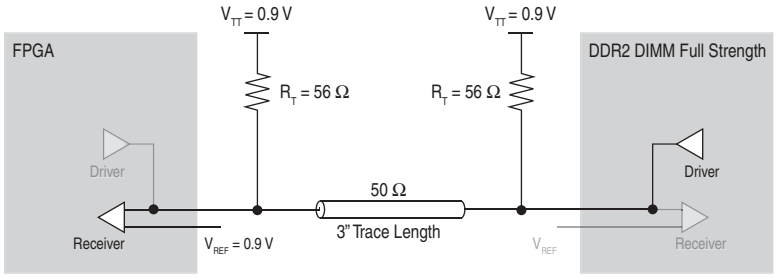


Figure 4–21 and Figure 4–22 show the simulation and measurement, respectively, of the signal at the FPGA side with the full drive strength setting on the DDR2 SDRAM DIMM. The simulation uses a Class II termination scheme with a source-series resistor transmission line. The FPGA is reading from the memory with a full drive strength setting on the DIMM.

Figure 4–21. HyperLynx Simulation, FPGA Reading from Memory

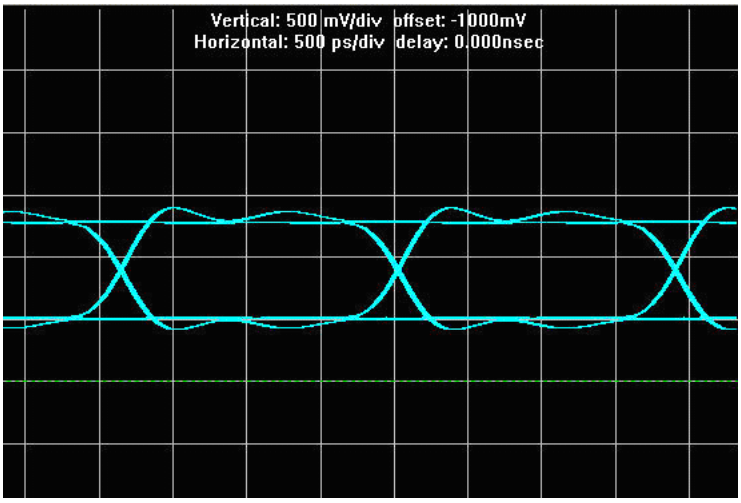


Figure 4–22. Board Measurement, FPGA Reading from Memory

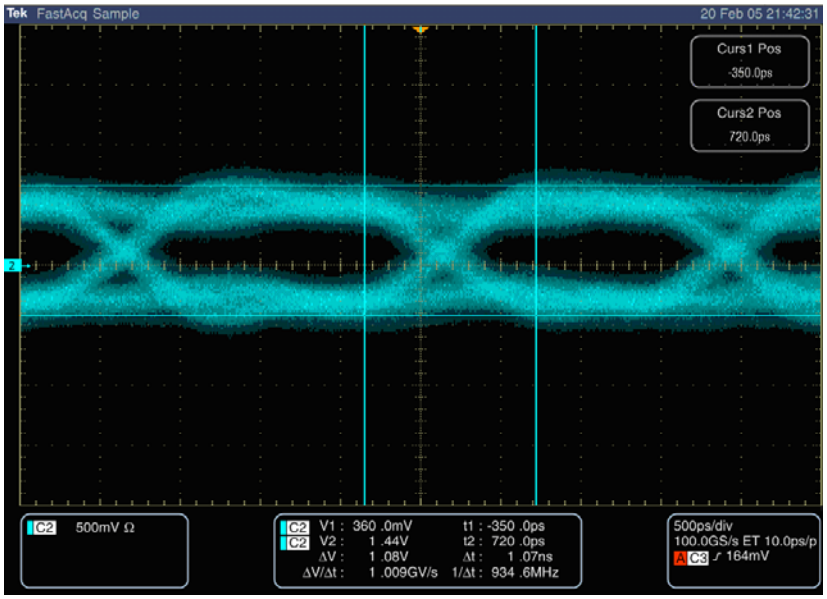


Table 4–8 lists the comparison between the simulation and board measurements of the signal seen by the FPGA when the FPGA is reading from memory (driver).

Table 4–8. Signal Comparison, FPGA is Reading from Memory (1), (2)

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
Simulation	1.73	0.76	N/A	N/A
Board Measurement	1.28	0.43	N/A	N/A

Notes to Table 4–8:

(1) The drive strength on the DDR2 SDRAM DIMM is set to full strength.

(2) N/A is not applicable.

Both simulation and measurement show a clean signal and a good eye opening without any over- and undershoot. However, the eye height when the FPGA is reading from the memory is smaller compared to the eye height when the FPGA is writing to the memory. The reduction in eye height is attributed to the voltage drop on the series resistor present on the DIMM. With the drive strength setting on the memory already set to full, you cannot increase the memory drive strength to improve the eye height. One option is to remove the series resistor on the DIMM when the FPGA is reading from memory (refer to the section "Component Versus DIMM" on page 4–52). Another option is to remove the external parallel resistor near the memory so that the memory driver sees less loading. For a DIMM configuration, the latter option is a better choice because the series resistors are part of the DIMM and you can easily turn on the ODT feature to use as the termination resistor when the FPGA is writing to the memory and turn off when the FPGA is reading from memory.

The results for the Class II termination scheme demonstrate that the scheme is ideal for bidirectional signals such as data strobe and data for DDR2 SDRAM memory. Terminations at the receiver eliminate reflections back to the driver and suppress any ringing at the receiver.

Class I External Parallel Termination

The single parallel (Class I) termination scheme refers to when the termination is located near the receiver side. Typically, this scheme is used for terminating unidirectional signals (such as clocks, address, and command signals) for DDR2 SDRAM.

However, because of board constraints, this form of termination scheme is sometimes used in bidirectional signals, such as data (DQ) and data strobe (DQS) signals. For bidirectional signals, you can place the termination on either the memory or the FPGA side. This section focuses only on the Class I termination scheme with memory-side termination. The memory-side termination ensures impedance matching when the signal reaches the receiver of the memory. However, when the FPGA is reading from the memory, there is no termination on the FPGA side, resulting in impedance mismatch. This section describes the signal quality of this termination scheme.

FPGA Writing to Memory

When the FPGA is writing to the memory (Figure 4–23), the transmission line is parallel-terminated at the memory side, resulting in minimal reflection on the receiver side because of the matched impedance seen by the transmission line. The benefit of this termination scheme is that only one external resistor is required. Alternatively, you can implement this termination scheme using an ODT resistor instead of an external resistor.

Refer to the section "Class I Termination Using ODT" on page 4–28 for more information about how an ODT resistor compares to an external termination resistor.

Figure 4–23. Class I Termination Scheme with Memory-Side Series Resistor

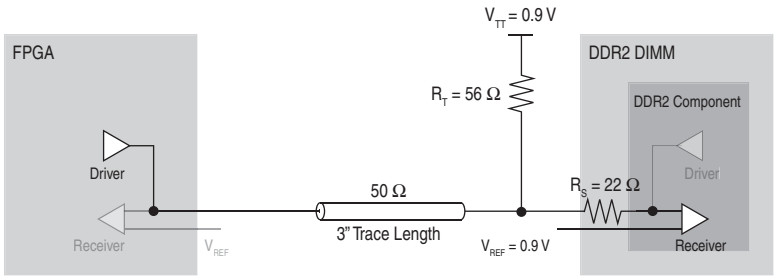


Figure 4–24 shows the simulation and measurement of the signal at the memory (DDR2 SDRAM DIMM) of Class I termination with a memory-side resistor. The FPGA writes to the memory with a 16 mA drive strength setting.

Figure 4–24. HyperLynx Simulation and Board Measurement, FPGA Writing to Memory

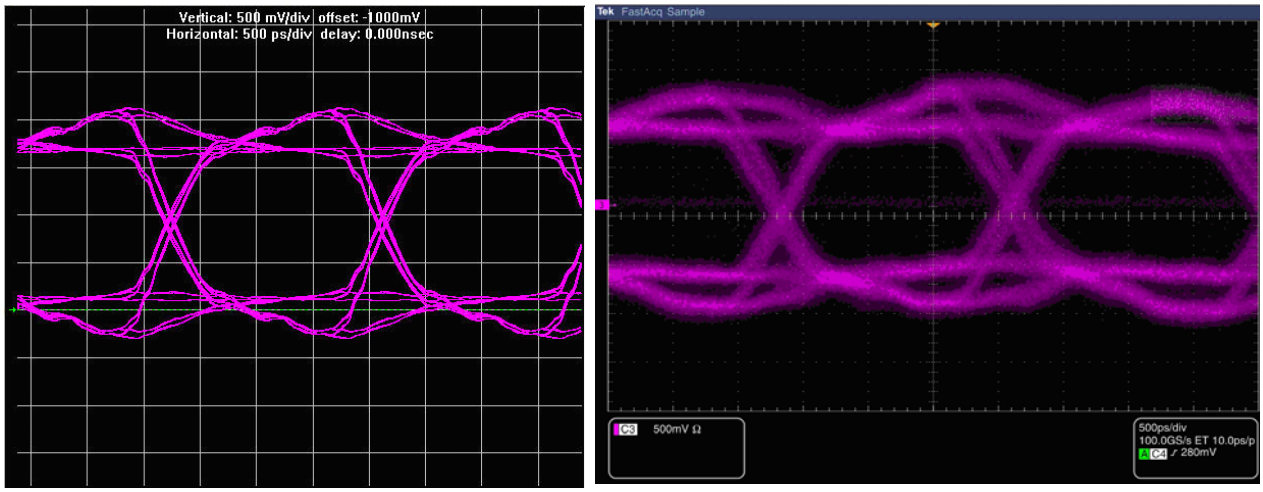


Table 4–9 lists the comparison of the signal at the DDR2 SDRAM DIMM of a Class I and Class II termination scheme using external resistors with memory-side series resistors. The FPGA (driver) writes to the memory (receiver).

Table 4–9. Signal Comparison When the FPGA is Writing to Memory *(1)*

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
Class I Termination Scheme With External Parallel Resistor				
Simulation	1.69	1.51	0.34	0.29
Board Measurement	1.25	1.08	0.41	0.34
Class II Termination Scheme With External Parallel Resistor				
Simulation	1.65	1.28	0.16	0.14
Board Measurement	1.35	0.83	0.16	0.18

Note to Table 4–9:

(1) The drive strength on the FPGA is set to 16 mA.

Table 4–9 lists the overall signal quality of a Class I termination scheme is comparable to the signal quality of a Class II termination scheme, except that the eye height of the Class I termination scheme is approximately 30% larger. The increase in eye height is due to the reduced loading "seen" by the driver, because the Class I termination scheme does not have an FPGA-side parallel termination resistor. However, increased eye height comes with a price: a 50% increase in the over- and undershoot of the signal using Class I versus Class II termination scheme. You can decrease the FPGA drive strength to compensate for the decreased loading seen by the driver to decrease the over- and undershoot.

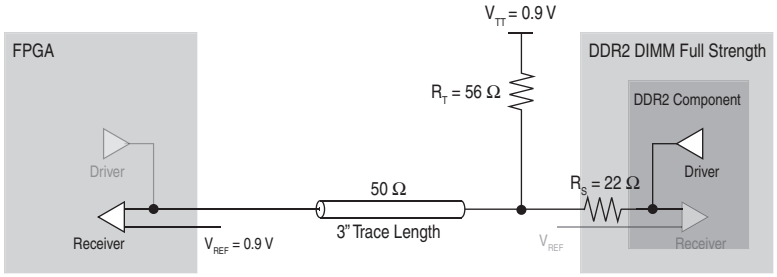
For more information about how drive strength affects the signal quality, refer to "Drive Strength" on page 4–50.

FPGA Reading from Memory

As described in the section "FPGA Writing to Memory" on page 4–25, in Class I termination, the termination is located near the receiver. However, if you use this termination scheme to terminate a bidirectional signal, the receiver can also be the driver. For example, in DDR2 SDRAM, the data signals are both receiver and driver.

Figure 4–25 shows a Class I termination scheme with a memory-side resistor. The FPGA reads from the memory.

Figure 4–25. Class I Termination Scheme with Memory-Side Series Resistor



When the FPGA reads from the memory (Figure 4–25), the transmission line is not terminated at the FPGA, resulting in an impedance mismatch, which then results in over- and undershoot. Figure 4–26 shows the simulation and measurement of the signal at the FPGA side (receiver) of a Class I termination. The FPGA reads from the memory with a full drive strength setting on the DDR2 SDRAM DIMM.

Figure 4–26. HyperLynx Simulation and Board Measurement, FPGA Reading from Memory

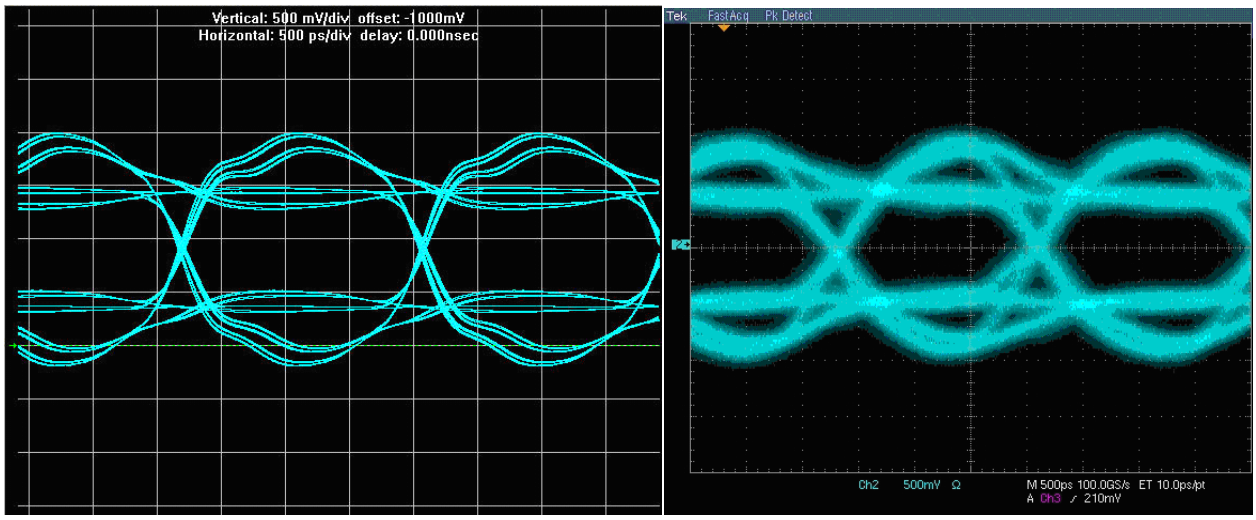


Table 4–10 lists the comparison of the signal "seen" at the FPGA of a Class I and Class II termination scheme using an external resistor with a memory-side series resistor. The FPGA (receiver) reads from the memory (driver).

Table 4–10. Signal Comparison When the FPGA is Reading From Memory (1), (2)

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
Class I Termination Scheme with External Parallel Resistor				
Simulation	1.73	0.74	0.20	0.18
Board Measurement	1.24	0.58	0.09	0.14
Class II Termination Scheme with External Parallel Resistor				
Simulation	1.73	0.76	N/A	N/A
Board Measurement	1.28	0.43	N/A	N/A

Notes to Table 4–10:

(1) The drive strength on the DDR2 SDRAM DIMM is set to full strength.

(2) N/A is not applicable.

When the FPGA reads from the memory using the Class I scheme, the signal quality is comparable to that of the Class II scheme, in terms of the eye height and width. Table 4–10 shows the lack of termination at the receiver (FPGA) results in impedance mismatch, causing reflection and ringing that is not visible in the Class II termination scheme. As such, Altera recommends using the Class I termination scheme for unidirectional signals (such as command and address signals), between the FPGA and the memory.

Class I Termination Using ODT

Presently, ODT is becoming a common feature in memory, including SDRAMs, graphics DRAMs, and SRAMs. ODT helps reduce board termination cost and simplify board routing. This section describes the ODT feature of DDR2 SDRAM and the signal quality when the ODT feature is used.

FPGA Writing to Memory

DDR2 SDRAM has built-in ODT that eliminates the need for external termination resistors. To use the ODT feature of the memory, you must configure the memory to turn on the ODT feature during memory initialization. For DDR2 SDRAM, set the ODT feature by programming the extended mode register. In addition to programming the extended mode register during initialization of the DDR2 SDRAM, an ODT input pin on the DDR2 SDRAM must be driven high to activate the ODT.

For additional information about setting the ODT feature and the timing requirements for driving the ODT pin in DDR2 SDRAM, refer to the respective memory data sheet

The ODT feature in DDR2 SDRAM is controlled dynamically—it is turned on while the FPGA is writing to the memory and turned off while the FPGA is reading from the memory. The ODT feature in DDR2 SDRAM has three settings: 50Ω, 75Ω, and 150Ω. If there are no external parallel termination resistors and the ODT feature is turned on, the termination scheme resembles the Class I termination described in "Class I External Parallel Termination" on page 4–25.

Figure 4–27 shows the termination scheme when the ODT on the DDR2 SDRAM is turned on.

Figure 4–27. Class I Termination Scheme Using ODT

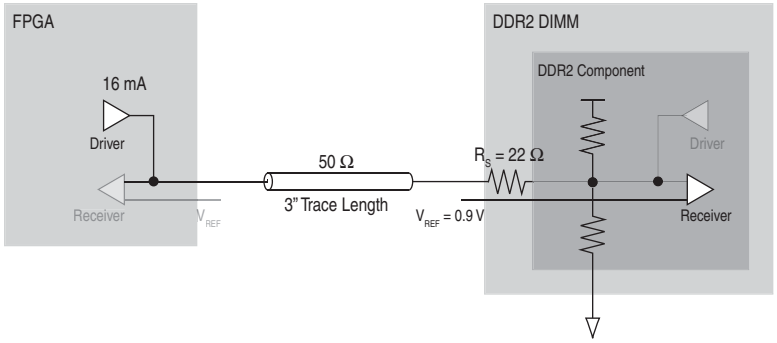


Figure 4–28 shows the simulation and measurement of the signal visible at the memory (receiver) using 50 ΩODT with a memory-side series resistor transmission line. The FPGA writes to the memory with a 16 mA drive strength setting.

Figure 4–28. Simulation and Board Measurement, FPGA Writing to Memory

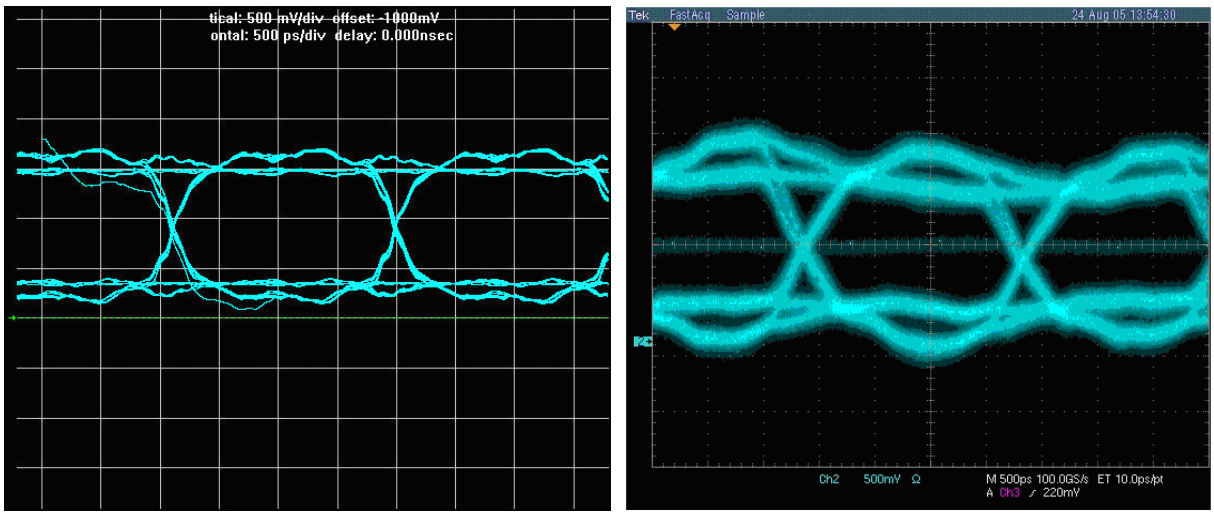


Table 4–11 lists the comparisons of the signal seen the DDR2 SDRAM DIMM of a Class I termination scheme using an external resistor and a Class I termination scheme using ODT with a memory-side series resistor. The FPGA (driver) writes to the memory (receiver).

Table 4–11. Signal Comparison When the FPGA is Writing to Memory (1), (2)

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
Class I Termination Scheme with ODT				
Simulation	1.63	0.84	N/A	0.12
Board Measurement	1.51	0.76	0.05	0.15
Class I Termination Scheme with External Parallel Resistor				
Simulation	1.69	1.51	0.34	0.29
Board Measurement	1.25	1.08	0.41	0.34

Notes to Table 4–11:

(1) The drive strength on the FPGA is set to 16 mA.

(2) N/A is not applicable.

When the ODT feature is enabled in the DDR2 SDRAM, the eye width is improved. There is some degradation to the eye height, but it is not significant. When ODT is enabled, the most significant improvement in signal quality is the reduction of the over- and undershoot, which helps mitigate any potential reliability issues on the memory devices.

Using memory ODT also eliminates the need for external resistors, which reduces board cost and simplifies board routing, allowing you to shrink your boards. Therefore, Altera recommends using the ODT feature on the DDR2 SDRAM memory.

FPGA Reading from Memory

Altera's Arria GX, Arria II GX, Cyclone series, and Stratix II series of devices are not equipped with parallel ODT. When the DDR2 SDRAM ODT feature is turned off when the FPGA is reading from the memory, the termination scheme resembles the no-parallel termination scheme illustrated by Figure 4–31 on page 4–33.

No-Parallel Termination

The no-parallel termination scheme is described in the JEDEC standards JESD8-6 for HSTL I/O, JESD8-9b for SSTL-2 I/O, and JESD8-15a for SSTL-18 I/O. Designers who attempt series-only termination schemes such as this often do so to eliminate the need for a V_{TT} power supply.

This is typically not recommended for any signals between an FPGA and DDR2 interface; however, information about this topic is included here as a reference point to clarify the challenges that may occur if you attempt to avoid parallel termination entirely.

FPGA Writing to Memory

Figure 4–29 shows a no-parallel termination transmission line of the FPGA driving the memory. When the FPGA is driving the transmission line, the signals at the memory-side (DDR2 SDRAM DIMM) may suffer from signal degradation (for example, degradation in rise and fall time). This is due to impedance mismatch, because there is no parallel termination at the memory-side. Also, because of factors such as trace length and drive strength, the degradation seen at the receiver-end might be sufficient to result in a system failure. To understand the effects of each termination scheme on a system, perform system-level simulations before and after the board is designed.

Figure 4–29. No-Parallel Termination Scheme

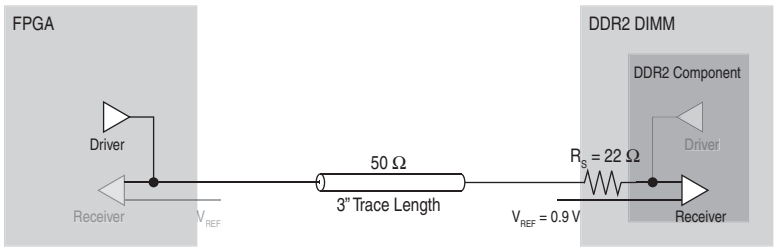
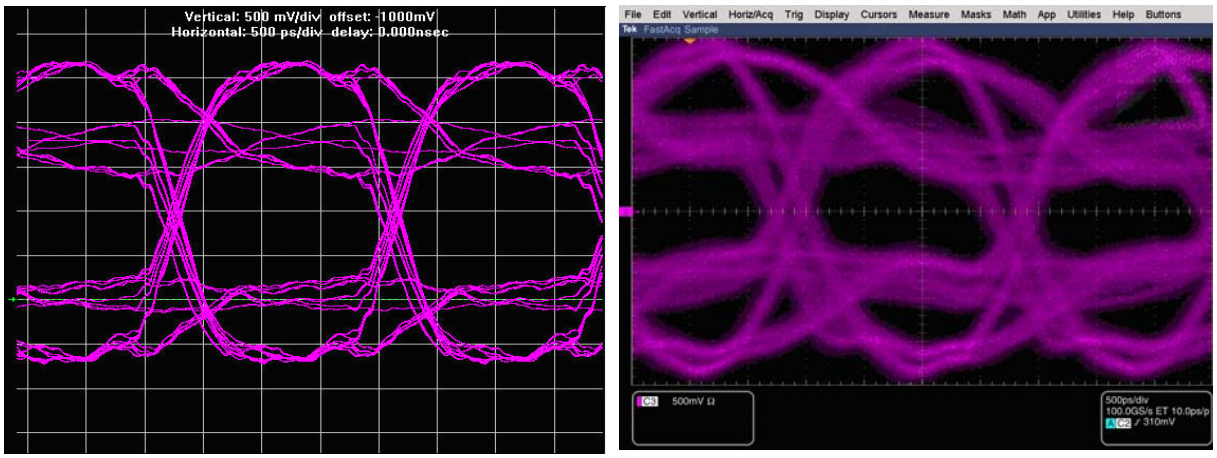


Figure 4–30 shows a HyperLynx simulation and measurement of the FPGA writing to the memory at 533 MHz with a no-parallel termination scheme using a 16 mA drive strength option. The measurement point is on the DDR2 SDRAM DIMM.

Figure 4–30. HyperLynx Simulation and Board Measurement, FPGA Writing to Memory



The simulated and measured signal shows that there is sufficient eye opening but also significant over- and undershoot of the 1.8-V signal specified by the DDR2 SDRAM. From the simulation and measurement, the overshoot is approximately 1 V higher than 1.8 V, and undershoot is approximately 0.8 V below ground. This over- and undershoot might result in a reliability issue, because it has exceeded the absolute maximum rating specification listed in the memory vendors' DDR2 SDRAM data sheet.

Table 4–12 lists the comparison of the signal visible at the DDR2 SDRAM DIMM of a no-parallel and a Class II termination scheme when the FPGA writes to the DDR2 SDRAM DIMM.

Table 4–12. Signal Comparison When the FPGA is Writing to Memory *(1)*

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
No-Parallel Termination Scheme				
Simulation	1.66	1.10	0.90	0.80
Board Measurement	1.25	0.60	1.10	1.08
Class II Termination Scheme With External Parallel Resistor				
Simulation	1.65	1.28	0.16	0.14
Board Measurement	1.35	0.83	0.16	0.18

Note to Table 4–12:

(1) The drive strength on the FPGA is set to Class II 16 mA.

Although the appearance of the signal in a no-parallel termination scheme is not clean, when you take the key parameters into consideration, the eye width and height is comparable to that of a Class II termination scheme. The major disadvantage of using a no-parallel termination scheme is the over- and undershoot. There is no termination on the receiver, so there is an impedance mismatch when the signal arrives at the receiver, resulting in ringing and reflection. In addition, the 16-mA drive strength setting on the FPGA also results in overdriving the transmission line, causing the over- and undershoot. By reducing the drive strength setting, the over- and undershoot decreases and improves the signal quality "seen" by the receiver.

For more information about how drive strength affects the signal quality, refer to "Drive Strength" on page 4–50.

FPGA Reading from Memory

In a no-parallel termination scheme (Figure 4–31), when the memory is driving the transmission line, the resistor, R_S acts as a source termination resistor. The DDR2 SDRAM driver has two drive strength settings:

- Full strength, in which the output impedance is approximately 18Ω
- Reduced strength, in which the output impedance is approximately 40Ω

When the DDR2 SDRAM DIMM drives the transmission line, the combination of the 22-Ω source-series resistor and the driver impedance should match that of the characteristic impedance of the transmission line. As such, there is less over- and undershoot of the signal visible at the receiver (FPGA).

Figure 4–31. No-Parallel Termination Scheme, FPGA Reading from Memory

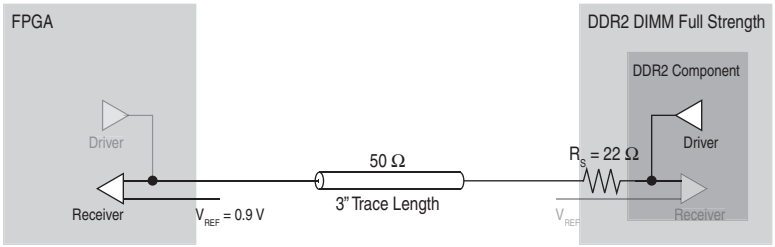


Figure 4–32 shows the simulation and measurement of the signal visible at the FPGA (receiver) when the memory is driving the no-parallel termination transmission line with a memory-side series resistor.

Figure 4–32. HyperLynx Simulation and Board Measurement, FPGA Reading from Memory

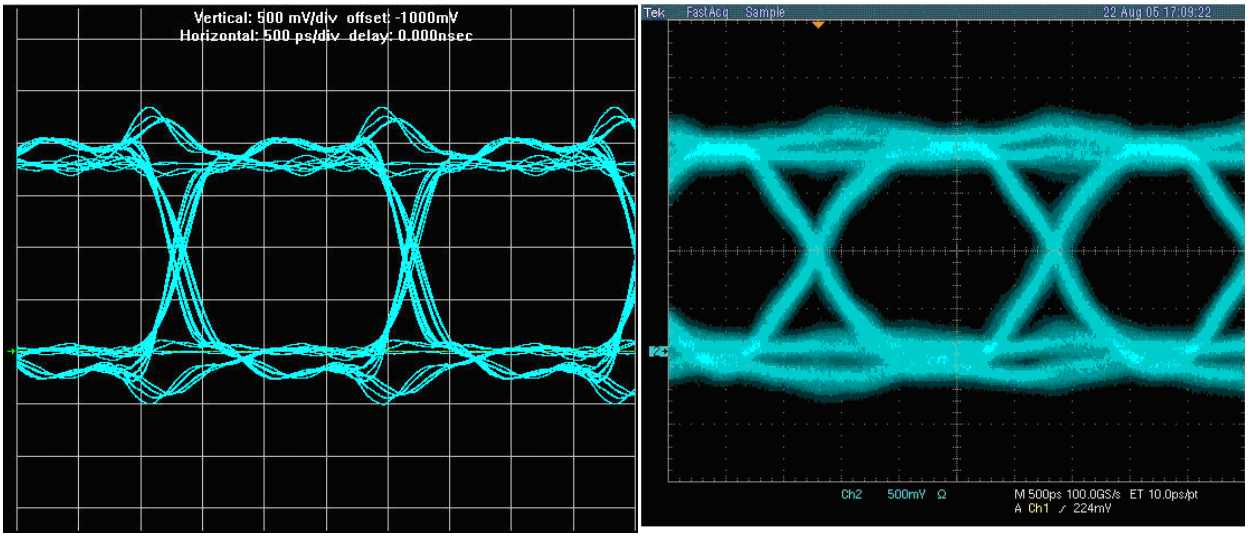


Table 4–13 lists the comparison of the signal seen on the FPGA with a no-parallel and a Class II termination scheme when the FPGA is reading from memory.

Table 4–13. Signal Comparison, FPGA Reading From Memory (1), (2)

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
No-Parallel Termination Scheme				
Simulation	1.82	1.57	0.51	0.51
Board Measurement	1.62	1.29	0.28	0.37
Class II Termination Scheme with External Parallel Resistor				
Simulation	1.73	0.76	N/A	N/A

Table 4–13. Signal Comparison, FPGA Reading From Memory (1), (2)

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
Board Measurement	1.28	0.43	N/A	N/A

Notes to Table 4–13:

(1) The drive strength on the DDR2 SDRAM DIMM is set to full strength.

(2) N/A is not applicable.

As in the section "FPGA Writing to Memory" on page 4–31, the eye width and height of the signal in a no-parallel termination scheme is comparable to a Class II termination scheme, but the disadvantage is the over- and undershoot. There is over- and undershoot because of the lack of termination on the transmission line, but the magnitude of the over- and undershoot is not as severe when compared to that described in "FPGA Writing to Memory" on page 4–31. This is attributed to the presence of the series resistor at the source (memory side), which dampens any reflection coming back to the driver and further reduces the effect of the reflection on the FPGA side.

When the memory-side series resistor is removed (Figure 4–33), the memory driver impedance no longer matches the transmission line and there is no series resistor at the driver to dampen the reflection coming back from the unterminated FPGA side.

Figure 4–33. No-Parallel Termination Scheme, FPGA REading from Memory

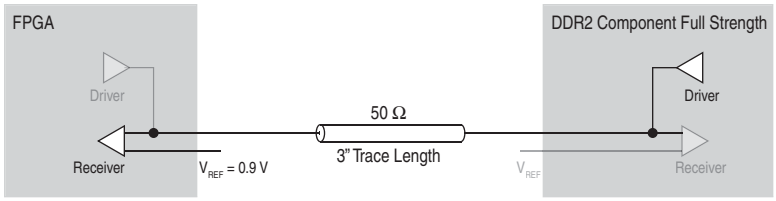


Figure 4–34 shows the simulation and measurement of the signal at the FPGA side in a no-parallel termination scheme with the full drive strength setting on the memory.

Figure 4–34. HyperLynx Simulation and Measurement, FPGA Reading from Memory

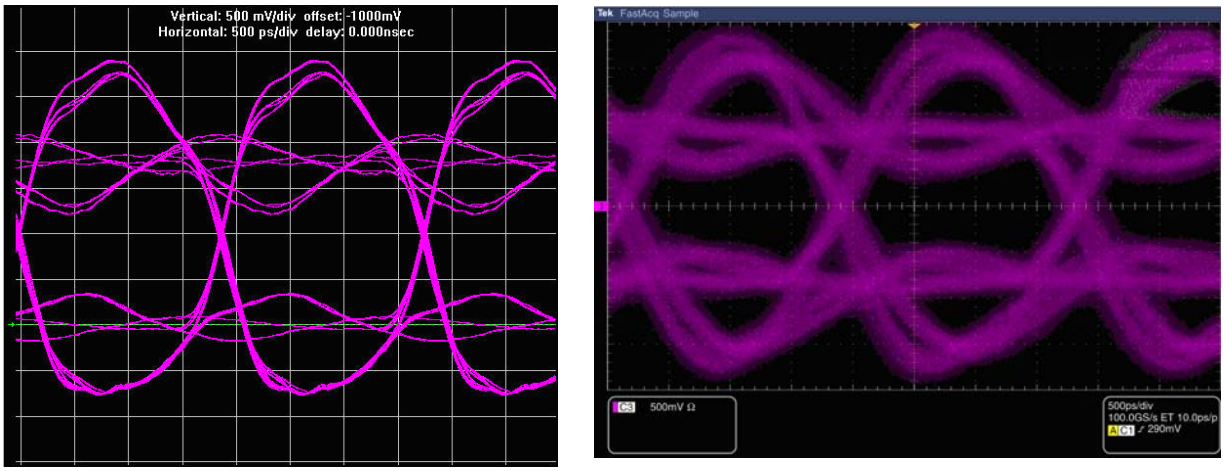


Table 4–14 lists the difference between no-parallel termination with and without memory-side series resistor when the memory (driver) writes to the FPGA (receiver).

Table 4–14. No-Parallel Termination with and without Memory-Side Series Resistor *(1)*

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
Without Series Resistor				
Simulation	1.81	0.85	1.11	0.77
Board Measurement	1.51	0.92	0.96	0.99
With Series Resistor				
Simulation	1.82	1.57	0.51	0.51
Board Measurement	1.62	1.29	0.28	0.37

Note to Table 4–14:

(1) The drive strength on the memory is set to full drive strength.

Table 4–14 highlights the effect of the series resistor on the memory side with the dramatic increase in over- and undershoot and the decrease in the eye height. This result is similar to that described in "FPGA Writing to Memory" on page 4–31. In that simulation, there is a series resistor but it is located at the receiver side (memory-side), so it does not have the desired effect of reducing the drive strength of the driver and suppressing the reflection coming back from the unterminated receiver-end. As such, in a system without receiver-side termination, the series resistor on the driver helps reduce the drive strength of the driver and dampen the reflection coming back from the unterminated receiver-end.

Board Termination for DDR3 SDRAM

The following sections describe the correct way to terminate a DDR3 SDRAM interface together with Stratix III, Stratix IV, and Stratix V FPGA devices.

DDR3 DIMMs have terminations on all unidirectional signals, such as memory clocks, and addresses and commands; thus eliminating the need for them on the FPGA PCB. In addition, using the ODT feature on the DDR3 SDRAM and the dynamic OCT feature of Stratix III, Stratix IV, and Stratix V FPGA devices completely eliminates any external termination resistors; thus simplifying the layout for the DDR3 SDRAM interface when compared to that of the DDR2 SDRAM interface.

This section describes the termination for the following DDR3 SDRAM components:

- Single-Rank DDR3 SDRAM Unbuffered DIMM
- Multi-Rank DDR3 SDRAM Unbuffered DIMM
- DDR3 SDRAM Registered DIMM
- DDR3 SDRAM Components With Leveling

If you are using a DDR3 SDRAM without leveling interface, refer to the "Board Termination for DDR2 SDRAM" on page 4–7.

Single-Rank DDR3 SDRAM Unbuffered DIMM

The most common implementation of the DDR3 SDRAM interface is the unbuffered DIMM (UDIMM). You can find DDR3 SDRAM UDIMMs in many applications, especially in PC applications.

Table 4–15 lists the recommended termination and drive strength setting for UDIMM and Stratix III, Stratix IV, and Stratix V FPGA devices.

These settings are just recommendations for you to get started. Simulate with real board and try different settings to get the best SI.

Table 4–15. Drive Strength and ODT Setting Recommendations for Single-Rank UDIMM

Signal Type	SSTL 15 I/O Standard (1)	FPGA End On-Board Termination (2)	Memory End Termination for Write	Memory Driver Strength for Read
DQ	Class I R50C/G50C (3)	—	60 Ω ODT (4)	40 Ω (4)
DQS	Differential Class I R50C/G50C (3)	—	60 Ω ODT (4)	40 Ω (4)
DM	Class I R50C (3)	—	60 Ω ODT (4)	40 Ω (4)
Address and Command	Class I with maximum drive strength	—	39 Ω on-board termination to V_{TT} (5)	
CK/CK#	Differential Class I R50C	—	On-board (5): 2.2 pf compensation cap before the first component; 36 Ω termination to V_{TT} for each arm (72 Ω differential); add 0.1 uF just before V_{TT} For more information, refer to Figure 4–38 on page 4–39.	

Notes to Table 4–16:

(1) UniPHY IP automatically implements these settings.

(2) Altera recommends that you use dynamic on-chip termination (OCT) for Stratix III and Stratix IV device families.

(3) R50C is series with calibration for write, G50C is parallel 50 with calibration for read.

(4) You can specify these settings in the parameter editor.

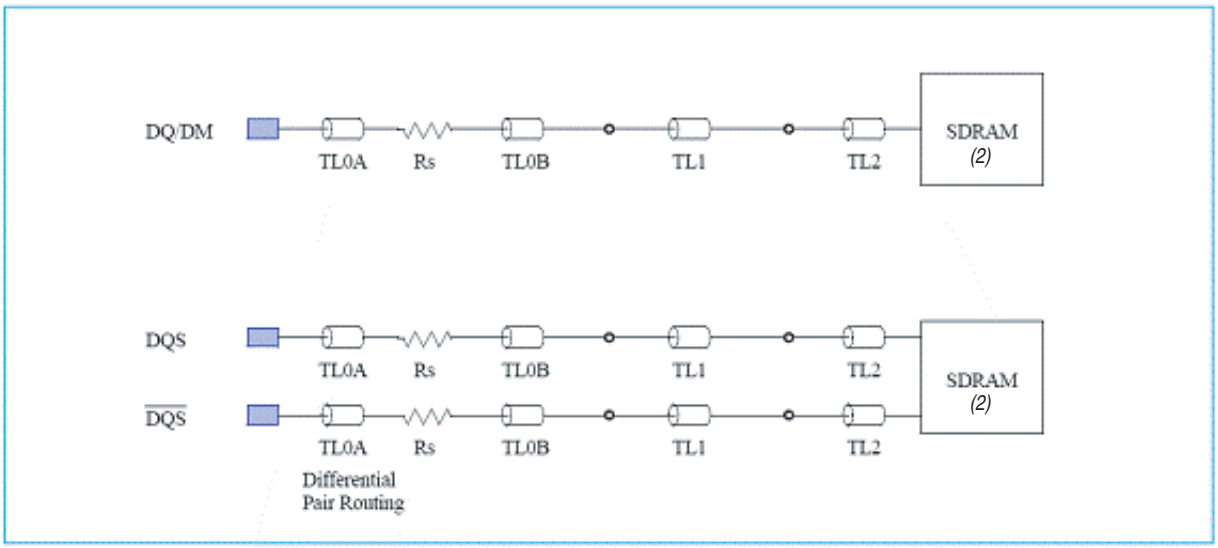
(5) For DIMM, these settings are already implemented on the DIMM card; for component topology, Altera recommends that you mimic termination scheme on the DIMM card on your board.

You can implement a DDR3 SDRAM UDIMM interface in several permutations, such as single DIMM or multiple DIMMs, using either single-ranked or dual-ranked UDIMMs. In addition to the UDIMM's form factor, these termination recommendations are also valid for small-outline (SO) DIMMs and MicroDIMMs.

DQS, DQ, and DM for DDR3 SDRAM UDIMM

On a single-ranked DIMM, DQS, and DQ signals are point-to-point signals. Figure 4–35 shows the net structure for differential DQS and DQ signals. There is an external 15-Ω stub resistor, R_S, on each of the DQS and DQ signals soldered on the DIMM, which helps improve signal quality by dampening reflections from unused slots in a multi-DIMM configuration.

Figure 4–35. DQ and DQS Net Structure for 64-Bit DDR3 SDRAM UDIMM *(1)*



Note to Figure 4–35:

(1) Source: *PC3-6400/PC3-8500/PC3-10600/PC3-12800 DDR3 SDRAM Unbuffered DIMM Design Specification*, July 2007, JEDEC Solid State Technology Association. For clarity of the signal connections in the illustration, the same SDRAM is drawn as two separate SDRAMs.

As mentioned in "Dynamic ODT" on page 4–5, DDR3 SDRAM supports calibrated ODT with different ODT value settings. If you do not enable dynamic ODT, there are three possible ODT settings available for `RTT_NORM`: 40 Ω, 60 Ω, and 120 Ω. If you enable dynamic ODT, the number of possible ODT settings available for `RTT_NORM` increases from three to five with the addition of 20 Ω and 30 Ω. Trace impedance on the DIMM and the recommended ODT setting is 60 Ω.

Figure 4–36 shows the simulated write-eye diagram at the DQ0 of a DDR3 SDRAM DIMM using the 60-Ω ODT setting, driven by a Stratix III or Stratix IV FPGA using a calibrated series 50-Ω OCT setting.

Figure 4–36. Simulated Write-Eye Diagram of a DDR3 SDRAM DIMM Using a 60-Ω ODT Setting

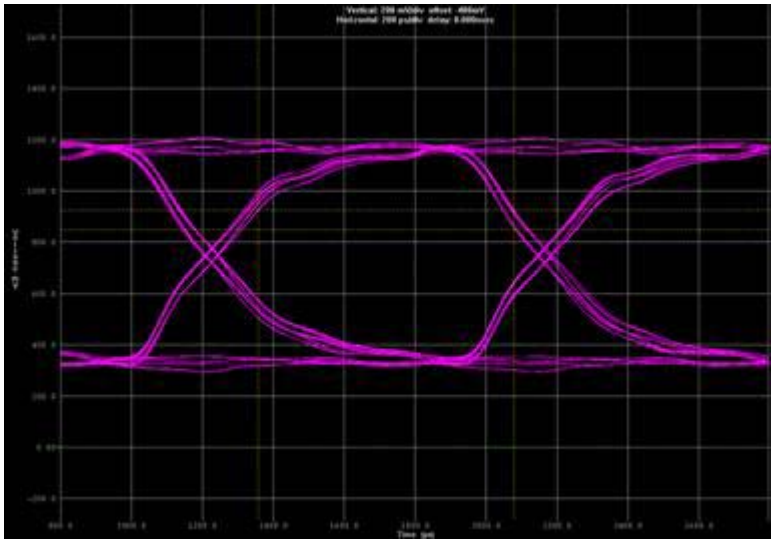
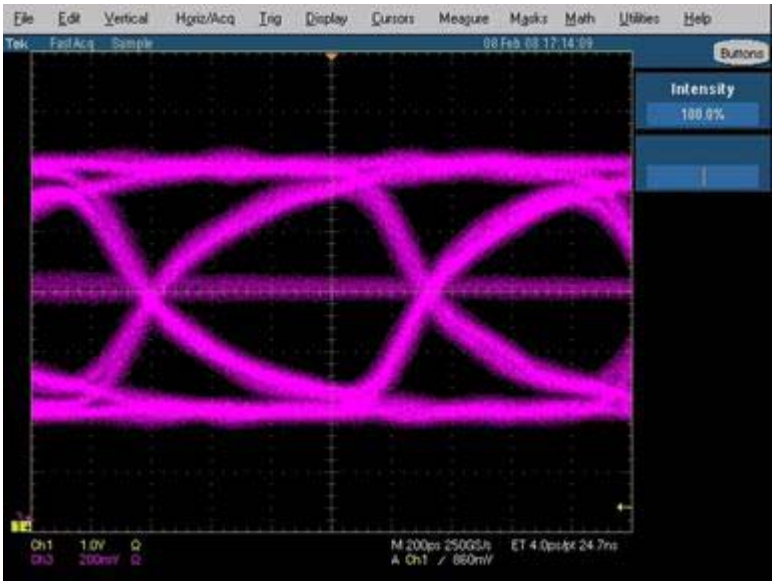


Figure 4–37 shows the measured write eye diagram using Altera's Stratix III or Stratix IV memory board.

Figure 4–37. Measured Write-Eye Diagram of a DDR3 SDRAM DIMM Using the 60-Ω ODT Setting

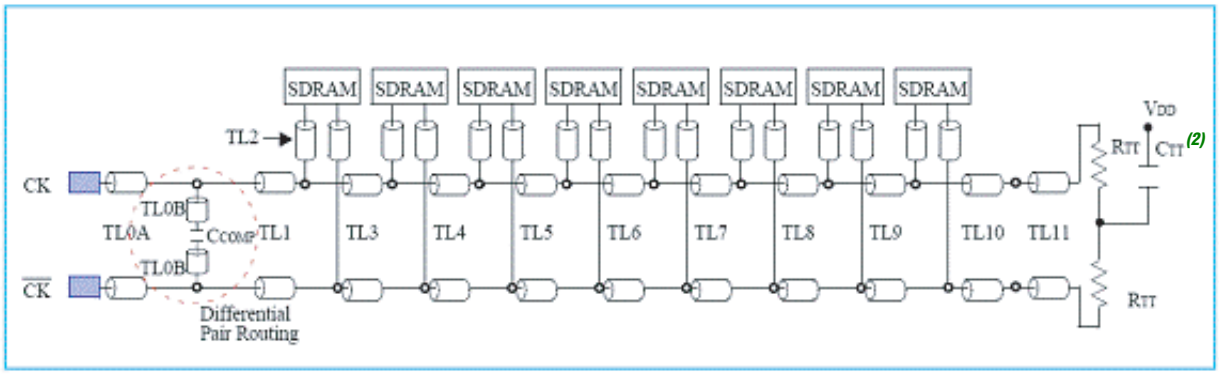


The measured eye diagram correlates well with the simulation. The faint line in the middle of the eye diagram is the effect of the refresh operation during a regular operation. Because these simulations and measurements are based on a narrow set of constraints, you must perform your own board-level simulation to ensure that the chosen ODT setting is right for your setup.

Memory Clocks for DDR3 SDRAM UDIMM

For the DDR3 SDRAM UDIMM, you do not need to place any termination on your board because the memory clocks are already terminated on the DIMM. Figure 4–38 shows the net structure for the memory clocks and the location of the termination resistors, R_{TT}. The value of R_{TT} is 36 Ω, which results in an equivalent differential termination value of 72 Ω. The DDR3 SDRAM DIMM also has a compensation capacitor, C_{COMP} of 2.2 pF, placed between the differential memory clocks to improve signal quality. The recommended center-tap-terminated (C_{TT}) value is 0.1 uF just before V_{TT}.

Figure 4–38. Clock Net Structure for a 64-Bit DDR3 SDRAM UDIMM *(1)*



Note to Figure 4–38:

(1) Source: *PC3-6400/PC3-8500/PC3-10600/PC3-12800 DDR3 SDRAM Unbuffered DIMM Design Specification*, July 2007, JEDEC Solid State Technology Association.

(2) The recommeded C_{TT} value is 0.1 uF just before V_{TT}..

From Figure 4–38, you can see that the DDR3 SDRAM clocks are routed in a fly-by topology, as mentioned in "Read and Write Leveling" on page 4–3, resulting in the need for write-and-read leveling. Figure 4–39 shows the HyperLynx simulation of the differential clock seen at the die of the first and last DDR3 SDRAM component on the UDIMM using the 50-Ω OCT setting on the output driver of the Stratix III or Stratix IV FPGA.

Figure 4–39. Differential Memory Clock of a DDR3 SDRAM DIMM at the First and Last Component on the DIMM

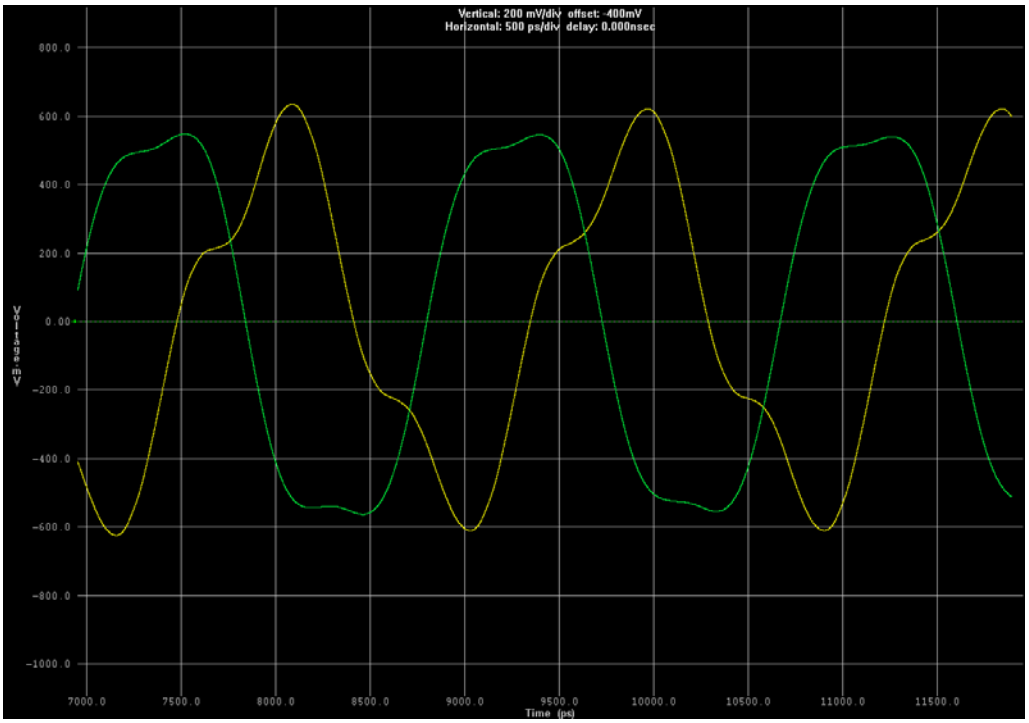
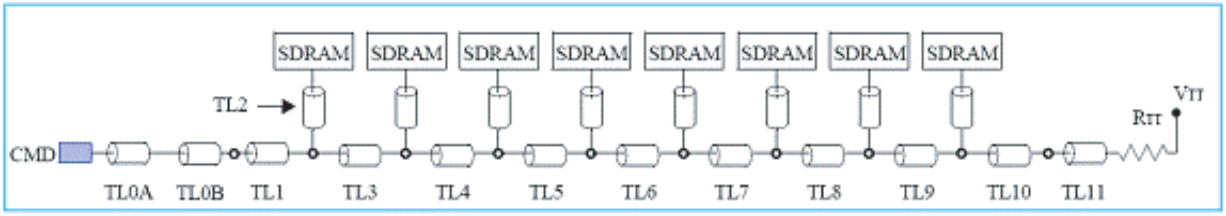


Figure 4–39 shows that the memory clock seen at the first DDR3 SDRAM component (the yellow signal) leads the memory clock seen at the last DDR3 SDRAM component (the green signal) by 1.3 ns, which is about 0.69 t_{CK} for a 533 MHz operation.

Commands and Addresses for DDR3 SDRAM UDIMM

Similar to memory clock signals, you do not need to place any termination on your board because the command and address signals are also terminated on the DIMM. Figure 4–40 shows the net structure for the command and address signals, and the location of the termination resistor, R_{TT}, which has an R_{TT} value of 39 Ω

Figure 4–40. Command and Address Net Structure for a 64-Bit DDR3 SDRAM Unbuffered DIMM *(1)*



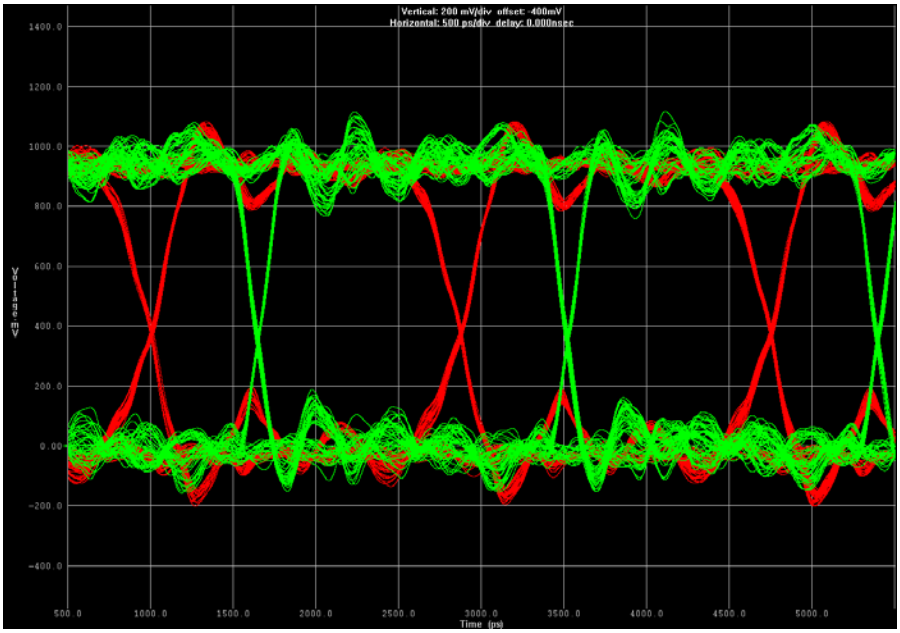
Note to Figure 4–40:

(1) Source: *PC3-6400/PC3-8500/PC3-10600/PC3-12800 DDR3 SDRAM Unbuffered DIMM Design Specification*, July 2007, JEDEC Solid State Technology Association

In Figure 4–40, observe that the DDR3 SDRAM command and address signals are routed in a fly-by topology, as mentioned in "Read and Write Leveling" on page 4–3, resulting in the need for write-and-read leveling.

Figure 4–41 shows the HyperLynx simulation of the command and address signal seen at the die of the first and last DDR3 SDRAM component on the UDIMM, using an OCT setting on the output driver of the Stratix III or Stratix IV FPGA.

Figure 4–41. Command and Address Eye Diagram of a DDR3 SDRAM DIMM at the First and Last DDR3 SDRAM Component at 533 MHz *(1)*



Note to Figure 4–41:

(1) The command and address simulation is performed using a bit period of 1.875 ns.

Figure 4–41 shows that the command and address signal seen at the first DDR3 SDRAM component (the green signal) leads the command and address signals seen at the last DDR3 SDRAM component (the red signal) by 1.2 ns, which is 0.64 t_{CK} for a 533-MHz operation.

Stratix III, Stratix IV, and Stratix V FPGAs

The following sections review the termination on the single-ranked single DDR3 SDRAM DIMM interface side and investigate the use of different termination features available in Stratix III, Stratix IV, and Stratix V FPGA devices to achieve optimum signal integrity for your DDR3 SDRAM interface.

DQS, DQ, and DM for Stratix III, Stratix IV, and Stratix V FPGA

As mentioned in "Dynamic OCT in Stratix III and Stratix IV Devices" on page 4–5, Stratix III, Stratix IV, and Stratix V FPGAs support the dynamic OCT feature, which switches from series termination to parallel termination depending on the mode of the I/O buffer. Because DQS and DQ are bidirectional signals, DQS and DQ can be both transmitters and receivers. "DQS, DQ, and DM for DDR3 SDRAM UDIMM" on page 4–37 describes the signal quality of DQ, DQS, and DM when the Stratix III, Stratix IV, or Stratix V FPGA device is the transmitter with the I/O buffer set to a 50-Ω series termination.

This section details the condition when the Stratix III, Stratix IV, or Stratix V device is the receiver, the Stratix III, Stratix IV, and Stratix V I/O buffer is set to a 50-Ω parallel termination, and the memory is the transmitter. DM is a unidirectional signal, so the DDR3 SDRAM component is always the receiver.

For receiver termination recommendations and transmitter output drive strength settings, refer to "DQS, DQ, and DM for DDR3 SDRAM UDIMM" on page 4–37.

Figure 4–42 illustrates the DDR3 SDRAM interface when the Stratix III, Stratix IV, or Stratix V FPGA device is reading from the DDR3 SDRAM using a 50-Ω parallel OCT termination on the Stratix III, Stratix IV, or Stratix V FPGA device, and the DDR3 SDRAM driver output impedance is set to 34 Ω.

Figure 4–42. DDR3 SDRAM Component Driving the Stratix III, Stratix IV, and Stratix V FPGA Device with Parallel 50-Ω OCT Turned On

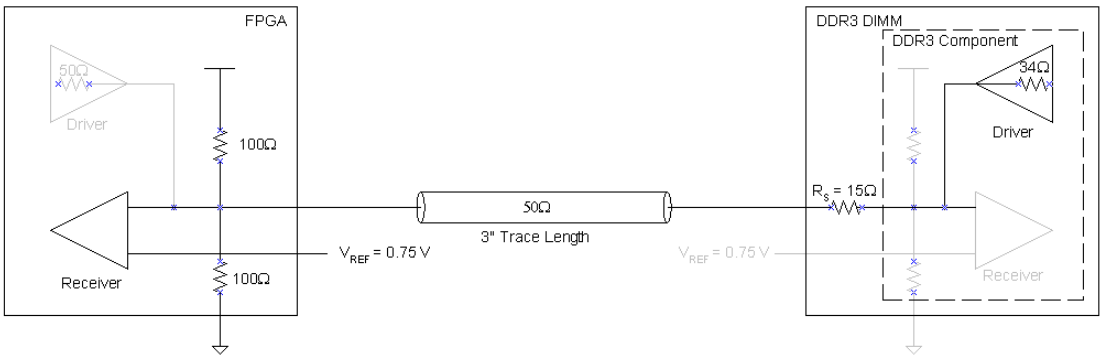
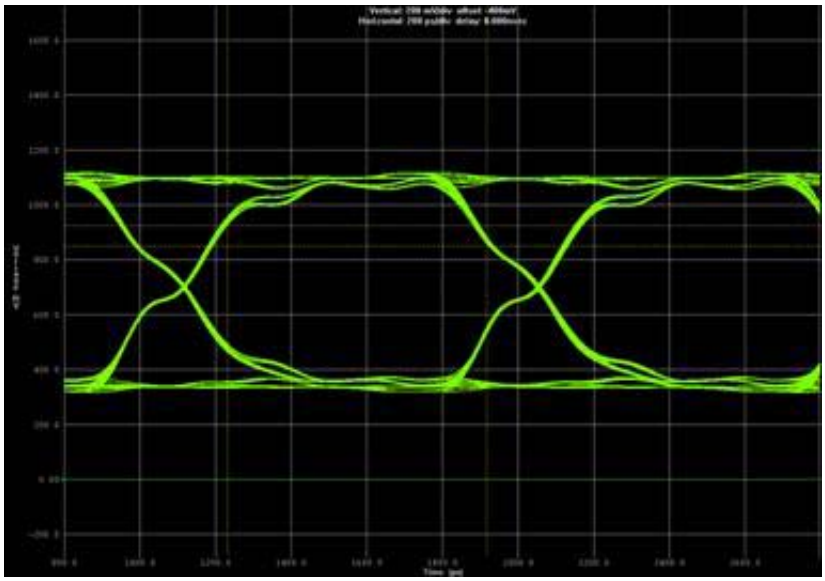


Figure 4–43 shows the simulation of a read from the DDR3 SDRAM DIMM with a 50-Ω parallel OCT setting on the Stratix III or Stratix IV FPGA device.

Figure 4–43. Read-Eye Diagram of a DDR3 SDRAM DIMM at the Stratix III and Stratix IV FPGA Using a Parallel 50-Ω OCT Setting



Use of the Stratix III, Stratix IV, or Stratix V parallel 50-Ω OCT feature matches receiver impedance with the transmission line characteristic impedance. This eliminates any reflection that causes ringing, and results in a clean eye diagram at the Stratix III, Stratix IV, or Stratix V FPGA.

Memory Clocks for Stratix III, Stratix IV, and Stratix V FPGA

Memory clocks are unidirectional signals. Refer to "Memory Clocks for DDR3 SDRAM UDIMM" on page 4–39 for receiver termination recommendations and transmitter output drive strength settings.

Commands and Addresses for Stratix III and Stratix IV FPGA

Commands and addresses are unidirectional signals. Refer to "Commands and Addresses for DDR3 SDRAM UDIMM" on page 4–41 for receiver termination recommendations and transmitter output drive strength settings.

Multi-Rank DDR3 SDRAM Unbuffered DIMM

You can implement a DDR3 SDRAM UDIMM interface in several permutations, such as single DIMM or multiple DIMMs, using either single-ranked or dual-ranked UDIMMs. In addition to the UDIMM's form factor, these termination recommendations are also valid for small-outline (SO) DIMMs and MicroDIMMs.

Table 4–16 lists the different permutations of a two-slot DDR3 SDRAM interface and the recommended ODT settings on both the memory and controller when writing to memory.

Table 4–16. DDR3 SDRAM ODT Matrix for Writes (1) and (2)

Slot 1	Slot 2	Write To	Controller OCT (3)	Slot 1		Slot 2	
				Rank 1	Rank 2	Rank 1	Rank 2
DR	DR	Slot 1	Series 50 Ω	120 Ω (4)	ODT off	ODT off	40 Ω (4)
		Slot 2	Series 50 Ω	ODT off	40 Ω (4)	120 Ω (4)	ODT off
SR	SR	Slot 1	Series 50 Ω	120 Ω (4)	Unpopulated	40 Ω (4)	Unpopulated
		Slot 2	Series 50 Ω	40 Ω (4)	Unpopulated	120 Ω (4)	Unpopulated
DR	Empty	Slot 1	Series 50 Ω	120 Ω	ODT off	Unpopulated	Unpopulated
Empty	DR	Slot 2	Series 50 Ω	Unpopulated	Unpopulated	120 Ω	ODT off
SR	Empty	Slot 1	Series 50 Ω	120 Ω	Unpopulated	Unpopulated	Unpopulated
Empty	SR	Slot 2	Series 50 Ω	Unpopulated	Unpopulated	120 Ω	Unpopulated

Notes to Table 4–16:

(1) SR: single-ranked DIMM; DR: dual-ranked DIMM.

(2) These recommendations are taken from the *DDR3 ODT and Dynamic ODT* session of the JEDEC DDR3 2007 Conference, Oct 3-4, San Jose, CA.

(3) The controller in this case is the FPGA.

(4) Dynamic ODT is required. For example, the ODT of Slot 2 is set to the lower ODT value of 40 Ω when the memory controller is writing to Slot 1, resulting in termination and thus minimizing any reflection from Slot 2. Without dynamic ODT, Slot 2 will not be terminated.

Table 4–17 lists the different permutations of a two-slot DDR3 SDRAM interface and the recommended ODT settings on both the memory and controller when reading from memory.

Table 4–17. DDR3 SDRAM ODT Matrix for Reads (1) and (2)

Slot 1	Slot 2	Read From	Controller OCT (3)	Slot 1		Slot 2	
				Rank 1	Rank 2	Rank 1	Rank 2
DR	DR	Slot 1	Parallel 50 Ω	ODT off	ODT off	ODT off	40 Ω
		Slot 2	Parallel 50 Ω	ODT off	40 Ω	ODT off	ODT off
SR	SR	Slot 1	Parallel 50 Ω	ODT off	Unpopulated	40 Ω	Unpopulated
		Slot 2	Parallel 50 Ω	40 Ω	Unpopulated	ODT off	Unpopulated
DR	Empty	Slot 1	Parallel 50 Ω	ODT off	ODT off	Unpopulated	Unpopulated
Empty	DR	Slot 2	Parallel 50 Ω	Unpopulated	Unpopulated	ODT off	ODT off
SR	Empty	Slot 1	Parallel 50 Ω	ODT off	Unpopulated	Unpopulated	Unpopulated
Empty	SR	Slot 2	Parallel 50 Ω	Unpopulated	Unpopulated	ODT off	Unpopulated

Notes to Table 4–17:

(1) SR: single-ranked DIMM; DR: dual-ranked DIMM.

(2) These recommendations are taken from the *DDR3 ODT and Dynamic ODT* session of the JEDEC DDR3 2007 Conference, Oct 3-4, San Jose, CA.

(3) The controller in this case is the FPGA. JEDEC typically recommends 60 Ω but this value assumes that the typical motherboard trace impedance is 60 Ω and that the controller supports this termination. Altera recommends using a 50-Ω parallel OCT when reading from the memory.

DDR3 SDRAM Registered DIMM

The difference between a registered DIMM (RDIMM) and a UDIMM is that the clock, address, and command pins of the RDIMM are registered or buffered on the DIMM before they are distributed to the memory devices. For a controller, each clock, address, or command signal has only one load, which is the register or buffer. In a UDIMM, each controller pin must drive a fly-by wire with multiple loads.

You do not need to terminate the clock, address, and command signals on your board because these signals are terminated at the register. However, because of the register, these signals become point-to-point signals and have improved signal integrity making the drive strength requirements of the FPGA driver pins more relaxed. Similar to the signals in a UDIMM, the DQS, DQ, and DM signals on a RDIMM are not registered. To terminate these signals, refer to "DQS, DQ, and DM for DDR3 SDRAM UDIMM" on page 4–37.

DDR3 SDRAM Load-Reduced DIMM

RDIMM and LRDIMM differ in that DQ, DQS, and DM signals are registered or buffered in the LRDIMM. The LRDIMM buffer IC is a superset of the RDIMM buffer IC. The buffer IC isolates the memory interface signals from loading effects of the memory chip. Reduced electrical loading allows a system to operate at higher frequency and higher density.

If you want to use your DIMM socket for UDIMM and RDIMM/LRDIMM, you must create the necessary redundant connections on the board from the FPGA to the DIMM socket. For example, the number of chip select signals required for a single-rank UDIMM is one, but for single-rank RDIMM the number of chip selects required is two. RDIMM and LRDIMM have parity signals associated with the address and command bus which UDIMM does not have. Consult the DIMM manufacturer's data sheet for detailed information about the necessary pin connections for various DIMM topologies.

DDR3 SDRAM Components With Leveling

This section discusses terminations used to achieve optimum performance for designing the DDR3 SDRAM interface using discrete DDR3 SDRAM components.

In addition to using DDR3 SDRAM DIMM to implement your DDR3 SDRAM interface, you can also use DDR3 SDRAM components. However, for applications that have limited board real estate, using DDR3 SDRAM components reduces the need for a DIMM connector and places components closer, resulting in denser layouts.

DDR3 SDRAM Components With or Without Leveling

The DDR3 SDRAM UDIMM is laid out to the JEDEC specification. The JEDEC specification is available from either the JEDEC Organization website (www.JEDEC.org) or from the memory vendors. However, when you are designing the DDR3 SDRAM interface using discrete SDRAM components, you may desire a layout scheme that is different than the DIMM specification. You have the following two options:

- Mimic the standard DDR3 SDRAM DIMM, using a fly-by topology for the memory clocks, address, and command signals. This options needs read and write leveling, so you must use the UniPHY IP with leveling.

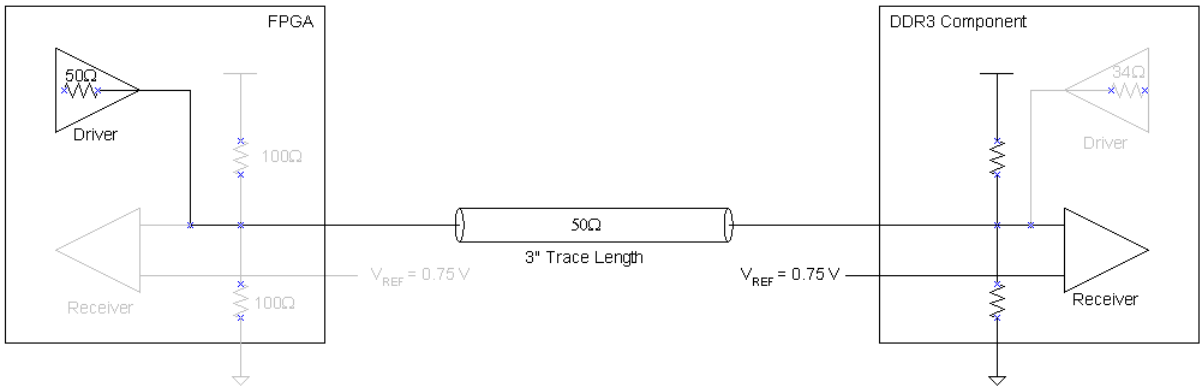
  For more information about this fly-by configuration, continue reading this chapter.

- Mimic a standard DDR2 SDRAM DIMM, using a balanced (symmetrical) tree-type topology for the memory clocks, address, and command signals. Using this topology results in unwanted stubs on the command, address, and clock, which degrades signal integrity and limits the performance of the DDR3 SDRAM interface.

DQS, DQ, and DM for DDR3 SDRAM Components

When you are laying out the DDR3 SDRAM interface using Stratix III, Stratix IV, or Stratix V devices, Altera recommends that you not include the 15-Ω stub series resistor that is on every DQS, DQ, and DM signal; unless your simulation shows that the absence of this resistor causes extra reflection. Although adding the 15-Ω stub series resistor may help to maintain constant impedance in some cases, it also slightly reduces signal swing at the receiver. It is unlikely that by removing this resistor the waveform shows a noticeable reflection, but it is your responsibility to prove by simulating your board trace. Therefore, Altera recommends the DQS, DQ, and DM topology shown in Figure 4–44 when the Stratix III, Stratix IV, or Stratix V FPGA is writing to the DDR3 SDRAM.

Figure 4–44. Stratix III, Stratix IV, and Stratix V FPGA Writing to a DDR3 SDRAM Components

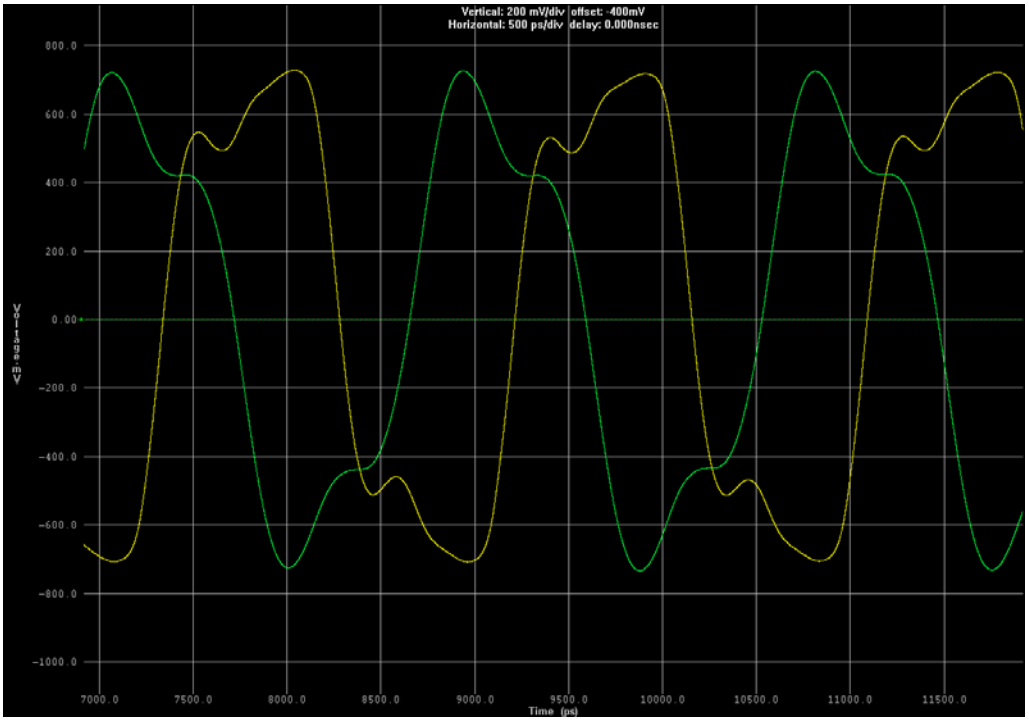


When you are using DDR3 SDRAM components, there are no DIMM connectors. This minimizes any impedance discontinuity, resulting in better signal integrity.

Memory Clocks for DDR3 SDRAM Components

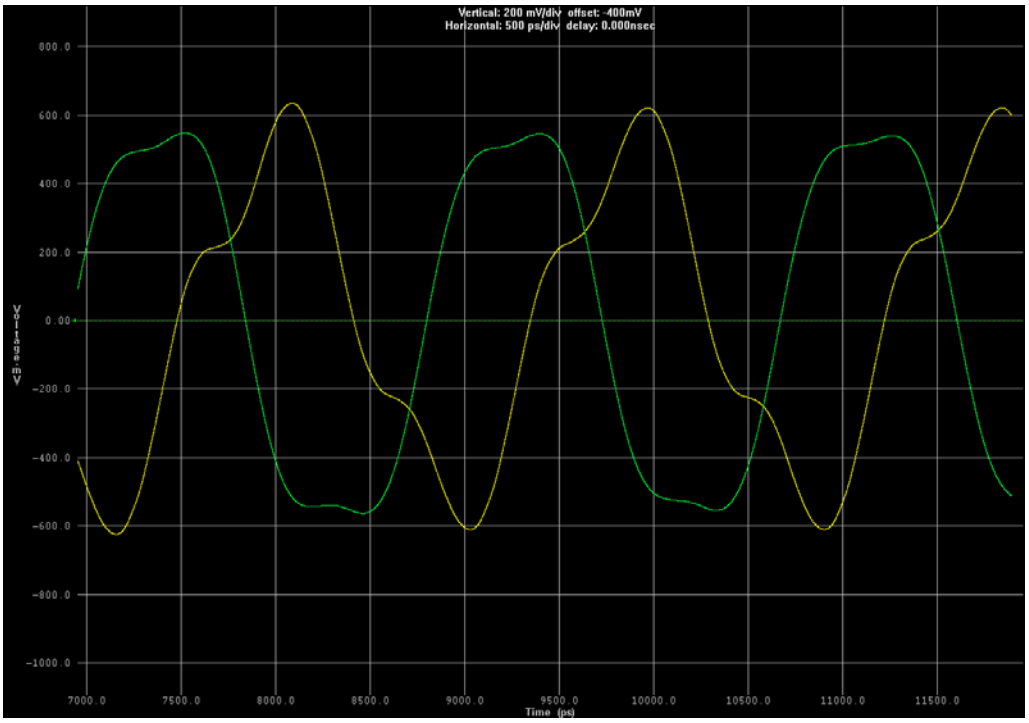
When you use DDR3 SDRAM components, you must account for the compensation capacitor and differential termination resistor between the differential memory clocks of the DIMM. Figure 4–45 shows the HyperLynx simulation of the differential clock seen at the die of the first and last DDR3 SDRAM component using a fly-by topology on a board, without the 2.2 pF compensation capacitor using the 50-ΩOCT setting on the output driver of the Stratix III, Stratix IV, or Stratix V FPGA.

Figure 4–45. Differential Memory Clock of a DDR3 SDRAM Component without the Compensation Capacitor at the First and Last Component Using a Fly-by Topology on a Board



Without the compensation capacitor, the memory clocks (the yellow signal) at the first component have significant ringing, whereas, with the compensation capacitor the ringing is dampened. Similarly, the differential termination resistor needs to be included in the design. Depending on your board stackup and layout requirements, you choose your differential termination resistor value. Figure 4–46 shows the HyperLynx simulation of the differential clock seen at the die of the first and last DDR3 SDRAM component using a fly-by topology on a board, and terminated with 100 Ω instead of the 72 Ω used in the DIMM.

Figure 4–46. Differential Memory Clock of a DDR3 SDRAM DIMM Terminated with 100 Ω at the First and Last Component Using a Fly-by Topology on a Board



Terminating with 100 Ω instead of 72 Ω results in a slight reduction in peak-to-peak amplitude. To simplify your design, use the terminations outlined in the JEDEC specification for DDR3 SDRAM UDIMM as your guide and perform simulation to ensure that the DDR3 SDRAM UDIMM terminations provide you with optimum signal quality.

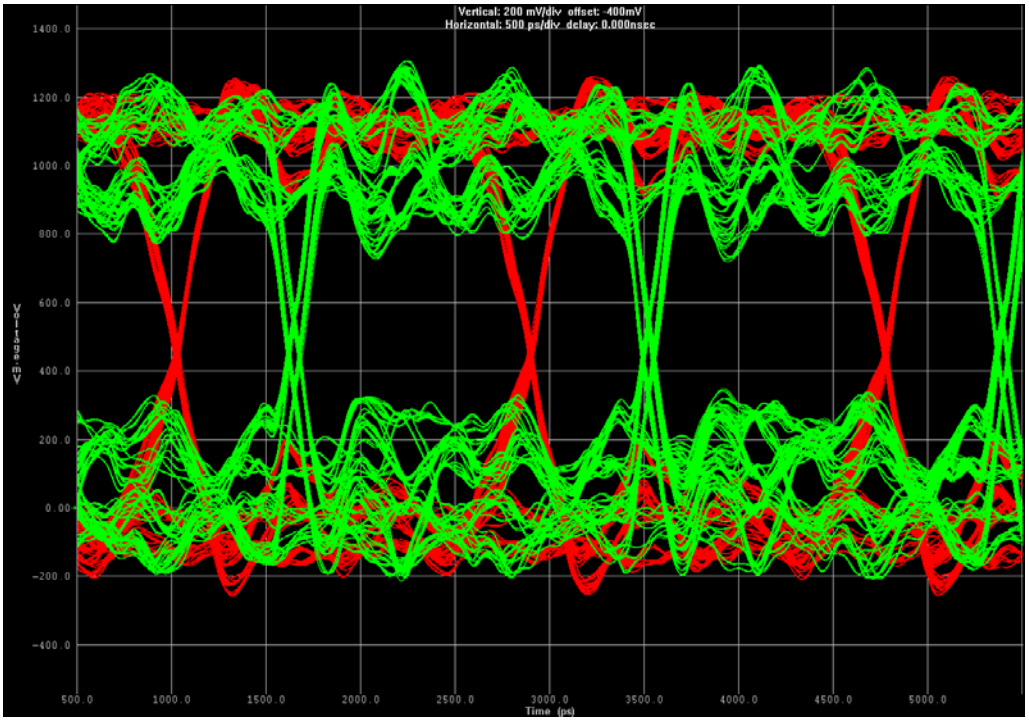
In addition to choosing the value of the differential termination, you must consider the trace length of the memory clocks. Altera's DDR3 UniPHY IP currently supports a flight-time skew of no more than 0.69 t_{CK} in between the first and last memory component. If you use Altera's DDR3 UniPHY IP to create your DDR3 SDRAM interface, ensure that the flight-time skew of your memory clocks is not more than 0.69 t_{CK}. UniPHY IP also requires that the total skew combination of the clock fly-by skew and DQS skew is less than 1 clock cycle.

Refer to "Layout Guidelines for DDR3 SDRAM Interface" on page 4–61 for more information about layout guidelines for DDR3 SDRAM components.

Command and Address Signals for DDR3 SDRAM

As with memory clock signals, you must account for the termination resistor on the command and address signals when you use DDR3 SDRAM components. Choose your termination resistor value depending on your board stackup and layout requirements. Figure 4–47 shows the HyperLynx simulation of the command and address seen at the die of the first and last DDR3 SDRAM component using a fly-by topology on a board terminated with 60 Ω instead of the 39 Ω used in the DIMM.

Figure 4–47. Command and Address Eye Diagram of a DDR3 SDRAM Component Using Fly-by Topology on a Board at the First and Last DDR3 SDRAM Component at 533 MHz, Terminated with 60 Ω



Terminating with 60 Ω instead of 39 Ω results in eye closure in the signal at the first component (the green signal), while there is no effect on the signal at the last component (the red signal). To simplify your design with discrete DDR3 SDRAM components, use the terminations outlined in the JEDEC specification for DDR3 SDRAM UDIMM as your guide, and perform simulation to ensure that the DDR3 SDRAM UDIMM terminations provide you with the optimum signal quality.

As with memory clocks, you must consider the trace length of the command and address signals so that they match the flight-time skew of the memory clocks.

Stratix III, Stratix IV, and Stratix V FPGAs

Stratix III, Stratix IV, or Stratix V FPGA termination settings for DIMM also applies to DDR3 SDRAM component interfaces.

Table 4–18 compares the effects of the series stub resistor on the eye diagram at the Stratix III or Stratix IV FPGA (receiver) when the Stratix III or Stratix IV FPGA is reading from the memory.

Table 4–18. Read-Eye Diagram with and without R_S Using 50-Ω Parallel OCT

ODT	Eye Height (V)	Eye Width (ps)	Overshoot (V)	Undershoot (V)
With R_S	0.70	685	—	—
Without R_S	0.73	724	—	—

Without the 15-Ω stub series resistor to dampen the signal, the signal at the receiver of the Stratix III or Stratix IV FPGA driven by the DDR3 SDRAM component is larger than the signal at the receiver of the Stratix III or Stratix IV FPGA driven by DDR3 SDRAM DIMM (Figure 4–42), and similar to the write-eye diagram in "DQS, DQ, and DM for DDR3 SDRAM Components" on page 4–46.

Drive Strength

Altera's FPGA products offer numerous drive strength settings, allowing you to optimize your board designs to achieve the best signal quality. This section focuses on the most commonly used drive strength settings of 8 mA and 16 mA, as recommended by JEDEC for Class I and Class II termination schemes.

You are not restricted to using only these drive strength settings for your board designs. You should perform simulations using I/O models available from Altera and memory vendors to ensure that you use the proper drive strength setting to achieve optimum signal integrity.

How Strong is Strong Enough?

Figure 4–19 on page 4–22 shows a signal probed at the DDR2 SDRAM DIMM (receiver) of a far-end series-terminated transmission line when the FPGA writes to the DDR2 SDRAM DIMM using a drive strength setting of 16 mA. The resulting signal quality on the receiver shows excessive over- and undershoot. To reduce the over- and undershoot, you can reduce the drive strength setting on the FPGA from 16 mA to 8 mA. Figure 4–48 shows the simulation and measurement of the FPGA with a drive strength setting of 8 mA driving a no-parallel termination transmission line.

Figure 4–48. HyperLynx Simulation and Measurement, FPGA Writing to Memory

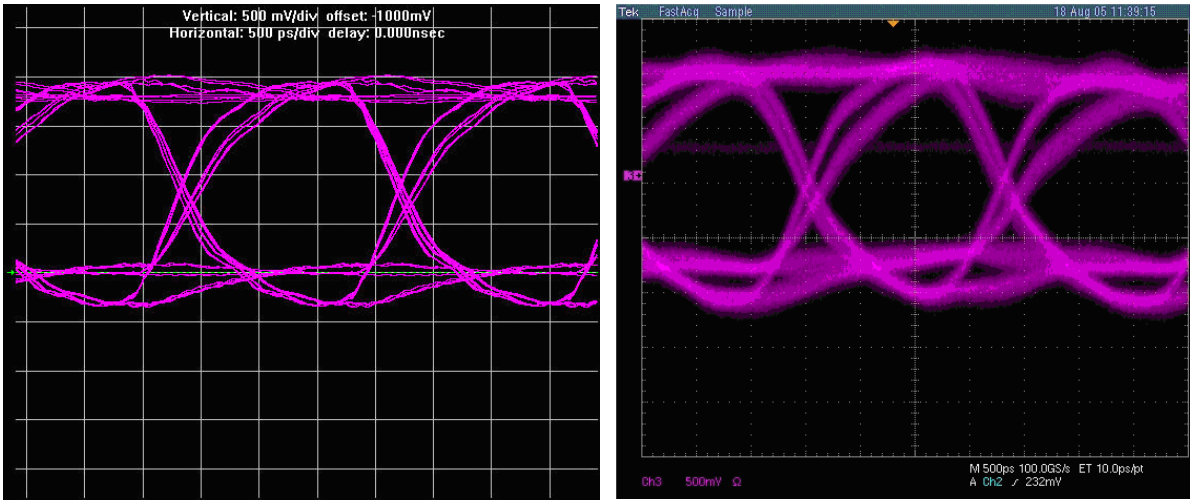


Table 4–19 compares the signals at the DDR2 SDRAM DIMM with no-parallel termination and memory-side series resistors when the FPGA is writing to the memory with 8-mA and 16-mA drive strength settings.

Table 4–19. Simulation and Board Measurement Results for 8 mA and 16 mA Drive Strength Settings

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)
8-mA Drive Strength Setting				
Simulation	1.48	1.71	0.24	0.35
Board Measurement	1.10	1.24	0.24	0.50
16-mA Drive Strength Setting				
Simulation	1.66	1.10	0.90	0.80
Board Measurements	1.25	0.60	1.10	1.08

With a lower strength drive setting, the overall signal quality is improved. The eye width is reduced, but the eye height is significantly larger with a lower drive strength and the over- and undershoot is reduced dramatically.

To improve the signal quality further, you should use 50-Ω on-chip series termination in place of an 8mA drive strength and 25-Ω on-chip series termination in place of a 16 mA drive strength. Refer to "On-Chip Termination (Non-Dynamic)" on page 4–17 for simulation and board measurements.

The drive strength setting is highly dependent on the termination scheme, so it is critical that you perform pre- and post-layout board-level simulations to determine the proper drive strength settings.

System Loading

You can use memory in a variety of forms, such as individual components or multiple DIMMs, resulting in different loading seen by the FPGA. This section describes the effect on signal quality when interfacing memories in component, dual rank, and dual DIMMs format.

Component Versus DIMM

When using discrete DDR2 SDRAM components, the additional loading from the DDR2 SDRAM DIMM connector is eliminated and the memory-side series resistor on the DDR2 SDRAM DIMM is no longer there. You must decide if the memory-side series resistor near the DDR2 SDRAM is required.

FPGA Writing to Memory

Figure 4–49 shows the Class II termination scheme without the memory-side series resistor when the FPGA is writing to the memory in the component format.

Figure 4–49. Class II Termination Scheme without Memory-Side Series Resistor

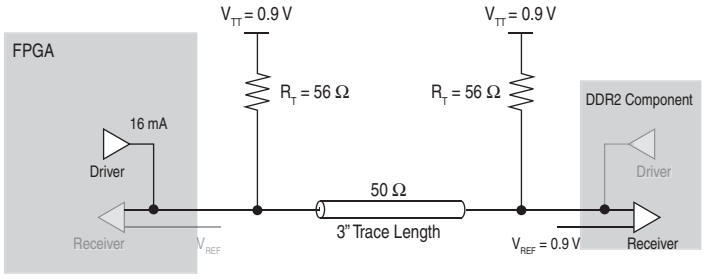


Figure 4–50 shows the simulation and measurement results of the signal seen at a DDR2 SDRAM component of a Class II termination scheme without the DIMM connector and the memory-side series resistor. The FPGA is writing to the memory with a 16-mA drive strength setting.

Figure 4–50. HyperLynx Simulation and Measurement of the Signal, FPGA Writing to Memory

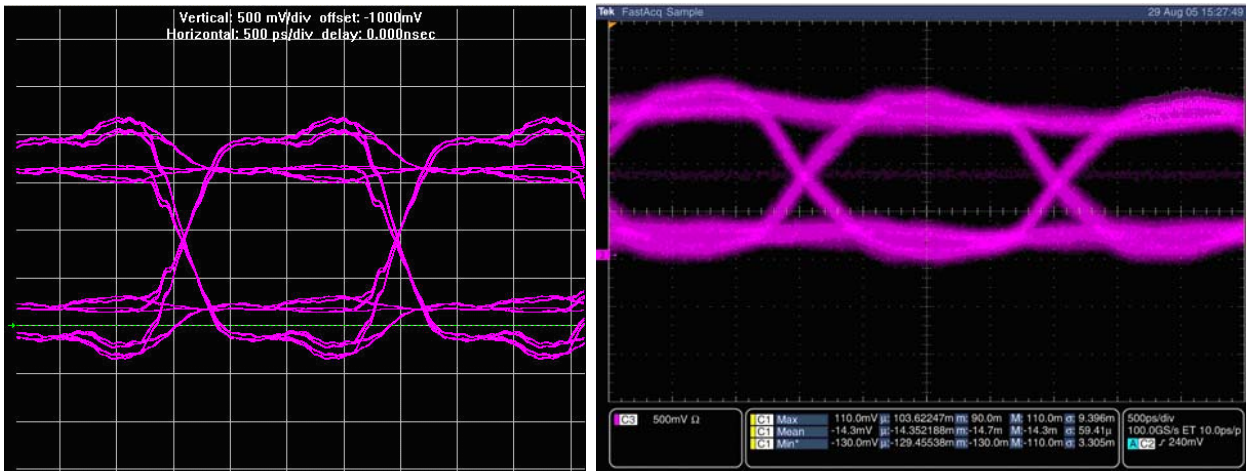


Table 4–20 compares the signal for a single rank DDR2 SDRAM DIMM and a single DDR2 SDRAM component in a Class II termination scheme when the FPGA is writing to the memory.

Table 4–20. Simulation and Board Measurement Results for Single Rank DDR2 SDRAM DIMM and Single DDR2 SDRAM Component *(1)*, *(2)*

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Single DDR2 SDRAM Component						
Simulation	1.79	1.15	0.39	0.33	3.90	3.43
Measurement	1.43	0.96	0.10	0.13	1.43	1.43
Single Rank DDR2 SDRAM DIMM						
Simulation	1.65	0.86	N/A	N/A	1.71	1.95
Measurement	1.36	0.41	N/A	N/A	1.56	1.56

Notes to Table 4–20:

(1) The drive strength on the FPGA is set to Class II 16 mA.

(2) N/A is not applicable.

The overall signal quality is comparable between the single rank DDR2 SDRAM DIMM and the single DDR2 SDRAM component, but the elimination of the DIMM connector and memory-side series resistor results in a more than 50% improvement in the eye height.

FPGA Reading from Memory

Figure 4–51 shows the Class II termination scheme without the memory-side series resistor when the FPGA is reading from memory. Without the memory-side series resistor, the memory driver has less loading to drive the Class II termination. Compare this result to the result of the DDR2 SDRAM DIMM described in "FPGA Reading from Memory" on page 4–32 where the memory-side series resistor is on the DIMM.

Figure 4–51. Class II Termination Scheme without Memory-Side Series Resistor

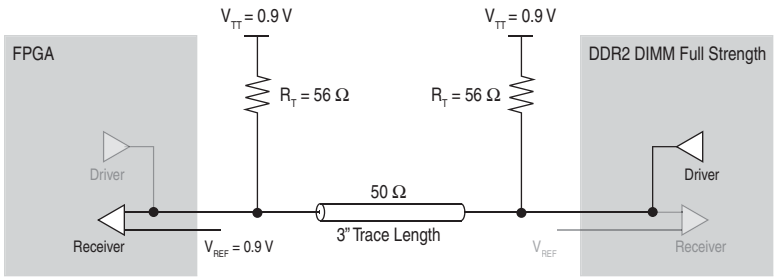


Figure 4–52 shows the simulation and measurement results of the signal seen at the FPGA. The FPGA reads from memory without the source-series resistor near the DDR2 SDRAM component on a Class II-terminated transmission line. The FPGA reads from memory with a full drive strength setting.

Figure 4–52. HyperLynx Simulation and Measurement, FPGA Reading from the DDR2 SDRAM Component

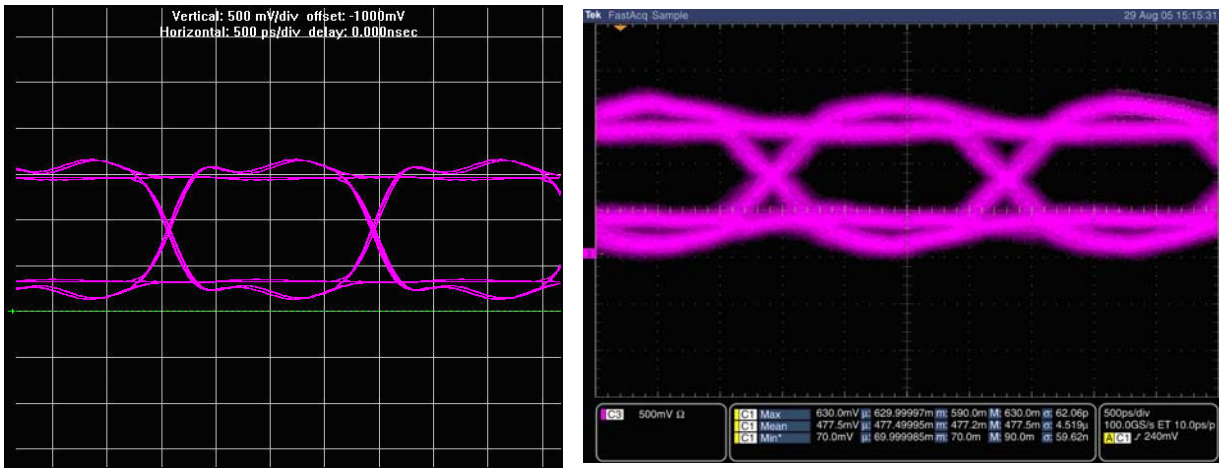


Table 4–21 compares the signal at a single rank DDR2 SDRAM DIMM and a single DDR2 SDRAM component of a Class II termination scheme. The FPGA is reading from memory with a full drive strength setting.

Table 4–21. Simulation and Board Measurement Results of Single Rank DDR2 SDRAM DIMM and DDR2 SDRAM Component *(1)*

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Single DDR2 SDRAM Component						
Simulation	1.79	1.06	N/A	N/A	2.48	3.03
Measurement	1.36	0.63	0.13	0.00	1.79	1.14
Single Rank DDR2 SDRAM DIMM						
Simulation	1.73	0.76	N/A	N/A	1.71	1.95
Measurement	1.28	0.43	N/A	N/A	0.93	0.86

Note to Table 4–21:

(1) N/A is not applicable.

The effect of eliminating the DIMM connector and memory-side series resistor is evident in the improvement in the eye height.

Single- Versus Dual-Rank DIMM

DDR2 SDRAM DIMMs are available in either single- or dual-rank DIMM. Single-rank DIMMs are DIMMs with DDR2 SDRAM memory components on one side of the DIMM. Higher-density DIMMs are available as dual-rank, which has DDR2 SDRAM memory components on both sides of the DIMM. With the dual-rank DIMM configuration, the loading is twice that of a single-rank DIMM. Depending on the board design, you must adjust the drive strength setting on the memory controller to account for this increase in loading. Figure 4–53 shows the simulation result of the signal seen at a dual rank DDR2 SDRAM DIMM. The simulation uses Class II termination with a memory-side series resistor transmission line. The FPGA uses a 16-

mA drive strength setting.

Figure 4–53. HyperLynx Simulation with a 16-mA Drive Strength Setting on the FPGA

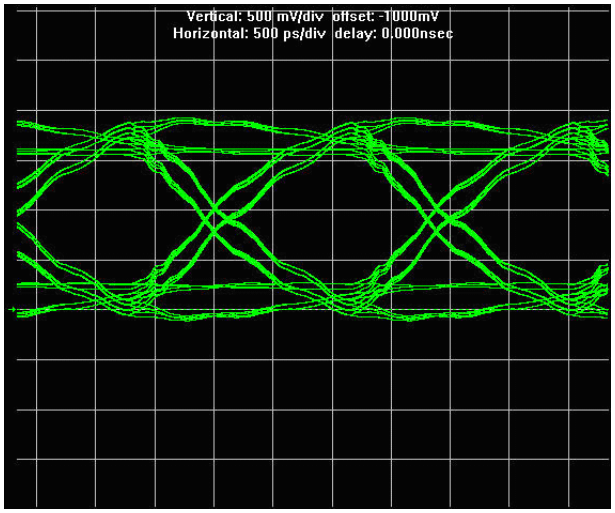


Table 4–22 compares the signals at a single- and dual-rank DDR2 SDRAM DIMM of a Class II and far-end source-series termination when the FPGA is writing to the memory with a 16-mA drive strength setting.

Table 4–22. Simulation Results of Single- and Dual-Rank DDR2 SDRAM DIMM *(1)*

	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Dual Rank DDR2 SDRAM DIMM						
Simulation	1.34	1.27	0.12	0.12	0.99	0.94
Single Rank DDR2 SDRAM DIMM						
Simulation	1.65	1.27	0.10	0.10	1.71	1.95

Note to Table 4–22:

(1) The drive strength on the FPGA is set to Class II 16 mA.

In a dual-rank DDR2 SDRAM DIMM, the additional loading leads to a slower edge rate, which affects the eye width. The slower edge rate leads to the degradation of the setup and hold time required by the memory as well, which must be taken into consideration during the analysis of the timing for the interface. The overall signal quality remains comparable, but eye width is reduced in the dual-rank DIMM. This reduction in eye width leads to a smaller data capture window that must be taken into account when performing timing analysis for the memory interface.

Single DIMM Versus Multiple DIMMs

Some applications, such as packet buffering, require deeper memory, making a single DIMM interface insufficient. If you use a multiple DIMM configuration to increase memory depth, the memory controller is required to interface with multiple data strobes and the data lines instead of the point-to-point interface in a single DIMM configuration. This results in heavier loading on the interface, which can potentially impact the overall performance of the memory interface.

For detailed information about a multiple DIMM DDR2 SDRAM memory interface, refer to the *Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

Design Layout Guidelines

This section discusses general layout guidelines for designing your DDR2 and DDR3 SDRAM interfaces. These layout guidelines help you plan your board layout, but are not meant as strict rules that must be adhered to. Altera recommends that you perform your own board-level simulations to ensure that the layout you choose for your board allows you to achieve your desired performance.

These layout guidelines are for both ALTMEMPHY- and UniPHY-based IP designs, unless specified otherwise.

For more information about how the memory manufacturers route these address and control signals on their DIMMs, refer to the Cadence PCB browser from the Cadence website, at www.cadence.com. The various JEDEC example DIMM layouts are available from the JEDEC website, at www.jedec.org.

For more information about board skew parameters, refer to Board Skews in the Implementing and Parameterizing Memory IP chapter. For assistance in calculating board skew parameters, refer to the Board Skew Parameters Tool, which is available through the solution at this location:
http://www.altera.com/support/kdb/solutions/rd10232012_771.html

The following layout guidelines include several +/- length based rules. These length based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristic of the interface. They do not include any margin for crosstalk.

Altera recommends that you get accurate time base skew numbers for your design when you simulate the specific implementation.

Layout Guidelines for DDR2 SDRAM Interface

Unless otherwise specified, the guidelines in this section apply to the following topologies:

- DIMM—UDIMM topology
- DIMM—RDIMM topology
- Discrete components laid out in UDIMM topology
- Discrete components laid out in RDIMM topology

Trace lengths for CLK and DQS should tightly match for each memory component. To match the trace lengths on the board, a balanced tree topology is recommended for clock and address and command signal routing. In addition to matching the trace lengths, you should ensure that DDR timing is passing in the Report DDR Timing report. For Stratix devices, this timing is shown as Write Leveling tDQSS timing. For Arria and Cyclone devices, this timing is shown as CK vs DQS timing

Table 4–23 lists device family topology support.

Table 4–23. Device Family Topology Support

Device	I/O Support
Arria II	Non-leveling
Arria V GX, Arria V GT	Non-leveling
Arria V GZ	Leveling
Cyclone V GX, Cyclone V GT	Non-leveling
Stratix III	Leveling
Stratix IV	Leveling
Stratix V	Leveling

Table 4–24 lists DDR2 SDRAM layout guidelines. These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the slew rate and propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

The following layout guidelines also apply to DDR3 SDRAM without leveling interfaces.

Table 4–24. DDR2 SDRAM Layout Guidelines (Part 1 of 4) (1)

Parameter	Guidelines
DIMMs	If you consider a normal DDR2 unbuffered, unregistered DIMM, essentially you are planning to perform the DIMM routing directly on your PCB. Therefore, each address and control pin routes from the FPGA (single pin) to all memory devices must be on the same side of the FPGA.
Impedance	■ All signal planes must be 50-60-Ω, single-ended, ±10% ■ All signal planes must be 100Ω, differential ±10% ■ All unused via pads must be removed, because they cause unwanted capacitance

Table 4–24. DDR2 SDRAM Layout Guidelines (Part 2 of 4) *(1)*

Parameter	Guidelines
Decoupling Parameter	■ Use 0.1 µF in 0402 size to minimize inductance ■ Make V_{TT} voltage decoupling close to pull-up resistors ■ Connect decoupling caps between V_{TT} and ground ■ Use a 0.1µF cap for every other V_{TT} pin and 0.01µF cap for every VDD and VDDQ pin
Power	■ Route GND, 1.8 V as planes ■ Route V_{CCIO} for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation ■ Route V_{TT} as islands or 250-mil (6.35-mm) power traces ■ Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces
General Routing	All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propogation variance, Altera recommend that signals from the same net group always be routed on the same layer. ■ Use 45° angles (*not* 90° corners) ■ Avoid T-Junctions for critical nets or clocks ■ Avoid T-junctions greater than 250 mils (6.35 mm) ■ Disallow signals across split planes ■ Restrict routing other signals close to system reset signals ■ Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks ■ All data, address, and command signals must have matched length traces ± 50 ps (±0.250 inches or 6.35 mm) ■ All signals within a given **Byte Lane Group** should be matched length with maximum deviation of ±10 ps or approximately ±0.050 inches (1.27 mm) and routed in the same layer.
Clock Routing	■ A 4.7 KΩ resistor to ground is recommended for each Clock Enable signal. You can place the resistor at either the memory end or the FPGA end of the trace. ■ Route clocks on inner layers with outer-layer run lengths held to under 500 mils (12.7 mm) ■ These signals should maintain a10-mil (0.254 mm) spacing from other nets ■ Clocks should maintain a length-matching between clock pairs of ±5 ps or approximately ±25 mils (0.635 mm) ■ Differential clocks should maintain a length-matching between `P` and `N` signals of ±2 ps or approximately ±10 mils (0.254 mm), routed in parallel ■ Space between different pairs should be at least three times the space between the differential pairs and must be routed differentially (5-mil trace, 10-15 mil space on centers), and equal to the signals in the `Address/Command` Group or up to 100 mils (2.54 mm) longer than the signals in the `Address/Command` Group. ■ Trace lengths for CLK and DQS should closely match for each memory component. To match trace lengths on the board, a balanced tree topology is recommended for clock and address and command signal routing. For Stratix device families, ensure that Write Leveling tDQSS is passing in the DDR timing report; for Arria and Cyclone device families, verify that CK vs DQS timing is passing in the DDR timing report.
Address and Command Routing	■ Unbuffered address and command lines are more susceptible to cross-talk and are generally noisier than buffered address or command lines. Therefore, un-buffered address and command signals should be routed on a different layer than data signals (`DQ`) and data mask signals (`DM`) and with greater spacing. ■ Do not route differential clock (`CK`) and clock enable (`CKE`) signals close to address signals.

Table 4–24. DDR2 SDRAM Layout Guidelines (Part 3 of 4) *(1)*

Parameter	Guidelines

DQ, DM, and DQS Routing Rules	■ Keep the distance from the pin on the DDR2 DIMM or component to the termination resistor pack (V_{TT}) to less than 500 mils for DQS[x] Data Groups.
	■ Keep the distance from the pin on the DDR2 DIMM or component to the termination resistor pack (V_{TT}) to less than 1000 mils for the ADR_CMD_CTL Address Group.
	■ Parallelism rules for the DQS[x] Data Groups are as follows:
	  ■ 4 mils for parallel runs < 0.1 inch (approximately 1× spacing relative to plane distance)
	  ■ 5 mils for parallel runs < 0.5 inch (approximately 1× spacing relative to plane distance)
	  ■ 10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance)
	  ■ 15 mils for parallel runs between 1.0 and 6.0 inch (approximately 3× spacing relative to plane distance)
	■ Parallelism rules for the ADR_CMD_CTL group and CLOCKS group are as follows:
	  ■ 4 mils for parallel runs < 0.1 inch (approximately 1× spacing relative to plane distance)
	  ■ 10 mils for parallel runs < 0.5 inch (approximately 2× spacing relative to plane distance)
	  ■ 15 mils for parallel runs between 0.5 and 1.0 inches (approximately 3× spacing relative to plane distance)
	  ■ 20 mils for parallel runs between 1.0 and 6.0 inches (approximately 4× spacing relative to plane distance)
	■ All signals are to maintain a 20-mil separation from other, non-related nets.
	■ All signals must have a total length of < 6 inches.
	■ Trace lengths for CLK and DQS should closely match for each memory component. To match trace lengths on the board, a balanced tree topology is recommended for clock and address and command signal routing. For Stratix device families, ensure that Write Leveling tDQSS is passing in the DDR timing report; for Arria and Cyclone device families, verify that CK vs DQS timing is passing in the DDR timing report.

Termination Rules	■ When pull-up resistors are used, fly-by termination configuration is recommended. Fly-by helps reduce stub reflection issues.
	■ Pull-ups should be within 0.5 to no more than 1 inch.
	■ Pull up is typically 56 Ω
	■ If using resistor networks:
	  ■ Do not share R-pack series resistors between address/command and data lines (DQ, DQS, and DM) to eliminate crosstalk within pack.
	  ■ Series and pull up tolerances are 1–2%.
	  ■ Series resistors are typically 10 to 20Ω
	  ■ Address and control series resistor typically at the FPGA end of the link.
	  ■ DM, DQS, DQ series resistor typically at the memory end of the link (or just before the first DIMM).
	■ If termination resistor packs are used:
	  ■ The distance to your memory device should be less than 750 mils.
	  ■ The distance from your Altera's FPGA device should be less than 1250 mils.

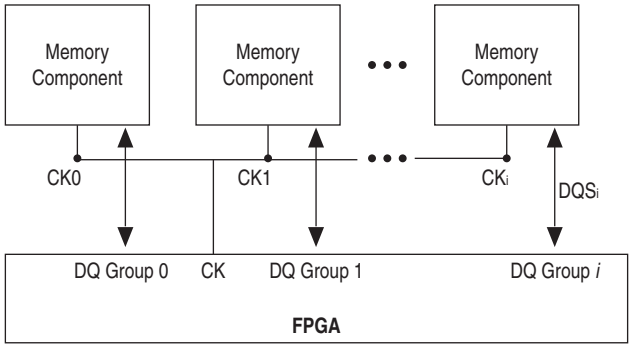
Table 4–24. DDR2 SDRAM Layout Guidelines (Part 4 of 4) *(1)*

Parameter	Guidelines
Quartus II Software Settings for Board Layout	■ To perform timing analyses on board and I/O buffers, use third party simulation tool to simulate all timing information such as skew, ISI, crosstalk, and type the simulation result into the UniPHY board setting panel. ■ Do not use advanced I/O timing model (AIOT) or board trace model unless you do not have access to any third party tool. AIOT provides reasonable accuracy but tools like HyperLynx provides better result. In operations with higher frequency, it is crucial to properly simulate all signal integrity related uncertainties. ■ The Quartus II software does timing check to find how fast the controller issues a write command after a read command, which limits the maximum length of the DQ/DQS trace. Check the turnaround timing in the Report DDR timing report and ensure the margin is positive before board fabrication. Functional failure happens if the margin is more than 0.

Note to Table 4–24:

(1) For point-to-point and DIMM interface designs, refer to the Micron website, www.micron.com.

Figure 4–54. Balanced Tree Topology



CK_i = Clock signal propagation delay to device *i*

DQS_i = DQ/DQS signals propagation delay to group *i*

Layout Guidelines for DDR3 SDRAM Interface

Table 4–25 lists DDR3 SDRAM layout guidelines.

Unless otherwise specified, the guidelines in Table 4–25 apply to the following topologies:

- DIMM—UDIMM topology
- DIMM—RDIMM topology
- DIMM—LRDIMM topology

☞ Not all versions of the Quartus II software support LRDIMM. Contact Altera if you require LRDIMM support in the Quartus II software.

- Discrete components laid out in UDIMM topology

- Discrete components laid out in RDIMM topology

These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the slew rate and propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

Refer to the *External Memory Interface Spec Estimator* for all supported frequencies and topologies.

For frequencies greater than 533 MHz, when you are calculating the delay associated with a trace, you must take the FPGA package delays into consideration. For more information, refer to Package Deskew.

For device families that do not support write leveling, refer to Layout Guidelines for DDR2 SDRAM Interface.

Table 4–25. DDR3 SDRAM Layout Guidelines (Part 1 of 5) *(1)*

Parameter	Guidelines
Impedance	■ All signal planes must be 50 Ω, single-ended, ±10%. ■ All signal planes must be 100 Ω, differential ±10%. ■ All unused via pads must be removed, because they cause unwanted capacitance.
Decoupling Parameter	■ Use 0.1 µF in 0402 size to minimize inductance. ■ Make V_{TT} voltage decoupling close to the DDR3 SDRAM components and pull-up resistors. ■ Connect decoupling caps between V_{TT} and V_{DD} using a 0.1µF cap for every other V_{TT} pin. ■ Use a 0.1µF cap and 0.01µF cap for every V_{DDQ} pin.
Power	■ Route GND,1.5 V and 0.75 V as planes. ■ Route V_{CCIO} for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation. ■ Route V_{TT} as islands or 250-mil (6.35-mm) power traces. ■ Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces.

Maximum Trace Length *(2)*	■ Even though there are no hard requirements for minimum trace length, you need to simulate the trace to ensure the signal integrity. Shorter routes result in better timing.
	■ For DIMM topology only:
	  ■ Maximum trace length for all signals from FPGA to the first DIMM slot is 4.5 inches.
	  ■ Maximum trace length for all signals from DIMM slot to DIMM slot is 0.425 inches.
	■ For discrete components only:
	  ■ Maximum trace length for address, command, control, and clock from FPGA to the first component must not be more than 7 inches.
	  ■ Maximum trace length for DQ, DQS, DQS#, and DM from FPGA to the first component is 5 inches.

Table 4–25. DDR3 SDRAM Layout Guidelines (Part 2 of 5) *(1)*

Parameter	Guidelines
General Routing	All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propogation variance, Altera recommend that you route signals from the same net group on the same layer. ■ Use 45° angles (*not* 90° corners). ■ Disallow critical signals across split planes. ■ Route over appropriate V_{CC} and GND planes. ■ Keep signal routing layers close to GND and power planes. ■ Avoid routing memory signals closer than 0.025 inch (0.635 mm) to memory clocks.

Clock Routing	■ A 4.7 KΩ resistor to ground is recommended for each Clock Enable signal. You can place the resistor at either the memory end or the FPGA end of the trace.
	■ Route clocks on inner layers with outer-layer run lengths held to under 500 mils (12.7 mm).
	■ Route clock signals in a daisy chain topology from the first SDRAM to the last SDRAM. The maximum length of the first SDRAM to the last SDRAM must not exceed 0.69 t_{CK}. For different DIMM configurations, check the appropriate JEDEC specification.
	■ These signals should maintain the following spacings:
	  ■ 10-mil (0.254 mm) spacing for parallel runs less than 0.5 inches or 2x trace-to-plane distance.
	  ■ 15-mil spacing for parallel runs between 0.5 and 1 inches or 3× trace-to-plane distance.
	  ■ 20-mil spacing for parallel runs between 1 and 6 inches or 4× trace-to-plane distance.
	■ Clocks should maintain a length-matching between clock pairs of ±5 ps or approximately ±25 mils (0.635 mm).
	■ Clocks should maintain a length-matching between positive (p) and negative (n) signals of ±2 ps or approximately ±10 mils (0.254 mm), routed in parallel.
	■ Space between different pairs should be at least two times the trace width of the differential pair to minimize loss and maximize interconnect density.
	■ To avoid mismatched transmission line to via, Altera recommends that you use Ground Signal Signal Ground (GSSG) topology for your clock pattern—GND|CLKP|CKLN|GND.
	■ Route all addresses and commands to match the clock signals to within ±20 ps or approximately ± 125 mil (± 3.175 mm) to each discrete memory component. Refer to Figure 4–55.

Table 4–25. DDR3 SDRAM Layout Guidelines (Part 3 of 5) *(1)*

Parameter	Guidelines
Address and Command Routing	■ Route address and command signals in a daisy chain topology from the first SDRAM to the last SDRAM. The maximum length of the first SDRAM to the last SDRAM must not be more than 0.69 t_{CK}. For different DIMM configurations, check the appropriate JEDEC specifications. ■ UDIMMs are more susceptible to cross-talk and are generally noisier than buffered DIMMs. Therefore, route address and command signals of UDIMMs on a different layer than data signals (DQ) and data mask signals (DM) and with greater spacing. ■ Do not route differential clock (CK) and clock enable (CKE) signals close to address signals. ■ Route all addresses and commands to match the clock signals to within ±20 ps or approximately ± 125 mil (± 3.175 mm) to each discrete memory component. Refer to Figure 4–55. ■ Parallelism rules for address and command and clock signals are as follows: ■ 4 mils for parallel runs <0.1 inch (approximately 1× spacing relative to plane distance) ■ 10 mils for parallel runs <0.5 inch (approximately 2× spacing relative to plane distance) ■ 15 mils for parallel runs between 0.5 and 1.0 inches (approximately 3× spacing relative to plane distance) ■ 20 mils for parallel runs between 1.0 and 6.0 inches (approximately 4× spacing relative to plane distance)

Table 4–25. DDR3 SDRAM Layout Guidelines (Part 4 of 5) *(1)*

Parameter	Guidelines

DQ, DM, and DQS Routing Rules	■ All the trace length matching requirements are from the FPGA package ball to DDR3 package ball, which means you have to take into account trace mismatching on different DIMM raw cards.
	■ Match in length all DQ, DQS, and DM signals within a given byte-lane group with a maximum deviation of ±10 ps or approximately ± 50 mils (± 1.27 mm).
	■ Ensure to route all DQ, DQS, and DM signals within a given byte-lane group on the same layer to avoid layer to layer transmission velocity differences, which otherwise increase the skew within the group.
	■ Parallelism rules for all signals (other than address and command) are as follows:
	  ■ 5 mils for parallel runs < 0.5 inch (approximately 1× spacing relative to plane distance)
	  ■ 10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance)
	  ■ 15 mils for parallel runs between 1.0 and 6.0 inch (approximately 3× spacing relative to plane distance)
	■ Do not use DDR3 deskew to correct for more than 20 ps of DQ group skew. The skew algorithm only removes the following possible uncertainties:
	  ■ Minimum and maximum die IOE skew or delay mismatch
	  ■ Minimum and maximum device package skew or mismatch
	  ■ Board delay mismatch of 20 ps
	  ■ Memory component DQ skew mismatch
	■ Increasing any of these four parameters runs the risk of the deskew algorithm limiting, failing to correct for the total observed system skew. If the algorithm cannot compensate without limiting the correction, timing analysis shows reduced margins.
	■ For ALTMEMPHY-based interfaces, keep the maximum byte-lane group-to-byte group matched length deviation to ± 150 ps or ± 0.8 inches (± 20 mm).
	■ For UniPHY-based interfaces, the timing between the DQS and clock signals on each device calibrates dynamically to meet t_{DQSS}. To make sure the skew is not too large for the leveling circuit's capability, refer to Figure 4–56 and follow these rules:
	  ■ Propagation delay of clock signal must not be shorter than propagation delay of DQS signal at every device:
	  $(CK_i) - DQS_i > 0$; $0 < i <$ number of components – 1
	  ■ Total skew of CLK and DQS signal between groups is less than one clock cycle:
	  $(CK_i + DQS_i)$ max – $(CK_i + DQS_i)$ min < 1 × t_{CK}
	
	  (If you are using a DIMM topology, your delay and skew must take into consideration values for the actual DIMM.)

Table 4–25. DDR3 SDRAM Layout Guidelines (Part 5 of 5) (1)

Parameter	Guidelines
Termination Rules	■ When using DIMMs, you have no concerns about terminations on memory clocks, addresses, and commands. ■ If you are using components, use an external parallel termination of 40 Ω to V_{TT} at the end of the fly-by daisy chain topology on the addresses and commands. ■ For memory clocks, use an external parallel termination of 75 Ω differential at the end of the fly-by daisy chain topology on the memory clocks. Fly-by daisy chain topology helps reduce stub reflection issues. ■ If you include a compensation capacitor at the first memory load, it may improve the waveform signal integrity. ■ Keep the length of the traces to the termination to within 0.5 inch (14 mm). ■ Use resistors with tolerances of 1 to 2%.
Quartus II Software Settings for Board Layout	■ To perform timing analyses on board and I/O buffers, use third party simulation tool to simulate all timing information such as skew, ISI, crosstalk, and type the simulation result into the UniPHY board setting panel. ■ Do not use advanced I/O timing model (AIOT) or board trace model unless you do not have access to any third party tool. AIOT provides reasonable accuracy but tools like HyperLynx provide better results.

Notes to Table 4–24:

(1) For point-to-point and DIMM interface designs, refer to the Micron website, www.micron.com.

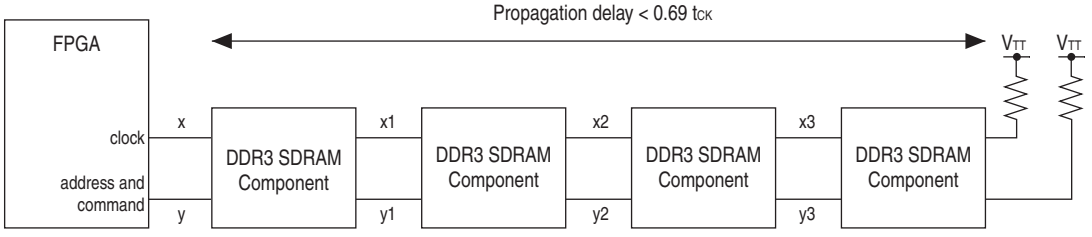
(2) For better efficiency, the UniPHY IP requires faster turnarounds from read commands to write.

Length Matching Rules

This section provides additional guidance on length matching for different types of DDR3 signals.

Route all addresses and commands to match the clock signals to within ±20 ps or approximately ± 125 mil (± 3.175 mm) to each discrete memory component. Figure 4–55 shows the DDR3 SDRAM component routing guidelines for address and command signals.

Figure 4–55. DDR3 SDRAM Component Address and Command Routing Guidelines



If using discrete components:
x = y ± 125 mil
x + x1 = y + y1 ± 125 mil
x + x1 + x2 = y + y1 + y2 ± 125 mil
x + x1 + x2 + x3 = y + y1 + y2 +y3 ± 125 mil

If using a DIMM topology:
x=y +/- 125 mil

The timing between the DQS and clock signals on each device calibrates dynamically to meet t_{DQSS}. Figure 4–56 shows the delay requirements to align DQS and clock signals. To ensure that the skew is not too large for the leveling circuit's capability, follow these rules:

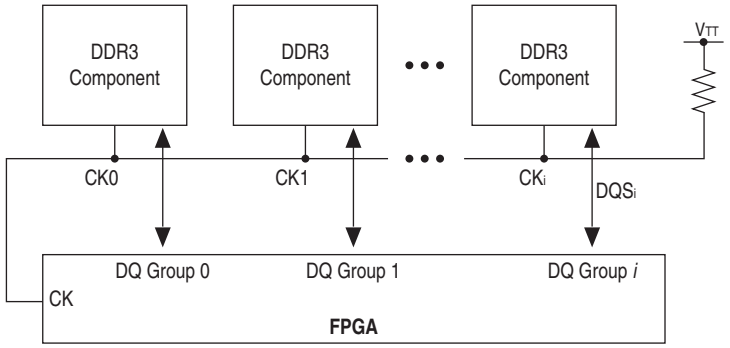
- Propagation delay of clock signal must not be shorter than propagation delay of DQS signal at every device:

```
CKi  – DQSi > 0; 0 < i < number of components – 1
```

- Total skew of CLK and DQS signal between groups is less than one clock cycle:

```
(CKi + DQSi) max – (CKi + DQSi) min < 1 × tCK
```

Figure 4–56. Delaying DQS Signal to Align DQS and Clock



CK_i = Clock signal propagation delay to device *i*

DQS_i = DQ/DQS signals propagation delay to group *i*

Clk pair matching—If you are using a DIMM (UDIMM, RDIMM, or LRDIMM) topology, match the trace lengths up to the DIMM connector. If you are using discrete components (UDIMM or RDIMM), match the lengths for all the memory components connected in the fly-by chain.

DQ group length matching—If you are using a DIMM (UDIMM, RDIMM, or LRDIMM) topology, match the trace lengths up to the DIMM connector. If you are using discrete components (UDIMM or RDIMM), match the lengths up to the respective memory components.

When you are using DIMMs, it is assumed that lengths are tightly matched within the DIMM itself. You should check that appropriate traces are length-matched within the DIMM.

Layout Guidelines for DDR3 SDRAM Wide Interface (>72 bits)

This section discusses the different ways to lay out a wider DDR3 SDRAM interface to the FPGA. Choose the topology based on board trace simulation and the timing budget of your system.

The UniPHY IP supports up to a 144-bit wide DDR3 interface. You can either use discrete components or DIMMs to implement a wide interface (any interface wider than 72 bits). Altera recommends using leveling when you implement a wide interface with DDR3 components.

When you lay out for a wider interface, all rules and constraints discussed in the previous sections still apply. The DQS, DQ, and DM signals are point-to-point, and all the same rules discussed in "Design Layout Guidelines" on page 4–57 apply.

The main challenge for the design of the fly-by network topology for the clock, command, and address signals is to avoid signal integrity issues, and to make sure you route the DQS, DQ, and DM signals with the chosen topology.

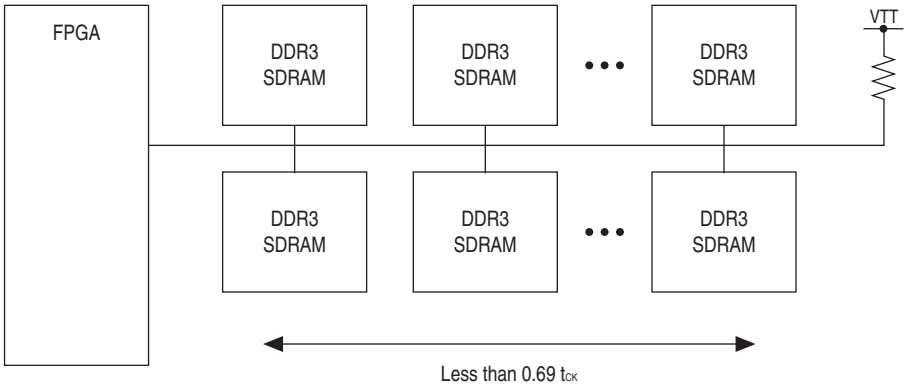
Fly-By Network Design for Clock, Command, and Address Signals

As described in "DDR3 SDRAM Components With Leveling" on page 4–45, the UniPHY IP requires the flight-time skew between the first DDR3 SDRAM component and the last DDR3 SDRAM component to be less than 0.69 t_{CK} for memory clocks. This constraint limits the number of components you can have for each fly-by network.

If you design with discrete components, you can choose to use one or more fly-by networks for the clock, command, and address signals.

Figure 4–57 shows an example of a single fly-by network topology.

Figure 4–57. Single Fly-By Network Topology



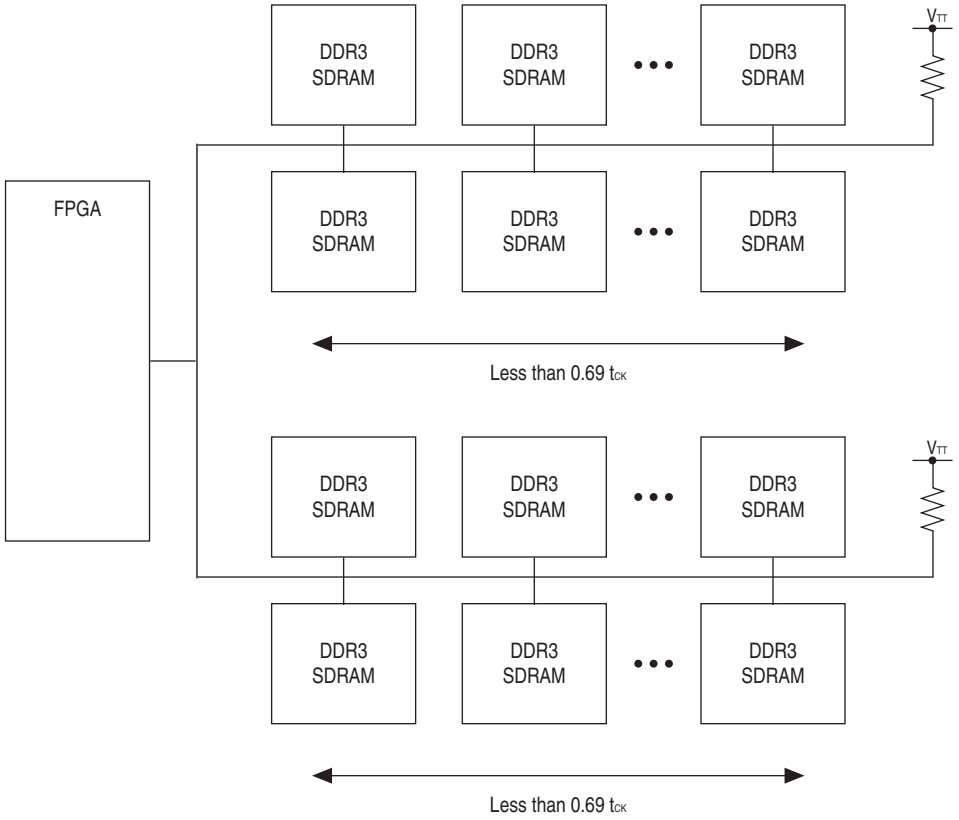
Every DDR3 SDRAM component connected to the signal is a small load that causes discontinuity and degrades the signal. When using a single fly-by network topology, to minimize signal distortion, follow these guidelines:

- Use ×16 device instead ×4 or ×8 to minimize the number of devices connected to the trace.
- Keep the stubs as short as possible.

- Even with added loads from additional components, keep the total trace length short; keep the distance between the FPGA and the first DDR3 SDRAM component less than 5 inches.
- Simulate clock signals to ensure a decent waveform.

Figure 4–58 shows an example of a double fly-by network topology. This topology is not rigid but you can use it as an alternative option. The advantage of using this topology is that you can have more DDR3 SDRAM components in a system without violating the 0.69 t_{CK} rule. However, as the signals branch out, the components still create discontinuity.

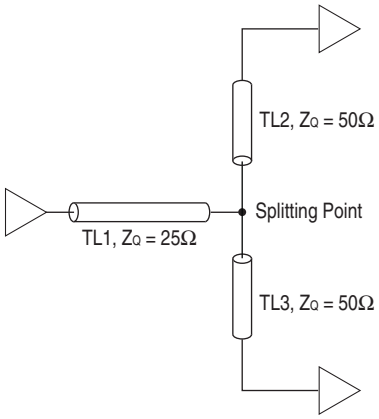
Figure 4–58. Double Fly-By Network Topology



You need to carry out some simulations to find the location of the split, and the best impedance for the traces before and after the split.

Figure 4–59 shows a way to minimize the discontinuity effect. In this example, keep TL2 and TL3 matches in length. Keep TL1 longer than TL2 and TL3, so that it is easier to route all the signals during layout.

Figure 4–59. Minimizing Discontinuity Effect



You can also consider using a DIMM on each branch to replace the components. Because the trade impedance on the DIMM card is 40 Ω to 60 Ω, perform a board trace simulation to control the reflection to within the level your system can tolerate.

By using the new features of the DDR3 SDRAM controller with UniPHY and the Stratix III, Stratix IV, or Stratix V devices, you simplify your design process. Using the fly-by daisy chain topology increases the complexity of the datapath and controller design to achieve leveling, but also greatly improves performance and eases board layout for DDR3 SDRAM.

You can also use the DDR3 SDRAM components without leveling in a design if it may result in a more optimal solution, or use with devices that support the required electrical interface standard, but do not support the required read and write leveling functionality.

Package Deskew

Trace lengths inside the device package are not uniform for all package pins. The nonuniformity of package traces can affect system timing for high frequencies. In the Quartus II software version 12.0 and later, a package deskew option is available.

If you do not enable the package deskew option, the Quartus II software uses the package delay numbers to adjust skews on the appropriate signals; you do not need to adjust for package delays on the board traces. If you do enable the package deskew option, the Quartus II software does not use the package delay numbers for timing analysis, and you must deskew the package delays with the board traces for the appropriate signals for your design.

DQ/DQS/DM Deskew

For Stratix V DDR3 UniPHY designs running at frequencies equal to or greater than 533 MHz, you must take into account the package delays of all data pins (DQ/DQS/DM) when you calculate the total trace delay. To get the package delay information, follow these steps:

1. Select the **DQ/DQS Package Deskew** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Run **Analysis and Synthesis** in the Quartus II software.
5. Run the `<core_name>_p0_pin_assignment.tcl` script.
6. Compile your design.
7. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

Address and Command Deskew

For frequencies of 1 GHz or higher, you should deskew address and command delays as follows:

1. Select the **Address/Command Package Deskew** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Run **Analysis and Synthesis** in the Quartus II software.
5. Run the `<core_name>_p0_pin_assignment.tcl` script.
6. Compile your design.
7. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

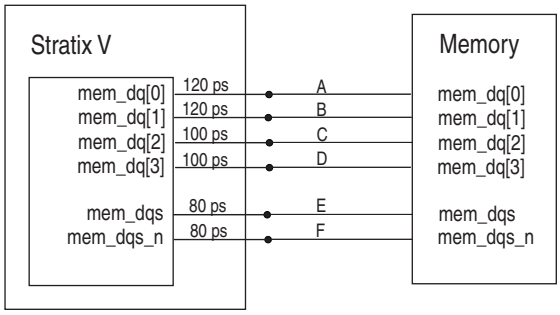
Deskew Example

Consider an example where you want to deskew an interface with 4 DQ pins, 1 DQS pin, and 1 DQSn pin. Let's assume an operating frequency of 667 MHz, and the package lengths for the pins reported in the **.pin** file as as follows:

```
dq[0] = 120 ps
dq[1] = 120 ps
dq[2] = 100 ps
dq[3] = 100 ps
dqs   = 80 ps
dqs_n = 80 ps
```

Figure 4–60 illustrates this example.

Figure 4–60. Deskew Example

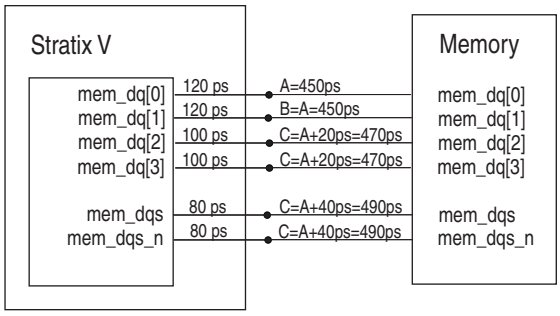


When you perform length matching for all the traces in the DQS group, you must take package delays into consideration. Because the package delays of traces A and B are 40 ps longer than the package delays of traces E and F, you would need to make the board traces for E and F 40 ps longer than the board traces for A and B.

A similar methodology would apply to traces C and D, which should be 20 ps longer than the lengths of traces A and B.

Figure 4–61 shows this scenario with the length of trace A at 450 ps.

Figure 4–61. Deskew Example With Trace Delay Calculations



When you enter the board skews into the Board Settings tab of the DDR3 parameter editor, you should calculate the board skew parameters as the sums of board delay and corresponding package delay. If a pin does not have a package delay (such as address and command pins), you should use the board delay only.

The example of Figure 4–61 shows an ideal case where board skews are perfectly matched. In reality, you should allow plus or minus 2 ps of skew mismatch within a DQS group (DQ/DQS/DM).

Package Migration

It is important to note that package delays can be different for the same pin in different packages. If you want to use multiple migratable packages in your system, you should compensate for package skew as described below:

Assume two migratable packages, device A and device B, and that you want to compensate for the board trace lengths for device A. Follow these steps:

1. Compile your design for device A, with the Package Skew option enabled.
2. Note the skews in the `<core_name>.pin` file for device A. Deskew these package skews with board trace lengths as described in the preceding examples.
3. Recompile your design for device A.
4. Compile your design for device B, with the Package Skew option enabled.
5. Note the skews in the `<core_name>.pin` file for device B. Calculate the maximum absolute package skew difference between the two packages.
6. Take the value calculated in step 4, and add to it the board deskew uncertainty (which, by default, is 20 ps). Enter the resulting value as the **Maximum skew within a DQS group** on the board settings tab.
7. Recompile the design for device B.
8. Verify that timing is correct for both device A and device B.

Document Revision History

Table 4–26 lists the revision history for this document.

Table 4–26. Document Revision History

Date	Version	Changes
November 2012	5.0	■ Updated Layout Guidelines for DDR2 SDRAM Interface and Layout Guidelines for DDR3 SDRAM Interface. ■ Added LRDIMM support. ■ Added Package Deskew section.
June 2012	4.1	Added Feedback icon.
November 2011	4.0	Added Arria V and Cyclone V information.
June 2011	3.0	■ Merged DDR2 and DDR3 chapters to *DDR2 and DDR3 SDRAM Interface Termination and Layout Guidelines* and updated with leveling information. ■ Added Stratix V information.
December 2010	2.1	Added *DDR3 SDRAM Interface Termination, Drive Strength, Loading, and Board Layout Guidelines* chapter with Stratix V information.
July 2010	2.0	Updated Arria II GX information.
April 2010	1.0	Initial release.

5. Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines

EMI_DG_005-4.1

This chapter describes guidelines for implementing dual unbuffered DIMM (UDIMM) DDR2 and DDR3 SDRAM interfaces. This chapter discusses the impact on signal integrity of the data signal with the following conditions in a dual-DIMM configuration:

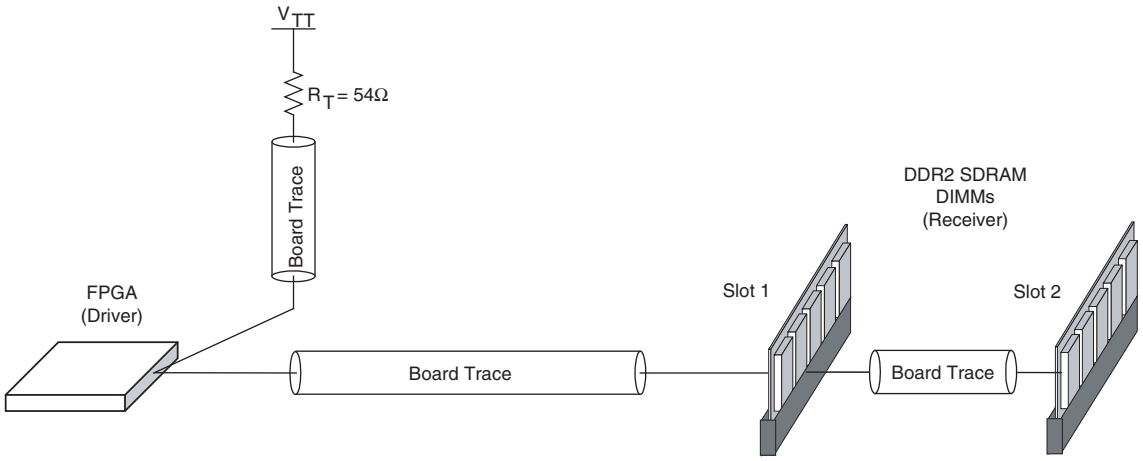
- Populating just one slot versus populating both slots
- Populating slot 1 versus slot 2 when only one DIMM is used
- On-die termination (ODT) setting of 75 Ω versus an ODT setting of 150 Ω

For detailed information about a single-DIMM DDR2 SDRAM interface, refer to the *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

DDR2 SDRAM

This section describes guidelines for implementing a dual slot unbuffered DDR2 SDRAM interface, operating at up to 400-MHz and 800-Mbps data rates. Figure 5–1 shows a typical DQS, DQ, and DM signal topology for a dual-DIMM interface configuration using the ODT feature of the DDR2 SDRAM components.

Figure 5–1. Dual-DIMM DDR2 SDRAM Interface Configuration *(1)*



Note to Figure 5–1:

(1) The parallel termination resistor R_T = 54 Ω to V_{TT} at the FPGA end of the line is optional for devices that support dynamic on-chip termination (OCT).

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

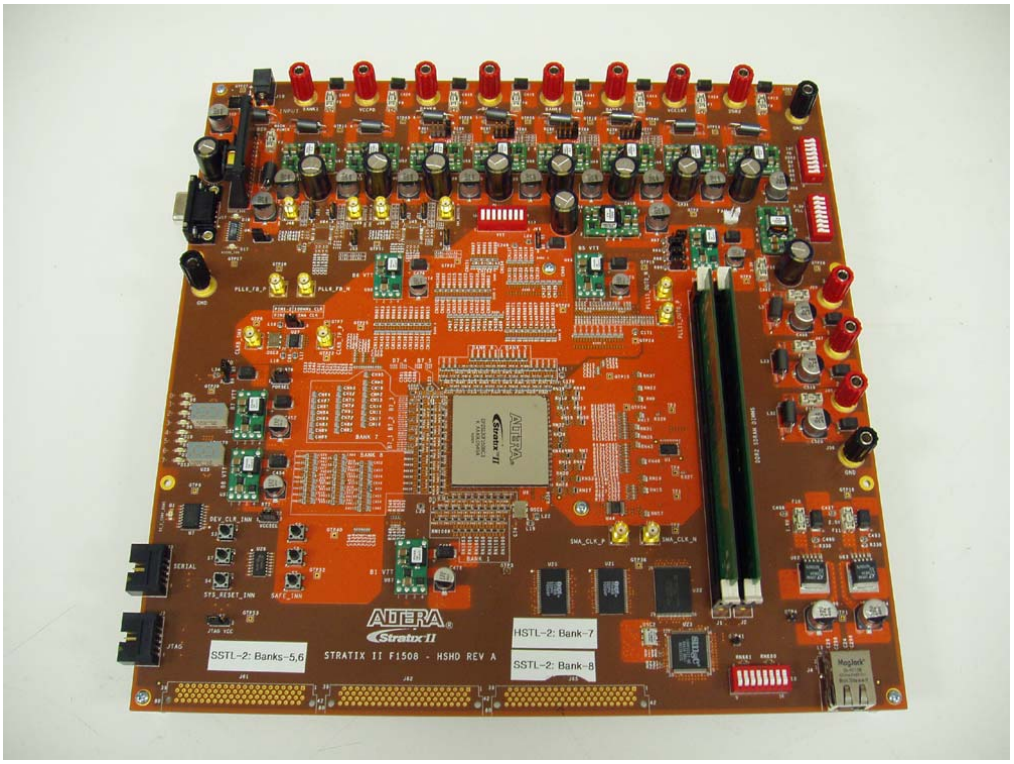
The simulations in this section use a Stratix® II device-based board. Because of limitations of this FPGA device family, simulations are limited to 266 MHz and 533 Mbps so that comparison to actual hardware results can be directly made.

Stratix II High Speed Board

To properly study the dual-DIMM DDR2 SDRAM interface, the simulation and measurement setup evaluated in the following analysis features a Stratix II FPGA interfacing with two 267-MHz DDR2 SDRAM UDIMMs. This DDR2 SDRAM interface is built on the Stratix II High-Speed High-Density Board (Figure 5–2).

For more information about the Stratix II High-Speed High-Density Board, contact your Altera® representative.

Figure 5–2. Stratix II High-Speed Board with Dual-DIMM DDR2 SDRAM Interface



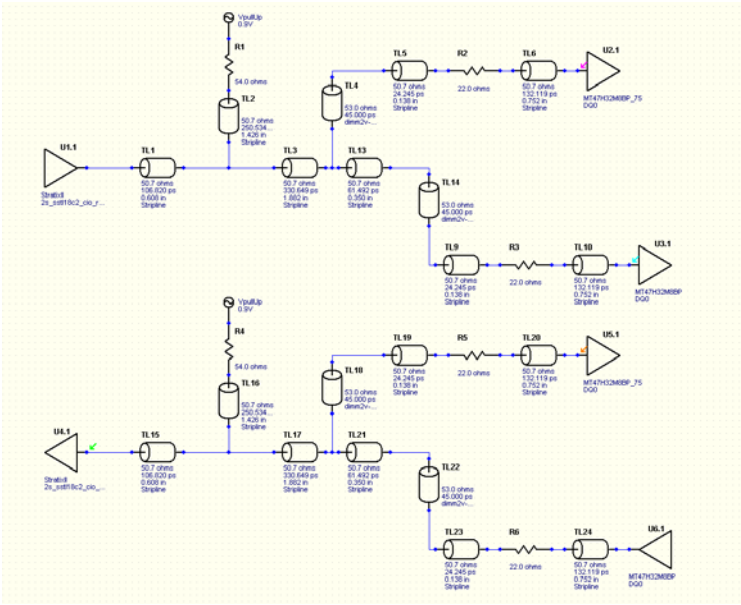
The Stratix II High-Speed Board uses a Stratix II 2S90F1508 device. For DQS, DQ, and DM signals, the board is designed without external parallel termination resistors near the DDR2 SDRAM DIMMs, to take advantage of the ODT feature of the DDR2 SDRAM components. Stratix II FPGA devices are not equipped with dynamic OCT, so external parallel termination resistors are used at the FPGA end of the line.

Stratix III and Stratix IV devices, which support dynamic OCT, do not require FPGA end parallel termination. Hence this discrete parallel termination is optional.

The DDR2 SDRAM DIMM contains a 22-Ω external series termination resistor for each data strobe and data line, so all the measurements and simulations need to account for the effect of these series termination resistors.

To correlate the bench measurements done on the Stratix II High Speed High Density Board, the simulations are performed using HyperLynx LineSim Software with IBIS models from Altera and memory vendors. Figure 5–3 is an example of the simulation setup in HyperLynx used for the simulation.

Figure 5–3. HyperLynx Setup for Simulating the Stratix II High Speed High Density with Dual-DIMM DDR2 SDRAM Interface



Overview of ODT Control

When there is only a single-DIMM on the board, the ODT control is relatively straightforward. During write to the memory, the ODT feature of the memory is turned on; during read from the memory, the ODT feature of the memory is turned off. However, when there are multiple DIMMs on the board, the ODT control becomes more complicated.

With a dual-DIMM interface on the system, the controller has different options for turning the memory ODT on or off during read or write. Table 5–1 lists the DDR2 SDRAM ODT control during write to the memory; Table 5–2 during read from the memory. These DDR2 SDRAM ODT controls are recommended by Samsung Electronics. The JEDEC DDR2 specification was updated to include optional support for R_{TT}(nominal) = 50 Ω

For more information about the DDR2 SDRAM ODT controls recommended by Samsung, refer to the *Samsung DDR2 Application Note: ODT (On Die Termination) Control*.

Table 5–1. DDR2 SDRAM ODT Control—Writes (1)

Slot 1 (2)	Slot 2 (2)	Write To	FPGA	Module in Slot 1		Module in Slot 2	
				Rank 1	Rank 2	Rank 3	Rank 4
DR	DR	Slot 1	Series 50 Ω	Infinite	Infinite	75 or 50 Ω	Infinite
		Slot 2	Series 50 Ω	75 or 50 Ω	Infinite	Infinite	infinite
SR	SR	Slot 1	Series 50 Ω	Infinite	Unpopulated	75 or 50 Ω	Unpopulated
		Slot 2	Series 50 Ω	75 or 50 Ω	Unpopulated	Infinite	Unpopulated
DR	Empty	Slot 1	Series 50 Ω	150 Ω	Infinite	Unpopulated	Unpopulated
Empty	DR	Slot 2	Series 50 Ω	Unpopulated	Unpopulated	150 Ω	Infinite
SR	Empty	Slot 1	Series 50 Ω	150 Ω	Unpopulated	Unpopulated	Unpopulated
Empty	SR	Slot 2	Series 50 Ω	Unpopulated	Unpopulated	150 Ω	Unpopulated

Notes to Table 5–1:

(1) For DDR2 at 400 MHz and 533 Mbps = 75 Ω; for DDR2 at 667 MHz and 800 Mbps = 50 Ω

(2) SR = single ranked; DR = dual ranked.

Table 5–2. DDR2 SDRAM ODT Control—Reads (1)

Slot 1 (2)	Slot 2 (2)	Read From	FPGA	Module in Slot 1		Module in Slot 2	
				Rank 1	Rank 2	Rank 3	Rank 4
DR	DR	Slot 1	Parallel 50 Ω	Infinite	Infinite	75 or 50 Ω	Infinite
		Slot 2	Parallel 50 Ω	75 or 50 Ω	Infinite	Infinite	Infinite
SR	SR	Slot 1	Parallel 50 Ω	Infinite	Unpopulated	75 or 50 Ω	Unpopulated
		Slot 2	Parallel 50 Ω	75 or 50 Ω	Unpopulated	Infinite	Unpopulated
DR	Empty	Slot 1	Parallel 50 Ω	Infinite	Infinite	Unpopulated	Unpopulated
Empty	DR	Slot 2	Parallel 50 Ω	Unpopulated	Unpopulated	Infinite	Infinite
SR	Empty	Slot 1	Parallel 50 Ω	Infinite	Unpopulated	Unpopulated	Unpopulated
Empty	SR	Slot 2	Parallel 50 Ω	Unpopulated	Unpopulated	Infinite	Unpopulated

Notes to Table 5–1:

(1) For DDR2 at 400 MHz and 533 Mbps = 75 Ω; for DDR2 at 667 MHz and 800 Mbps = 50 Ω

(2) SR = single ranked; DR = dual ranked.

A 54-Ω external parallel termination resistor is placed on all the data strobes and data lines near the Stratix II device on the Stratix II High Speed High Density Board. Although the characteristic impedance of the transmission is designed for 50 Ω, to account for any process variation, it is advisable to underterminate the termination seen at the receiver. This is why the termination resistors at the FPGA side use 54-Ω resistors.

DIMM Configuration

While populating both memory slots is common in a dual-DIMM memory system, there are some instances when only one slot is populated. For example, some systems are designed to have a certain amount of memory initially and as applications get more complex, the system can be easily upgraded to accommodate more memory by populating the second memory slot without re-designing the system. The following section discusses a dual-DIMM system where the dual-DIMM system only has one slot populated at one time and a dual-DIMM system where both slots are populated. ODT controls recommended by the memory vendors listed in Table 5–1 as well as other possible ODT settings will be evaluated for usefulness in an FPGA system.

Dual-DIMM Memory Interface with Slot 1 Populated

This section focuses on a dual-DIMM memory interface where slot 1 is populated and slot 2 is unpopulated. This section examines the impact on the signal quality due to an unpopulated DIMM slot and compares it to a single-DIMM memory interface.

FPGA Writing to Memory

In the DDR2 SDRAM, the ODT feature has two settings: 150 Ω and 75 Ω In Table 5–1, the recommended ODT setting for a dual DIMM configuration with one slot occupied is 150 Ω

On DDR2 SDRAM devices running at 333 MHz/667 Mbps and above, the ODT feature supports an additional setting of 50 Ω

Refer to the respective memory decathlete for additional information about the ODT settings in DDR2 SDRAM devices.

Write to Memory Using an ODT Setting of 150Ω

Figure 5–4 shows a double parallel termination scheme (Class II) using ODT on the memory with a memory-side series resistor when the FPGA is writing to the memory using a 25-Ω OCT drive strength setting on the FPGA.

Figure 5–4. Double Parallel Termination Scheme (Class II) Using ODT on DDR2 SDRAM DIMM with Memory-Side Series Resistor

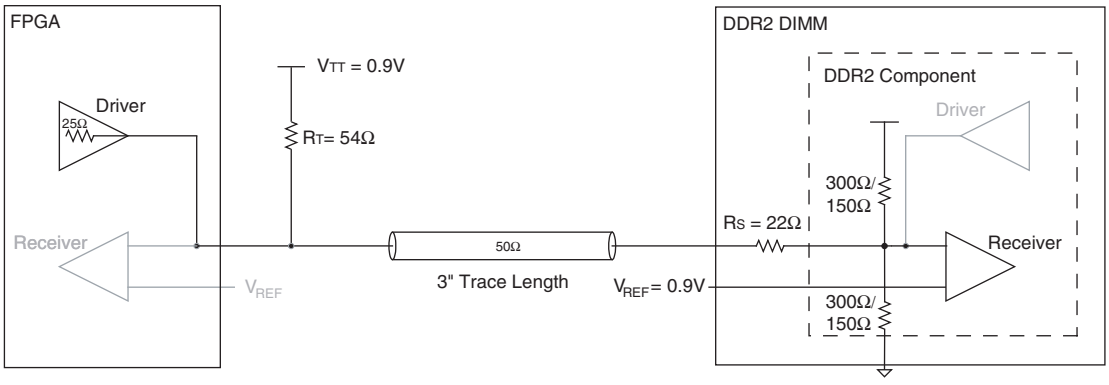


Figure 5–5 shows a HyperLynx simulation and board measurement of a signal at the memory of a double parallel termination using ODT 150 Ω with a memory-side series resistor transmission line when the FPGA is writing to the memory with a 25-Ω OCT drive strength setting.

Figure 5–5. HyperLynx Simulation and Board Measurement of the Signal at the Memory in Slot 1 with Slot 2 Unpopulated

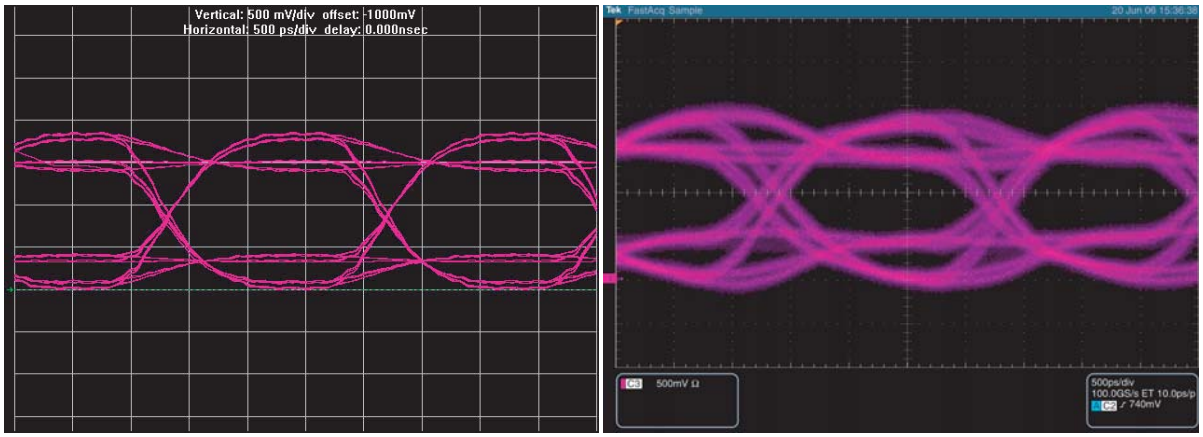


Table 5–3 summarizes the comparison between the simulation and board measurements of the signal at the memory of a single-DIMM and a dual-DIMM memory interface with slot 1 populated using a double parallel termination using an ODT setting of 150 Ω with a memory-side series resistor with a 25-Ω OCT strength setting on the FPGA.

Table 5–3. Comparison of Signal at the Memory of a Single-DIMM and a Dual-DIMM Interface with Slot 1 Populated *(1)*

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Dual-DIMM memory interface with slot 1 populated						
Simulation	1.68	0.97	0.06	NA	2.08	1.96
Measurements	1.30	0.63	0.22	0.20	1.74	1.82
Single-DIMM						
Simulation	1.62	0.94	0.10	0.05	2.46	2.46
Measurements	1.34	0.77	0.04	0.13	1.56	1.39

Note to Table 5–3:

(1) The simulation and board measurements of the single-DIMM DDR2 SDRAM interface are based on the Stratix II Memory Board 2. For more information about the single-DIMM DDR2 SDRAM interface, refer to the *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

Table 5–3 indicates that there is not much difference between a single-DIMM memory interface or a dual-DIMM memory interface with slot 1 populated. The over and undershooting observed in both the simulations and board measurements can be attributed to the use of the ODT setting of 150 Ω on the memory resulting in over-termination at the receiver. In addition, there is no significant effect of the extra DIMM connector due to the unpopulated slot.

When the ODT setting is set to 75 Ω, there is no difference in the eye width and height compared to the ODT setting of 150 Ω However, there is no overshoot and undershoot when the ODT setting is set to 75 Ω, which is attributed to proper termination resulting in matched impedance seen by the DDR2 SDRAM devices.

For information about results obtained from using an ODT setting of 75 Ω refer to page 5–24.

Reading from Memory

During read from the memory, the ODT feature is turned off. Thus, there is no difference between using an ODT setting of 150 Ω and 75 Ω As such, the termination scheme becomes a single parallel termination scheme (Class I) where there is an external resistor on the FPGA side and a series resistor on the memory side as shown in Figure 5–6.

Figure 5–6. Single Parallel Termination Scheme (Class I) Using External Resistor and Memory-Side Series Resistor

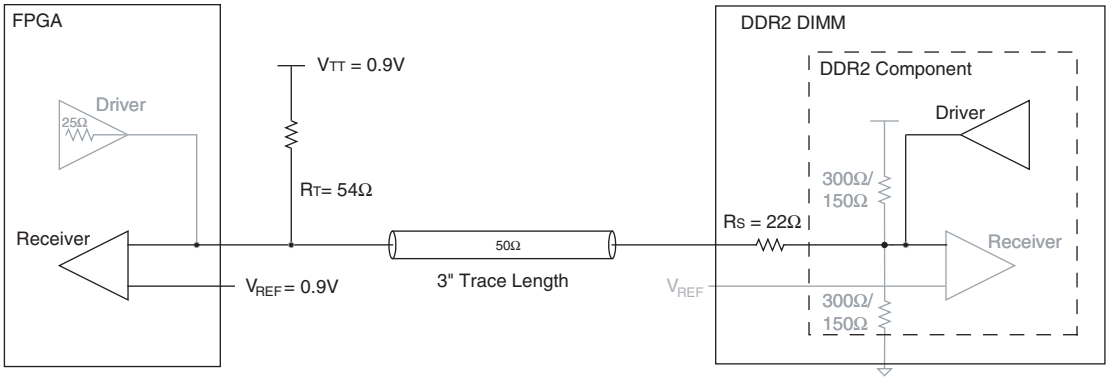


Figure 5–7 shows the simulation and board measurement of the signal at the FPGA of a single parallel termination using an external parallel resistor on the FPGA side with a memory-side series resistor with full drive strength setting on the memory.

Figure 5–7. HyperLynx Simulation and Board Measurement of the Signal at the FPGA When Reading From Slot 1 With Slot 2 Unpopulated

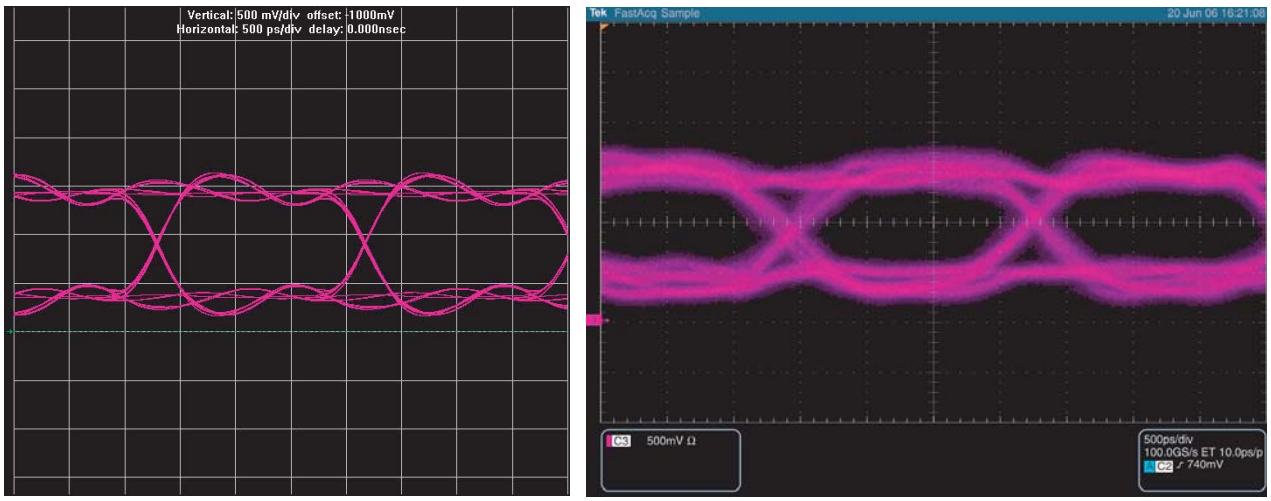


Table 5–4 summarizes the comparison between the simulation and board measurements of the signal seen at the FPGA of a single-DIMM and a dual-DIMM memory interface with a slot 1 populated memory interface using a single parallel termination using an external parallel resistor at the FPGA with a memory-side series resistor with full strength setting on the memory.

Table 5–4. Comparison of Signal at the FPGA of a Dual-DIMM Memory Interface with Slot 1 Populated *(1)*

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Dual-DIMM memory interface with slot 1 populated						
Simulation	1.76	0.80	NA	NA	2.29	2.29
Measurements	1.08	0.59	NA	NA	1.14	1.59
Single-DIMM[1]						
Simulation	1.80	0.95	NA	NA	2.67	2.46
Measurements	1.03	0.58	NA	NA	1.10	1.30

Note to Table 5–4:

(1) The simulation and board measurements of the single-DIMM DDR2 SDRAM interface are based on the Stratix II Memory Board 2. For more information about the single-DIMM DDR2 SDRAM interface, refer to the *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

Table 5–4 demonstrates that there is not much difference between a single-DIMM memory interface or a dual-DIMM memory interface with only slot 1 populated. There is no significant effect of the extra DIMM connector due to the unpopulated slot.

Dual-DIMM with Slot 2 Populated

This section focuses on a dual-DIMM memory interface where slot 2 is populated and slot 1 is unpopulated. Specifically, this section discusses the impact of location of the DIMM on the signal quality.

FPGA Writing to Memory

The previous section focused on the dual-DIMM memory interface where slot 1 is populated resulting in the memory being located closer to the FPGA. When slot 2 is populated, the memory is located further away from the FPGA, resulting in additional trace length that potentially affects the signal quality seen by the memory. The next section explores if there are any differences between populating slot 1 and slot 2 of the dual-DIMM memory interface.

Write to Memory Using an ODT Setting of 150Ω

Figure 5–8 shows the double parallel termination scheme (Class II) using ODT on the memory with the memory-side series resistor when the FPGA is writing to the memory using a 25-Ω OCT drive strength setting on the FPGA.

Figure 5–8. Double Parallel Termination Scheme (Class II) Using ODT on DDR2 SDRAM DIMM with Memory-side Series Resistor

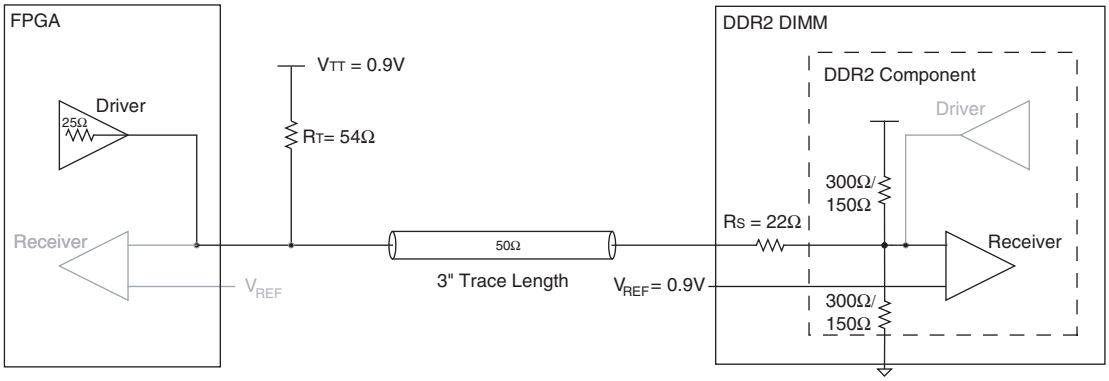


Figure 5–9 shows the simulation and board measurement of the signal at the memory of a double parallel termination using an ODT setting of 150 Ω with a memory-side series resistor transmission line when the FPGA is writing to the memory with a 25-Ω OCT drive strength setting.

Figure 5–9. HyperLynx Simulation and Board Measurement of the Signal at the Memory in Slot 2 With Slot 1 Unpopulated

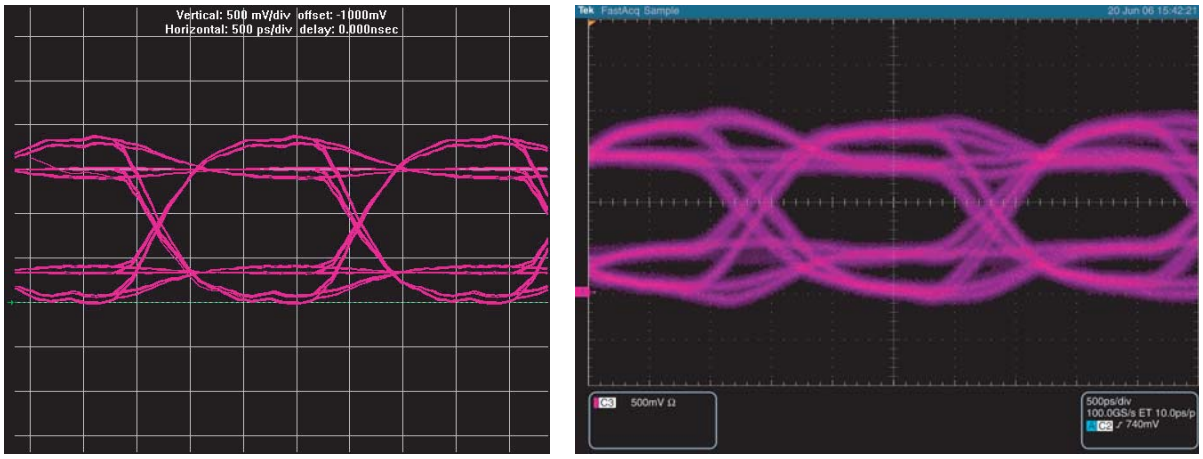


Table 5–5 summarizes the comparison between the simulation and board measurements of the signal seen at the DDR2 SDRAM DIMM of a dual-DIMM memory interface with either only slot 1 populated or only slot 2 populated using a double parallel termination using an ODT setting of 150 Ω with a memory-side series resistor with a 25-Ω OCT strength setting on the FPGA.

Table 5–5. Comparison of Signal at the Memory of a Dual-DIMM Interface with Either Only Slot 1 Populated or Only Slot 2 Populated

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Dual-DIMM Memory Interface with Slot 2 Populated						
Simulation	1.69	0.94	0.07	0.02	1.96	2.08
Measurements	1.28	0.68	0.24	0.20	1.60	1.60
Dual-DIMM Memory Interface with Slot 1 Populated						
Simulation	1.68	0.97	0.06	NA	2.08	2.08
Measurements	1.30	0.63	0.22	0.20	1.74	1.82

Table 5–5 shows that there is not much difference between populating slot 1 or slot 2 in a dual-DIMM memory interface. The over and undershooting observed in both the simulations and board measurements can be attributed to the use of the ODT setting of 150 Ω on the memory, resulting in under-termination at the receiver.

When the ODT setting is set to 75 Ω, there is no difference in the eye width and height compared to the ODT setting of 150 Ω However, there is no overshoot and undershoot when the ODT setting is set to 75 Ω, which is attributed to proper termination resulting in matched impedance seen by the DDR2 SDRAM devices.

For detailed results for the ODT setting of 75 Ω, refer to page 5–25.

Reading from Memory

During read from memory, the ODT feature is turned off, thus there is no difference between using an ODT setting of 150 Ω and 75 Ω As such, the termination scheme becomes a single parallel termination scheme (Class I) where there is an external resistor on the FPGA side and a series resistor on the memory side, as shown in Figure 5–10.

Figure 5–10. Single Parallel Termination Scheme (Class I) Using External Resistor and Memory-Side Series Resistor

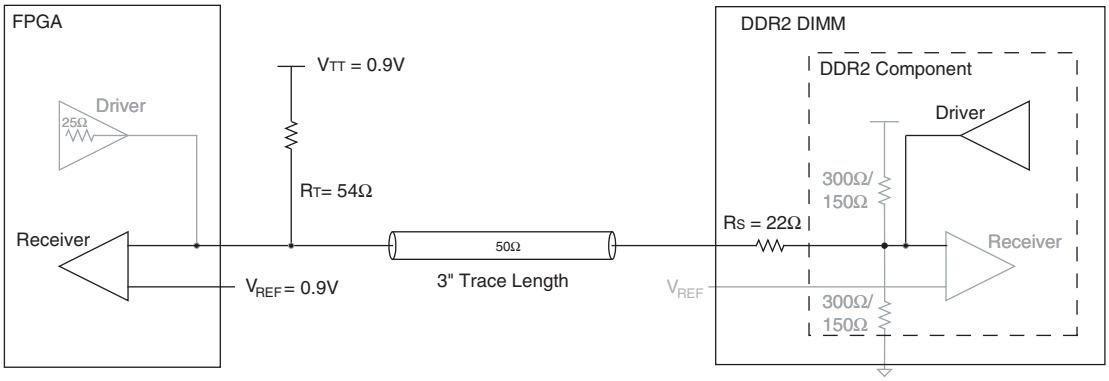


Figure 5–11 shows the simulation and board measurement of the signal at the FPGA of a single parallel termination using an external parallel resistor on the FPGA side with a memory-side series resistor with full drive strength setting on the memory.

Figure 5–11. HyperLynx Simulation and Board Measurement of the Signal at the FPGA When Reading From Slot 2 With Slot 1 Unpopulated

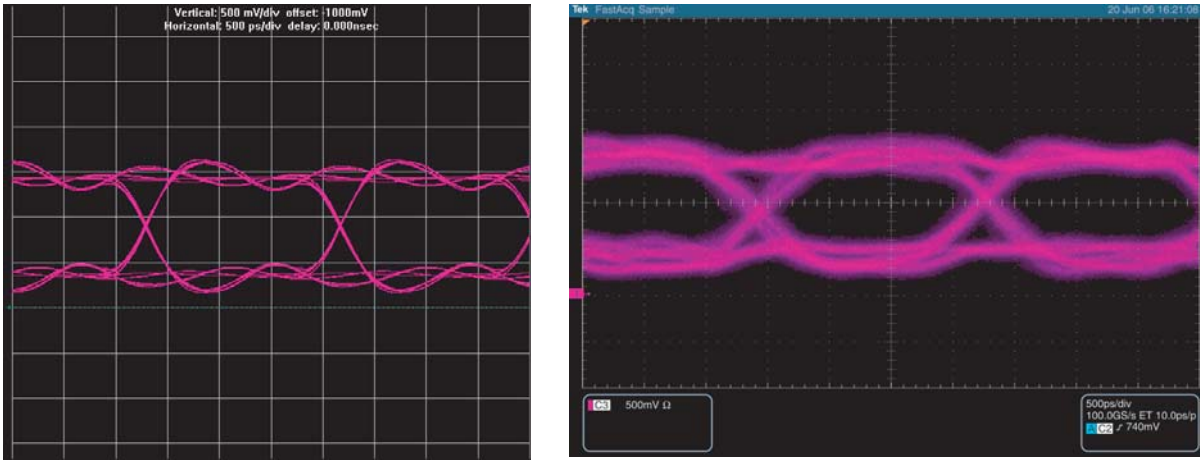


Table 5–6 summarizes the comparison between the simulation and board measurements of the signal seen at the FPGA of a dual-DIMM memory interface with either slot 1 or slot 2 populated using a single parallel termination using an external parallel resistor at the FPGA with a memory-side series resistor with full strength setting on the memory.

Table 5–6. Comparison of the Signal at the FPGA of a Dual-DIMM Memory Interface with Either Slot 1 or Slot 2 Populated

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Slot 2 Populated						
Simulation	1.80	0.80	NA	NA	3.09	2.57
Measurements	1.17	0.66	NA	NA	1.25	1.54
Slot 1 Populated						
Simulation	1.80	0.95	NA	NA	2.67	2.46
Measurements	1.08	0.59	NA	NA	1.14	1.59

From Table 5–6, you can see the signal seen at the FPGA is similar whether the memory DIMM is located at either slot 1 or slot 2.

Dual-DIMM Memory Interface with Both Slot 1 and Slot 2 Populated

This section focuses on a dual-DIMM memory interface where both slot 1 and slot 2 are populated. As such, you can write to either the memory in slot 1 or the memory in slot 2.

FPGA Writing to Memory

In Table 5–1, the recommended ODT setting for a dual DIMM configuration with both slots occupied is 75 Ω Since there is an option for an ODT setting of 150 Ω, this section explores the usage of the 150 Ω setting and compares the results to that of the recommended 75 Ω

Write to Memory in Slot 1 Using an ODT Setting of 75-Ω

Figure 5–12 shows the double parallel termination scheme (Class II) using ODT on the memory with the memory-side series resistor when the FPGA is writing to the memory using a 25-Ω OCT drive strength setting on the FPGA. In this scenario, the FPGA is writing to the memory in slot 1 and the ODT feature of the memory at slot 2 is turned on.

Figure 5–12. Double Parallel Termination Scheme (Class II) Using ODT on DDR2 SDRAM DIMM with a Memory-Side Series Resistor

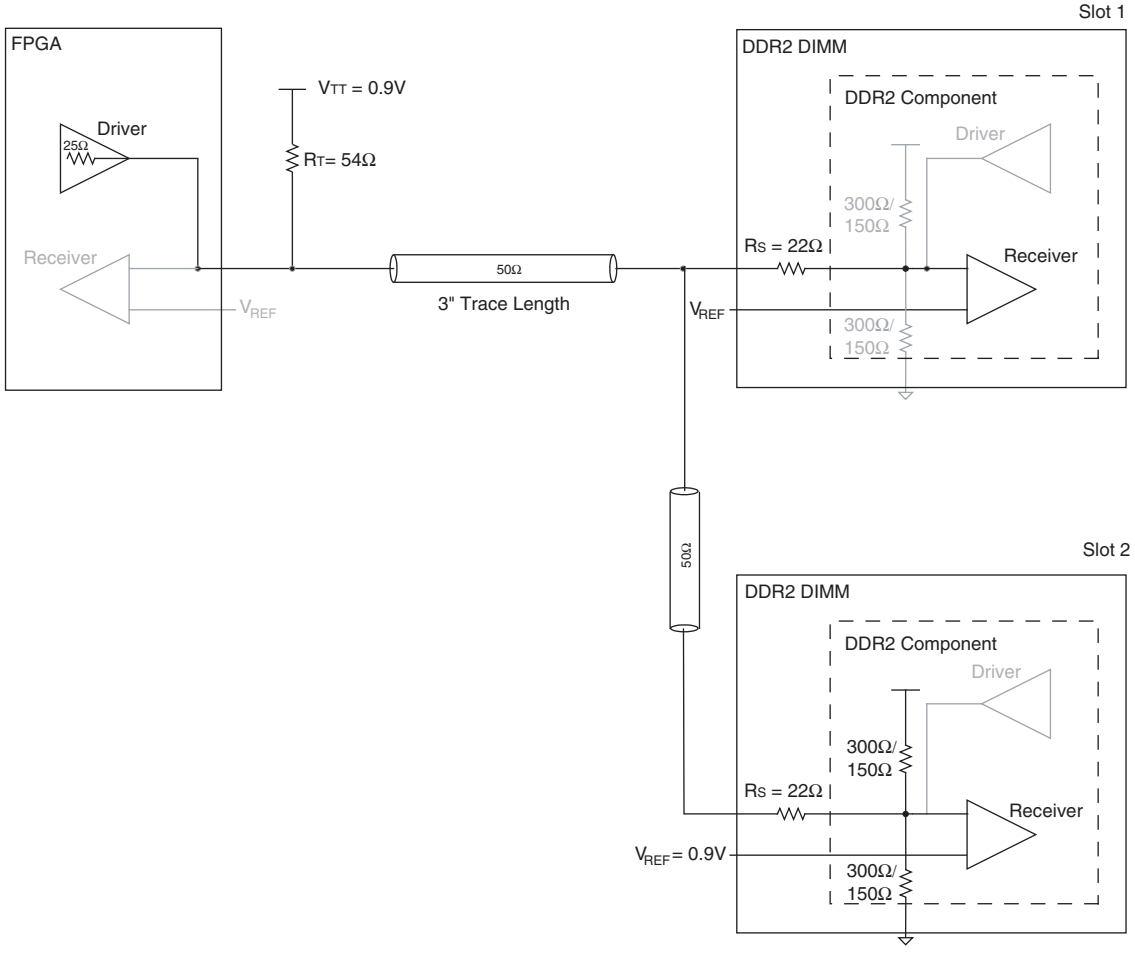


Figure 5–13 shows a HyperLynx simulation and board measurement of the signal at the memory in slot 1 of a double parallel termination using an ODT setting of 75 Ω with a memory-side series resistor transmission line when the FPGA is writing to the memory with a 25-Ω OCT drive strength setting.

Figure 5–13. HyperLynx Simulation and Board Measurements of the Signal at the Memory in Slot 1 with Both Slots Populated

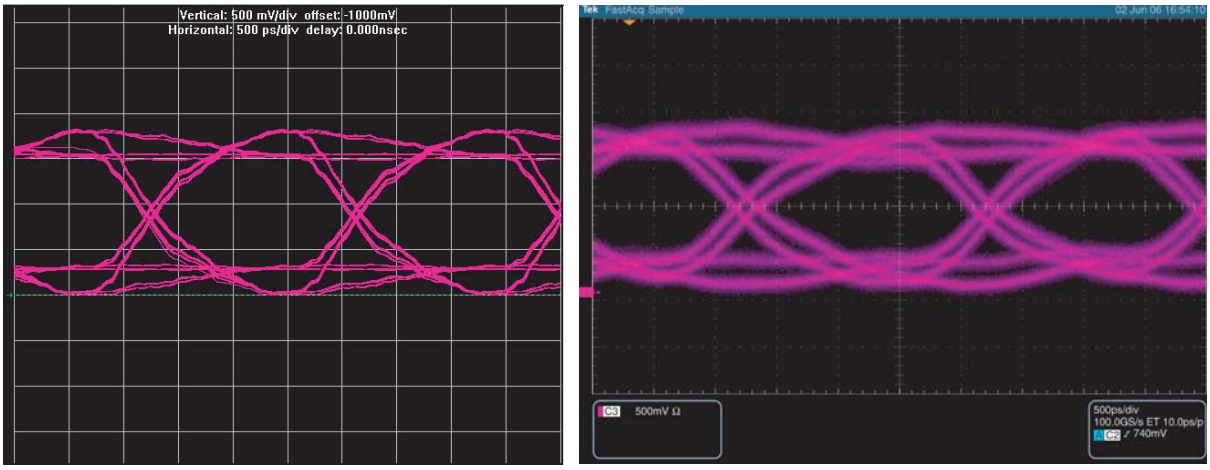


Table 5–7 summarizes the comparison of the signal at the memory of a dual-DIMM memory interface with one slot and with both slots populated using a double parallel termination using an ODT setting of 75 Ω with a memory-side series resistor with a 25-Ω OCT strength setting on the FPGA.

Table 5–7. Comparison of the Signal at the Memory of a Dual-DIMM Interface With One Slot and With Both Slots Populated

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Dual-DIMM Interface with Both Slots Populated Writing to Slot 1						
Simulation	1.60	1.18	0.02	NA	1.71	1.71
Measurements	0.97	0.77	0.05	0.04	1.25	1.25
Dual-DIMM Interface with Slot 1 Populated						
Simulation	1.68	0.97	0.06	NA	2.08	2.08
Measurements	1.30	0.63	0.22	0.20	1.74	1.82

Table 5–7 shows that there is not much difference in the eye height between populating one slot or both slots. However, the additional loading due to the additional memory DIMM results in a slower edge rate, which results in smaller eye width and degrades the setup and hold time of the memory. This reduces the available data valid window.

When the ODT setting is set to 150 Ω, there is no difference in the eye width and height compared to the ODT setting of 75 Ω However, there is some overshoot and undershoot when the ODT setting is set to 150 Ω, which is attributed to under termination resulting in mismatched impedance seen by the DDR2 SDRAM devices.

For more information about the results obtained from using an ODT setting of 150 Ω, refer to page 5–26.

Write to Memory in Slot 2 Using an ODT Setting of 75-Ω

In this scenario, the FPGA is writing to the memory in slot 2 and the ODT feature of the memory at slot 1 is turned on. Figure 5–14 shows the HyperLynx simulation and board measurement of the signal at the memory in slot 1 of a double parallel termination using an ODT setting of 75 Ω with a memory-side series resistor transmission line when the FPGA is writing to the memory with a 25-Ω OCT drive strength setting.

Figure 5–14. HyperLynx Simulation and Board Measurements of the Signal at the Memory in Slot 2 With Both Slots Populated

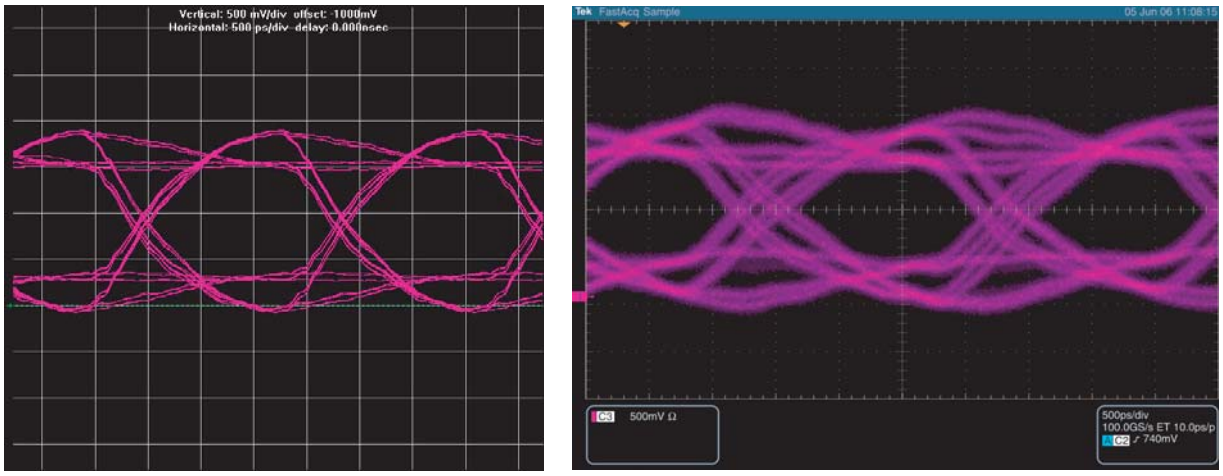


Table 5–8 summarizes the comparison of the signal at the memory of a dual-DIMM memory interface with slot 1 populated using a double parallel termination using an ODT setting of 75 Ω with a memory-side series resistor with a 25-Ω OCT strength setting on the FPGA.

Table 5–8. Comparison of the Signal at the Memory of a Dual-DIMM Interface With Both Slots Populated

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rise Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Dual-DIMM Interface with Both Slots Populated Writing to Slot 2						
Simulation	1.60	1.16	0.10	0.08	1.68	1.60
Measurements	1.10	0.85	0.16	0.19	1.11	1.25
Dual-DIMM Interface with Both Slots Populated Writing to Slot 1						
Simulation	1.60	1.18	0.02	NA	1.71	1.71
Measurements	1.30	0.77	0.05	0.04	1.25	1.25

From Table 5–8, you can see that both simulations and board measurements demonstrate that the eye width is larger when writing to slot 1, which is due to better edge rate seen when writing to slot 1. The improvement on the eye when writing to slot 1 can be attributed to the location of the termination. When you are writing to slot 1, the ODT feature of slot 2 is turned on, resulting in a fly-by topology. When you are writing to slot 2, the ODT feature of slot 1 is turned on resulting in a non fly-by topology.

When the ODT setting is set to 150 Ω, there is no difference in the eye width and height compared to the ODT setting of 75 Ω However, there is some overshoot and undershoot when the ODT setting is set to 150 Ω, which is attributed to under termination resulting in mismatched impedance seen by the DDR2 SDRAM devices.

For more information about the results obtained from using an ODT setting of 150 Ω, refer to "Write to Memory in Slot 2 Using an ODT Setting of 150 Ω With Both Slots Populated" on page 5–27.

Reading From Memory

In Table 5–2, the recommended ODT setting for a dual-DIMM configuration with both slots occupied is to turn on the ODT feature using a setting of 75 Ω on the slot that is not read from. As there is an option for an ODT setting of 150 Ω, this section explores the usage of the 150 Ω setting and compares the results to that of the recommended 75 Ω.

Read From Memory in Slot 1 Using an ODT Setting of 75-Ω on Slot 2

Figure 5–15 shows the double parallel termination scheme (Class II) using ODT on the memory with the memory-side series resistor when the FPGA is reading from the memory using a full drive strength setting on the memory. In this scenario, the FPGA is reading from the memory in slot 1 and the ODT feature of the memory at slot 2 is turned on.

Figure 5–15. Double Parallel Termination Scheme (Class II) Using External Resistor and Memory-Side Series Resistor and ODT Feature Turned On

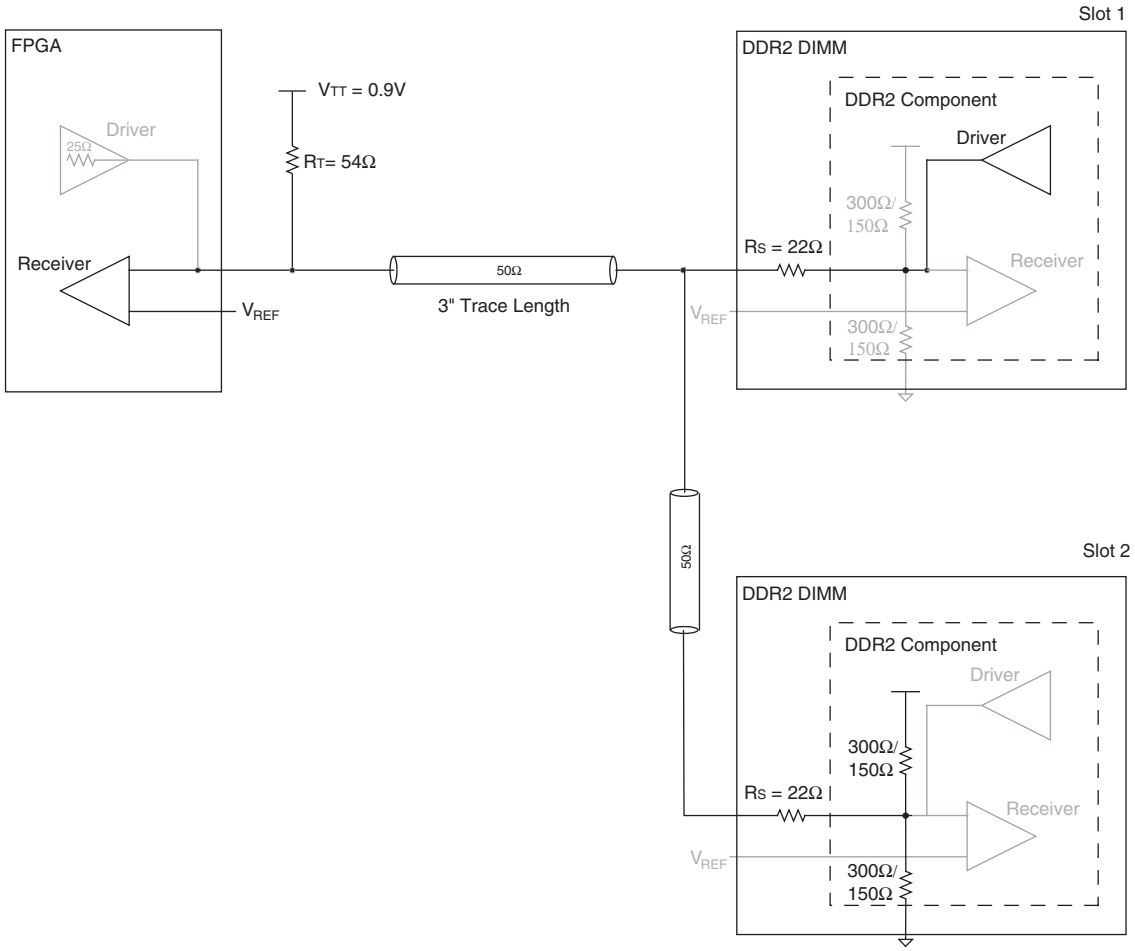
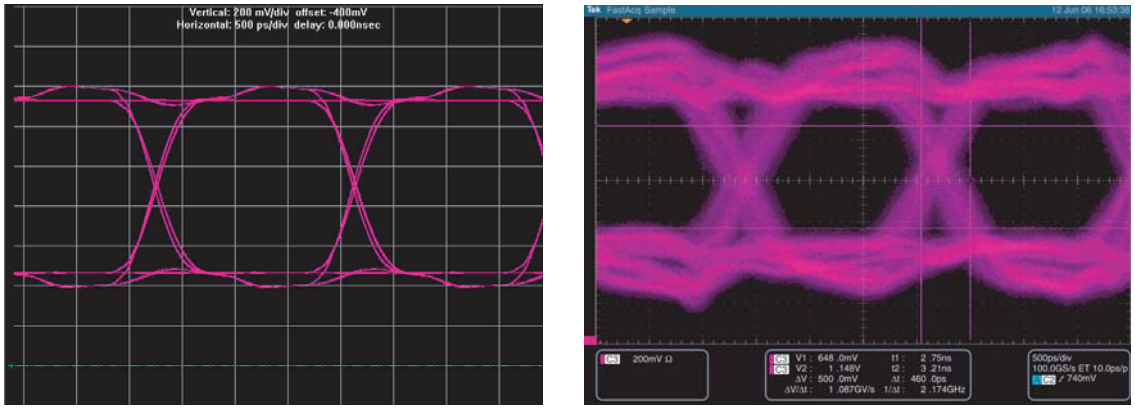


Figure 5–16 shows the simulation and board measurement of the signal at the FPGA when the FPGA is reading from the memory in slot 1 using a full drive strength setting on the memory.

Figure 5–16. HyperLynx Simulation and Board Measurement of the Signal at the FPGA When Reading From Slot 1 With Both Slots Populated *(1)*



Note to Figure 5–16:

(1) The vertical scale used for the simulation and measurement is set to 200 mV per division.

Table 5–9 summarizes the comparison between the simulation and board measurements of the signal seen at the FPGA of a dual-DIMM memory interface with both slots populated and a dual-DIMM memory interface with a slot 1 populated memory interface.

Table 5–9. Comparison of the Signal at the FPGA of a Dual-DIMM Interface Reading From Slot 1 With One Slot and With Both Slots Populated

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Dual-DIMM with One Slot Populated with an ODT Setting of 75-Ω on Slot 2						
Simulation	1.74	0.87	NA	NA	1.91	1.88
Measurements	0.86	0.58	NA	NA	1.11	1.09
Dual-DIMM with One Slot Populated in Slot 1 without ODT Setting						
Simulation	1.76	0.80	NA	NA	2.29	2.29
Measurements	1.08	0.59	NA	NA	1.14	1.59

Table 5–9 shows that when both slots are populated, the additional loading due to the additional memory DIMM results in a slower edge rate, which results in a degradation in the eye width.

For more information about the results obtained from using an ODT setting of 150 Ω, refer to "Read from Memory in Slot 1 Using an ODT Setting of 150 Ω on Slot 2 with Both Slots Populated" on page 5–28.

Read From Memory in Slot 2 Using an ODT Setting of 75-Ω on Slot 1

In this scenario, the FPGA is reading from the memory in slot 2 and the ODT feature of the memory at slot 1 is turned on.

Figure 5–17. Double Parallel Termination Scheme (Class II) Using External Resistor and a Memory-Side Series Resistor and ODT Feature Turned On

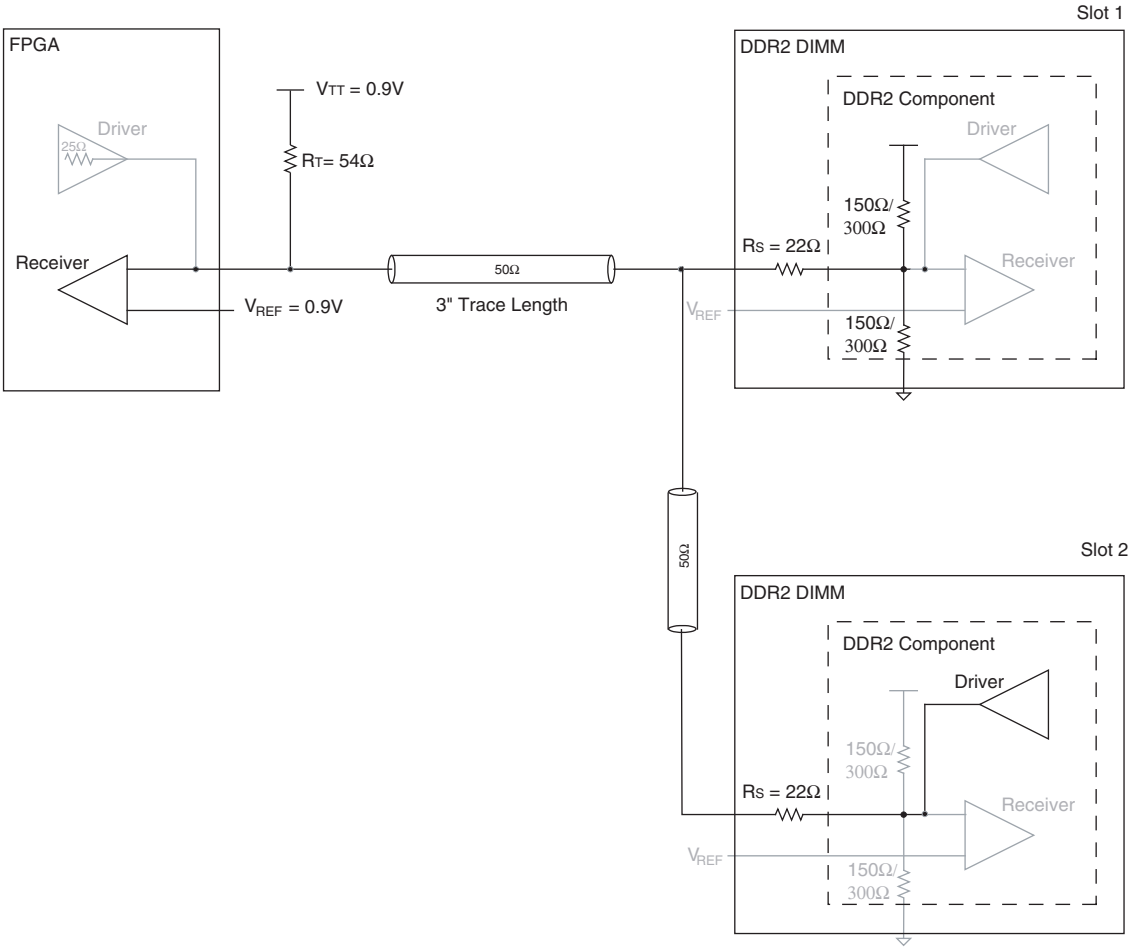
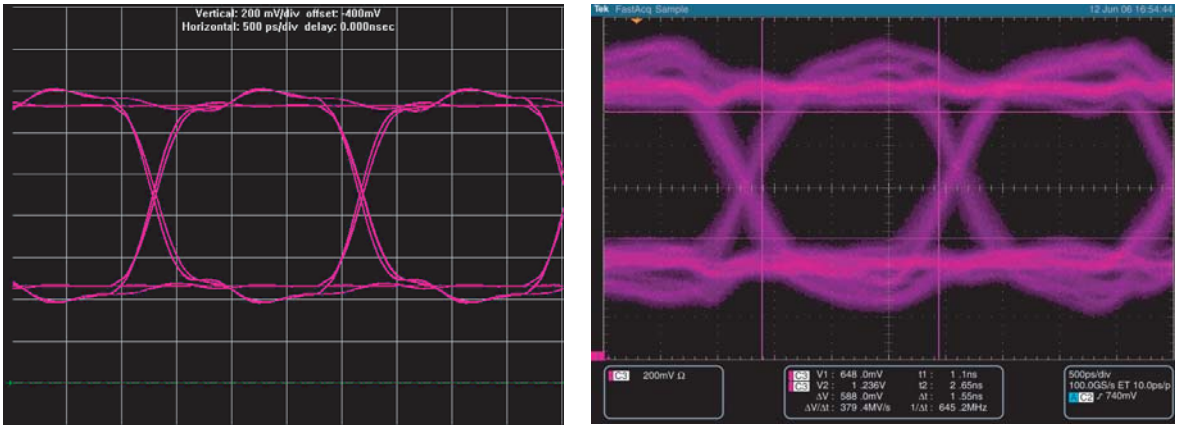


Figure 5–18 shows the HyperLynx simulation and board measurement of the signal at the FPGA of a double parallel termination using an external parallel resistor on the FPGA side with a memory-side series resistor and an ODT setting of 75 Ω with a full drive strength setting on the memory.

Figure 5–18. HyperLynx Simulation and Board Measurements of the Signal at the FPGA When Reading From Slot 2 With Both Slots Populated *(1)*



Note to Figure 5–18:

(1) The vertical scale used for the simulation and measurement is set to 200 mV per division.

Table 5–10 summarizes the comparison between the simulation and board measurements of the signal seen at the FPGA of a dual-DIMM memory interface with both slots populated and a dual-DIMM memory interface with a slot 1 populated memory interface.

Table 5–10. Comparison of the Signal at the FPGA of a Dual-DIMM Interface Reading From Slot 2 With One Slot and With Both Slots Populated

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
Dual-DIMM with Both Slots Populated with an ODT Setting of 75-Ω Setting on Slot 1						
Simulation	1.70	0.81	NA	NA	1.72	1.99
Measurements	0.87	0.59	NA	NA	1.09	1.14
Dual-DIMM with One Slot Populated in Slot 2 without an ODT Setting						
Simulation	1.80	0.80	NA	NA	3.09	2.57
Measurements	1.17	0.66	NA	NA	1.25	1.54

Table 5–10 shows that when only one slot is populated in a dual-DIMM memory interface, the eye width is larger as compared to a dual-DIMM memory interface with both slots populated. This can be attributed to the loading from the DIMM located in slot 1.

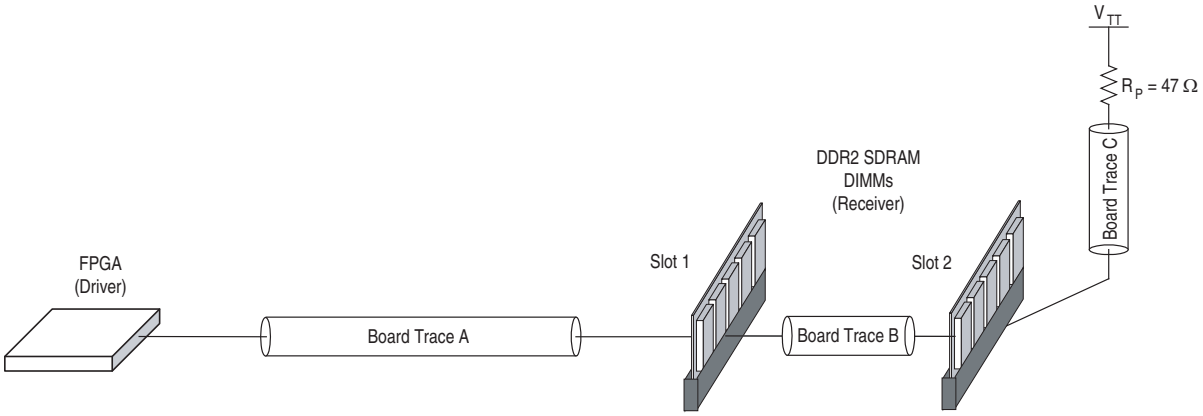
When the ODT setting is set to 150 Ω, there is no difference in the signal quality compared to the ODT setting of 75 Ω

For more information about the results obtained from using an ODT setting of 150 Ω, refer to "Read From Memory in Slot 2 Using an ODT Setting of 150 Ωon Slot 1 With Both Slots Populated" on page 5–29.

Dual-DIMM DDR2 Clock, Address, and Command Termination and Topology

The address and command signals on a DDR2 SDRAM interface are unidirectional signals that the FPGA memory controller drives to the DIMM slots. These signals are always Class-I terminated at the memory end of the line (Figure 5–19). Always place DDR2 SDRAM address and command Class-I termination after the last DIMM. The interface can have one or two DIMMs, but never more than two DIMMs total.

Figure 5–19. Multi DIMM DDR2 Address and Command Termination Topology



In Figure 5–19, observe the following points:

- Board trace A = 1.9 to 4.5 inches (48 to 115 mm)
- Board trace B = 0.425 inches (10.795 mm)
- Board trace C = 0.2 to 0.55 inches (5 to 13 mm)
- Total of board trace A + B + C = 2.5 to 5 inches (63 to 127 mm)
- R_P = 36 to 56 Ω
- Length match all address and command signals to +250 mils (+5 mm) or +/– 50 ps of memory clock length at the DIMM.

You may place a compensation capacitor directly before the first DIMM slot 1 to improve signal quality on the address and command signal group. If you fit a capacitor, Altera recommends a value of 24 pF.

For more information, refer to *Micron TN47-01*.

Address and Command Signals

The address and command group of signals: bank address, address, RAS#, CAS#, and WE#, operate a different toggle rate depending on whether a you implement a full-rate or half-rate memory controller.

In full-rate designs, the address and command group of signals are 1T signals, which means that the signals can change every memory clock cycle. Address and command signals are also single data rate (SDR). Hence in a full-rate PHY design, the address and command signals operate at a maximum frequency of 0.5 × the data rate. For example in a 266-MHz full rate design, the maximum address and command frequency is 133 MHz.

In half-rate designs the address and command group of signals are 2T signals, which means that the signals change only every two memory clock cycles. As the signals are also SDR, in a half-rate PHY design, the address and command signals operate at a maximum frequency of 0.25 × the data rate. For example, in a 400-MHz half-rate design, the maximum address and command frequency is 100 MHz.

Control Group Signals

The control group of signals: chip select CS#, clock enable CKE, and ODT are always 1T regardless of whether you implement a full-rate or half-rate design. As the signals are also SDR, the control group signals operate at a maximum frequency of 0.5 × the data rate. For example, in a 400-MHz design, the maximum control group frequency is 200 MHz.

Clock Group Signals

Depending on the specific form factor, DDR2 SDRAM DIMMs have two or three differential clock pairs, to ensure that the loading on the clock signals is not excessive. The clock signals are always terminated on the DIMMs and hence no termination is required on your PCB. Additionally, each DIMM slot is required to have its own dedicated set of clock signals. Hence clock signals are always point-to-point from the FPGA PHY to each individual DIMM slot. Individual memory clock signals should never be shared between two DIMM slots.

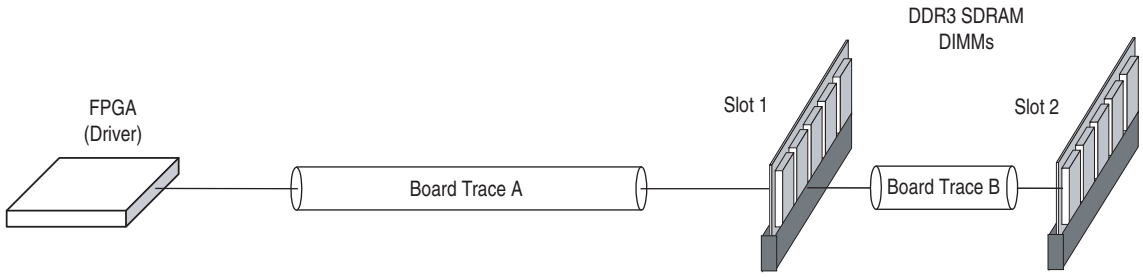
A typical two slot DDR2 DIMM design therefore has six differential memory clock pairs—three to the first DIMM and three to the second DIMM. All six memory clock pairs must be delay matched to each other to ±25 mils (±0.635 mm) and ±10 mils (±0.254 mm) for each CLK to CLK# signal.

You may place a compensation capacitor between each clock pair directly before the DIMM connector, to improve the clock slew rates. As FPGA devices have fully programmable drive strength and slew rate options, this capacitor is usually not required for FPGA design. However, Altera advise that you simulate your specific implementation to ascertain if this capacitor is required or not. If fitted the best value is typically 5 pF.

DDR3 SDRAM

This section details the system implementation of a dual slot unbuffered DDR3 SDRAM interface, operating at up to 400 MHz and 800 Mbps data rates. Figure 5–20 shows a typical DQS, DQ, and DM, and address and command signal topology for a dual-DIMM interface configuration, using the ODT feature of the DDR3 SDRAM components combined with the dynamic OCT features available in Stratix III and Stratix IV devices.

Figure 5–20. Multi DIMM DDR3 DQS, DQ, and DM, and Address and Command Termination Topology



In Figure 5–20, observe the following points:

- Board trace A = 1.9 to 4.5 inches (48 to 115 mm)
- Board trace B = 0.425 inches (10.795 mm)
- This topology to both DIMMs is accurate for DQS, DQ, and DM, and address and command signals
- This topology is not correct for CLK and CLK# and control group signals (CS#, CKE, and ODT), which are always point-to-point single rank only.

Comparison of DDR3 and DDR2 DQ and DQS ODT Features and Topology

DDR3 and DDR2 SDRAM systems are quite similar. The physical topology of the data group of signals may be considered nearly identical. The FPGA end (driver) I/O standard changes from SSTL18 for DDR2 to SSTL15 for DDR3, but all other OCT settings are identical. DDR3 offers enhanced ODT options for termination and drive-strength settings at the memory end of the line.

For more information, refer to the DDR3 SDRAM ODT matrix for writes and the DDR3 SDRAM ODT matrix for reads tables in the *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

Dual-DIMM DDR3 Clock, Address, and Command Termination and Topology

One significant difference between DDR3 and DDR2 DIMM based interfaces is the address, command and clock signals. DDR3 uses a daisy chained based architecture when using JEDEC standard modules. The address, command, and clock signals are routed on each module in a daisy chain and feature a fly-by termination on the module. Impedance matching is required to make the dual-DIMM topology work effectively—40 to 50 Ω traces should be targeted on the main board.

Address and Command Signals

Two UDIMMs result in twice the effective load on the address and command signals, which reduces the slew rate and makes it more difficult to meet setup and hold timing (t_{IS} and t_{IH}). However, address and command signals operate at half the interface rate and are SDR. Hence a 400-Mbps data rate equates to an address and command fundamental frequency of 100 MHz.

Control Group Signals

The control group signals (chip Select CS#, clock enable CKE, and ODT) are only ever single rank. A dual-rank capable DDR3 DIMM slot has two copies of each signal, and a dual-DIMM slot interface has four copies of each signal. Hence the signal quality of these signals is identical to a single rank case. The control group of signals, are always 1T regardless of whether you implement a full-rate or half-rate design. As the signals are also SDR, the control group signals operate at a maximum frequency of 0.5 × the data rate. For example, in a 400 MHz design, the maximum control group frequency is 200 MHz.

Clock Group Signals

Like the control group signals, the clock signals in DDR3 SDRAM are only ever single rank loaded. A dual-rank capable DDR3 DIMM slot has two copies of the signal, and a dual-slot interface has four copies of the `mem_clk` and `mem_clk_n` signals.

For more information about a DDR3 two-DIMM system design, refer to Micron *TN-41-08: DDR3 Design Guide for Two-DIMM Systems*.

The Altera DDR3 ALTMEMPHY megafunction does not support the 1T address and command topology referred to in this Micron Technical Note—only 2T implementations are supported.

Write to Memory in Slot 1 Using an ODT Setting of 75 Ω With One Slot Populated

Figure 5–21 shows the simulation and board measurement of the signal at the memory when the FPGA is writing to the memory with an ODT setting of 75 Ω and using a 25-Ω OCT drive strength setting on the FPGA.

Figure 5–21. HyperLynx Simulation and Board Measurement of the Signal at the Memory in Slot 1 With Slot 2 Unpopulated

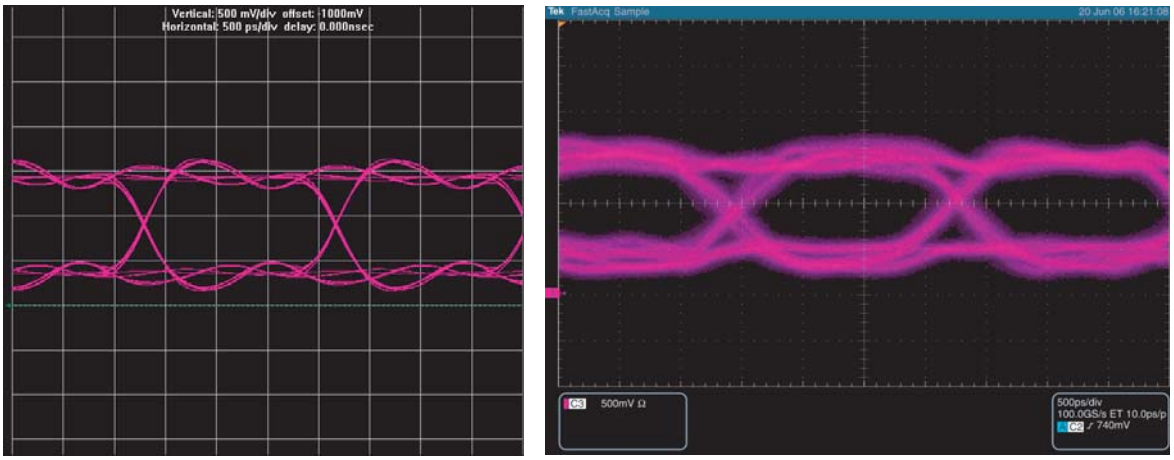


Table 5–11 summarizes the comparison between the simulation and board measurements of the signal seen at the DDR2 SDRAM of a dual-DIMM with slot 1 populated by a memory interface using a different ODT setting.

Table 5–11. Comparison of the Signal at the Memory of a Dual-DIMM Interface With Only Slot 1 Populated and a Different ODT Setting

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
ODT Setting of 75 Ω						
Simulation	1.68	0.91	NA	NA	1.88	1.88
Measurements	1.28	0.57	NA	NA	1.54	1.38
ODT Setting of 150 Ω						
Simulation	1.68	0.97	0.06	NA	2.67	2.13
Measurements	1.30	0.63	0.22	0.20	1.74	1.82

Write to Memory in Slot 2 Using an ODT Setting of 75 Ω With One Slot Populated

Figure 5–22 shows the simulation and measurements result of the signal seen at the memory when the FPGA is writing to the memory with an ODT setting of 75 Ω and using a 25-Ω OCT drive strength setting on the FPGA.

Figure 5–22. HyperLynx Simulation and Board Measurement of the Signal at the Memory in Slot 2 with Slot 1 Unpopulated

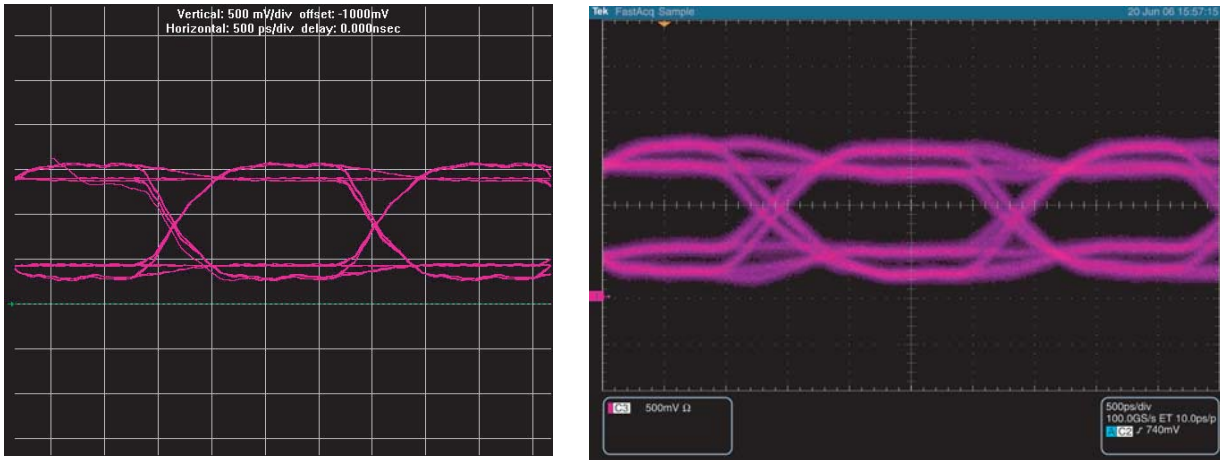


Table 5–12 summarizes the comparison of the signal at the memory of a dual-DIMM memory interface with either slot 1 or slot 2 populated using a double parallel termination using an ODT setting of 75 Ω with a memory-side series resistor with a 25-Ω OCT strength setting on the FPGA.

Table 5–12. Comparison of Signal at the Memory of a Dual-DIMM Interface With Only Slot 2 Populated and a Different ODT Setting

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
ODT Setting of 75 Ω						
Simulation	1.68	0.89	NA	NA	1.82	1.93
Measurements	1.29	0.59	NA	NA	1.60	1.29
ODT Setting of 150 Ω						
Simulation	1.69	0.94	0.07	0.02	1.88	2.29
Measurements	1.28	0.68	0.24	0.20	1.60	1.60

Write to Memory in Slot 1 Using an ODT Setting of 150 Ω With Both Slots Populated

Figure 5–23 shows the HyperLynx simulation and board measurement of the signal at the memory in slot 1 of a double parallel termination using an ODT setting of 150 Ω on Slot 2 with a memory-side series resistor transmission line when the FPGA is writing to the memory with a 25-Ω OCT drive strength setting.

Figure 5–23. HyperLynx Simulation and Board Measurement of the Signal at the Memory in Slot 1 With Both Slots Populated

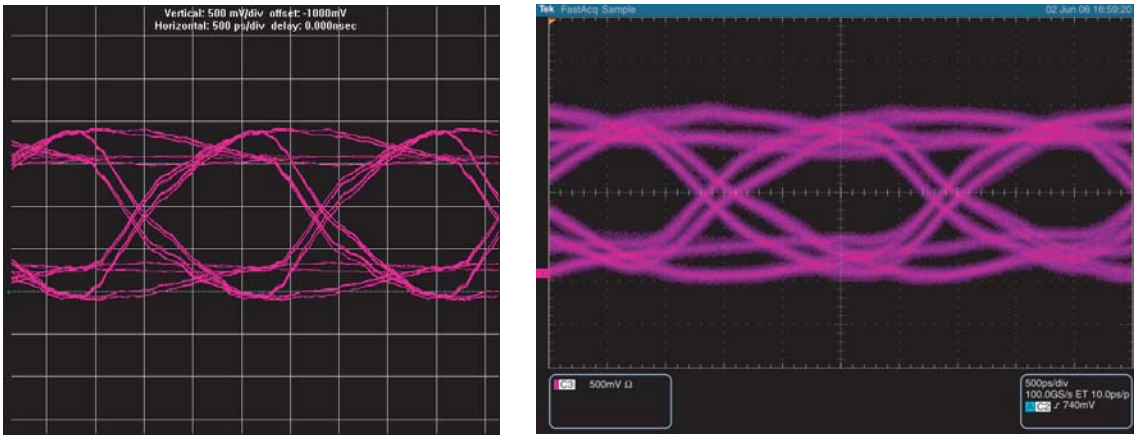


Table 5–13 summarizes the comparison between the simulation and board measurements of the signal seen at the memory in slot 1 of a dual-DIMM memory interface with both slots populated using a double parallel termination using a different ODT setting on Slot 2 with a memory-side series resistor with a 25-Ω OCT strength setting on the FPGA.

Table 5–13. Comparison of Signal at the Memory of a Dual-DIMM Interface with Both Slots Populated and a Different ODT Setting on Slot 2

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
ODT Setting of 150 Ω						
Simulation	1.60	1.18	0.02	NA	1.71	1.71
Measurements	0.89	0.78	0.13	0.17	1.19	1.32
ODT Setting of 75 Ω						
Simulation	1.60	1.18	0.02	NA	1.71	1.71
Measurements	0.97	0.77	0.05	0.04	1.25	1.25

Write to Memory in Slot 2 Using an ODT Setting of 150 Ω With Both Slots Populated

Figure 5–24 shows the HyperLynx simulation and board measurement of the signal at the memory in slot 2 of a double parallel termination using an ODT setting of 150 Ω on slot 1 with a memory-side series resistor transmission line when the FPGA is writing to the memory with a 25-Ω OCT drive strength setting.

Figure 5–24. HyperLynx Simulation and Board Measurements of the Signal at the Memory in Slot 2 with Both Slots Populated

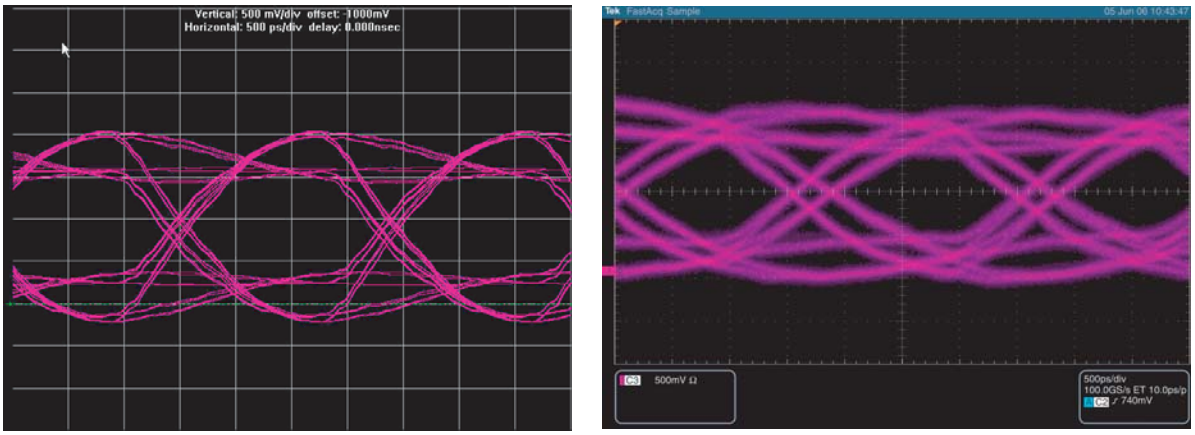


Table 5–14 summarizes the comparison between the simulation and board measurements of the signal seen at the memory of a dual-DIMM memory interface with both slots populated using a double parallel termination using a different ODT setting on Slot 1 with a memory-side series resistor with a 25-Ω OCT strength setting on the FPGA.

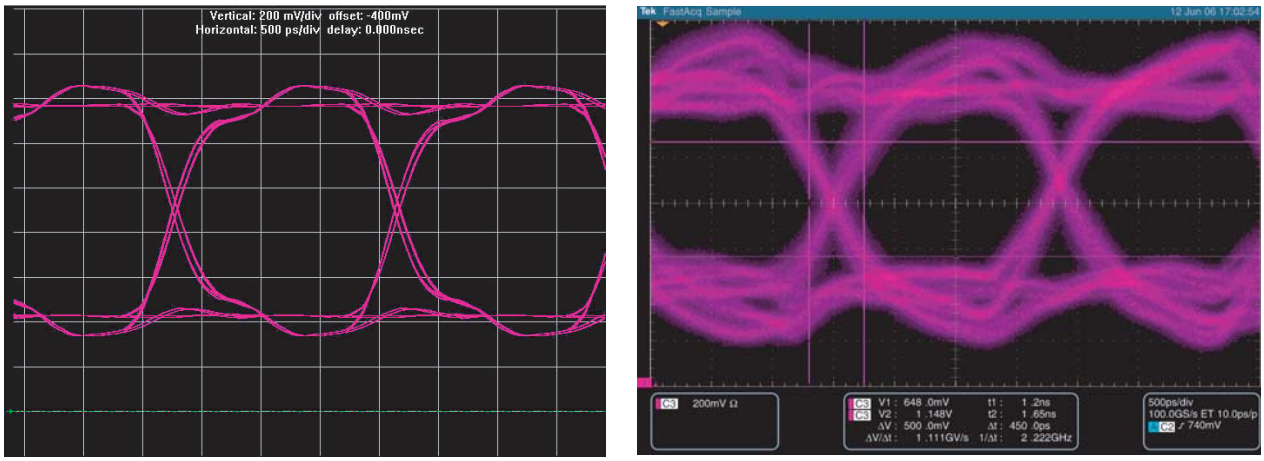
Table 5–14. Comparison of the Signal at the Memory of a Dual-DIMM Interface With Both Slots Populated and a Different ODT Setting on Slot 1

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
ODT Setting of 150 Ω						
Simulation	1.45	1.11	0.19	0.17	1.43	2.21
Measurements	0.71	0.81	0.12	0.20	0.93	1.00
ODT Setting of 75 Ω						
Simulation	1.60	1.16	0.10	0.08	1.68	1.60
Measurements	1.10	0.85	0.16	0.19	1.11	1.25

Read from Memory in Slot 1 Using an ODT Setting of 150 Ω on Slot 2 with Both Slots Populated

Figure 5–25 shows the HyperLynx simulation and board measurement of the signal at the FPGA of a double parallel termination using an external parallel resistor on the FPGA side with a memory-side series resistor and an ODT setting of 150 Ω with a full drive strength setting on the memory.

Figure 5–25. HyperLynx Simulation and Board Measurement of the Signal at the FPGA When Reading From Slot 1 With Both Slots Populated *(1)*



Note to Figure 5–25:

(1) The vertical scale used for the simulation and measurement is set to 200 mV per division.

Table 5–15 summarizes the comparison between the simulation and board measurements of the signal seen at the FPGA of a dual-DIMM memory interface with both slots populated using a different ODT setting on Slot 2.

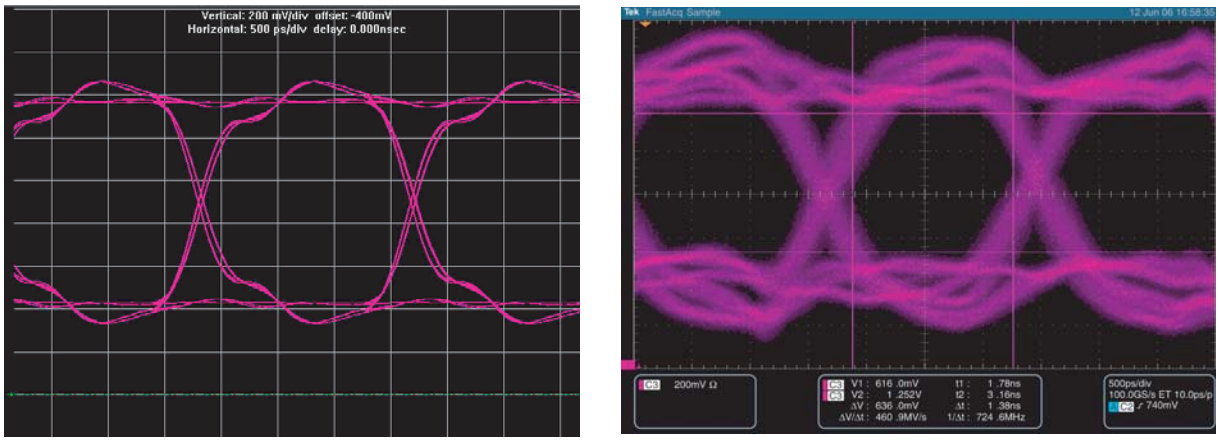
Table 5–15. Comparison of Signal at the FPGA of a Dual-DIMM Interface With Both Slots Populated and a Different ODT Setting on Slot 2

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rise Edge Rate (V/ns)	Falling Edge Rate (V/ns)
ODT Setting of 150 Ω						
Simulation	1.68	0.77	NA	NA	1.88	1.88
Measurements	0.76	0.55	NA	NA	1.11	1.14
ODT Setting of 75 Ω						
Simulation	1.74	0.87	NA	NA	1.91	1.88
Measurements	0.86	0.59	NA	NA	1.11	1.09

Read From Memory in Slot 2 Using an ODT Setting of 150 Ω on Slot 1 With Both Slots Populated

Figure 5–26 shows the HyperLynx simulation board measurement of the signal seen at the FPGA of a double parallel termination using an external parallel resistor on the FPGA side with memory-side series resistor and an ODT setting of 150 Ω with a full drive strength setting on the memory.

Figure 5–26. HyperLynx Simulation Board Measurement of the Signal at the FPGA When Reading From Slot 2 With Both Slots Populated *(1)*



Note to Figure 5–26:

(1) The vertical scale used for the simulation and measurement is set to 200 mV per division.

Table 5–16 summarizes the comparison between the simulation and board measurements of the signal seen at the FPGA of a dual-DIMM memory interface with both slots populated using a different ODT setting on Slot 1.

Table 5–16. Comparison of Signal at the FPGA of a Dual-DIMM Interface With Both Slots Populated and a Different ODT Setting on Slot 1

Type	Eye Width (ns)	Eye Height (V)	Overshoot (V)	Undershoot (V)	Rising Edge Rate (V/ns)	Falling Edge Rate (V/ns)
ODT Setting of 150 Ω						
Simulation	1.70	0.74	NA	NA	1.91	1.64
Measurements	0.74	0.64	NA	NA	1.14	1.14
ODT Setting of 75 Ω						
Simulation	1.70	0.81	NA	NA	1.72	1.99
Measurements	0.87	0.59	NA	NA	1.09	1.14

FPGA OCT Features

Many FPGA devices offer OCT. Depending on the chosen device family, series (output), parallel (input) or dynamic (bidirectional) OCT may be supported.

For more information specific to your device family, refer to the respective I/O features chapter in the relevant device handbook.

Use series OCT in place of the near-end series terminator typically used in both Class I or Class II termination schemes that both DDR2 and DDR3 type interfaces use.

Use parallel OCT in place of the far-end parallel termination typically used in Class I termination schemes on unidirectional input only interfaces. For example, QDR-II type interfaces, when the FPGA is at the far end.

Use dynamic OCT in place of both the series and parallel termination at the FPGA end of the line. Typically use dynamic OCT for DQ and DQS signals in both DDR2 and DDR3 type interfaces. As the parallel termination is dynamically disabled during writes, the FPGA driver only ever drives into a Class I transmission line. When combined with dynamic ODT at the memory, a truly dynamic Class I termination scheme exists where both reads and writes are always fully Class I terminated in each direction. Hence, you can use a fully dynamic bidirectional Class I termination scheme instead of a static discretely terminated Class II topology, which saves power, printed circuit board (PCB) real estate, and component cost.

Arria V, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices

Arria® V, Cyclone® V, Stratix III, Stratix IV, and Stratix V devices feature full dynamic OCT termination capability, Altera advise that you use this feature combined with the SDRAM ODT to simplify PCB layout and save power.

Arria II GX Devices

Arria II GX devices do not support dynamic OCT. Altera recommends that you use series OCT with SDRAM ODT. Use parallel discrete termination at the FPGA end of the line when necessary,

For more information, refer to the *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

Document Revision History

Table 5–17 lists the revision history for this document.

Table 5–17. Document Revision History

Date	Version	Changes
June 2012	4.1	Added Feedback icon.
November 2011	4.0	Added Arria V and Cyclone V information.
June 2011	3.0	Added Stratix V information.
December 2010	2.1	Maintenance update.
July 2010	2.0	Updated Arria II GX information.
April 2010	1.0	Initial release.

6. LPDDR2 SDRAM Board Design Guidelines

EMI_DG_016-1.0

This chapter provides guidelines to improve your system's signal integrity and to successfully implement an LPDDR2 SDRAM interface in your system.

The LPDDR2 SDRAM Controller with UniPHY intellectual property (IP) enables you to implement LPDDR2 SDRAM interfaces with Arria® V and Cyclone® V devices.

This chapter focuses on the following key factors that affect signal integrity:

- I/O standards
- LPDDR2 configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

I/O Standards

LPDDR2 SDRAM interface signals use HSUL-12 JEDEC I/O signaling standards, which provide low power and low emissions. The HSUL-12 JEDEC I/O standard is mainly for point-to-point unterminated bus topology. This standard eliminates the need for external series or parallel termination resistors in LPDDR2 SDRAM implementation. With this standard, termination power is greatly reduced and programmable drive strength is used to match the impedance.

To select the most appropriate standard for your interface, refer to the the *Device Datasheet for Arria V Devices* chapter in the *Arria V Device Handbook*, or the *Device Datasheet for Cyclone V Devices* chapter in the *Cyclone V Device Handbook*.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

LPDDR2 SDRAM Configurations

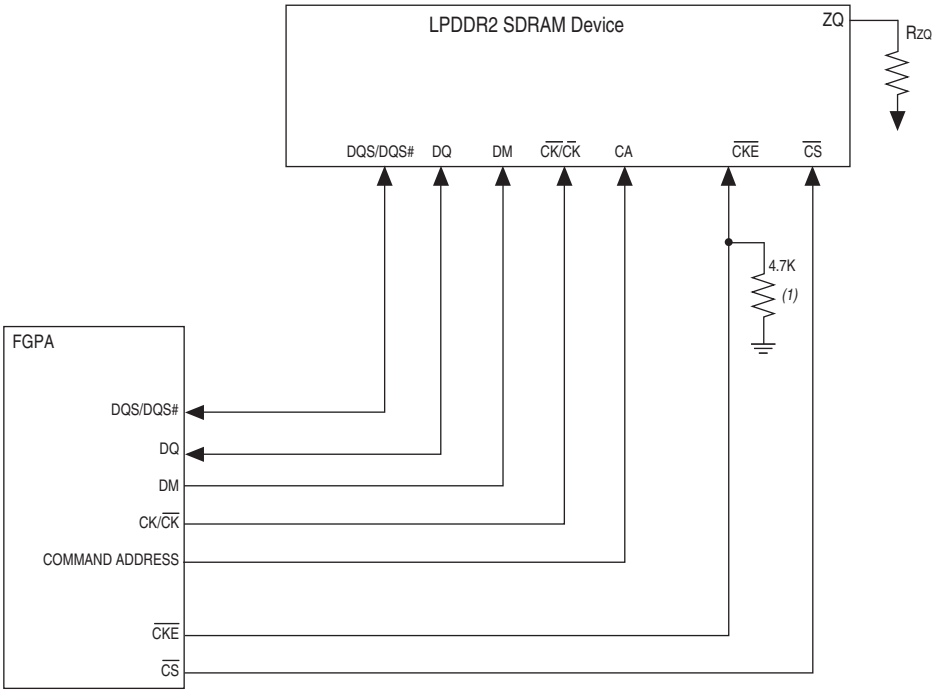
The LPDDR2 SDRAM Controller with UniPHY IP supports interfaces for LPDDR2 SDRAM with a single device, and multiple devices up to a maximum width of 32 bits.

When using multiple devices, a balanced-T topology is recommended for the signal connected from single point to multiple point, to maintain equal flight time.
You should connect a 200 ohm differential termination resistor between CK/CK# in multiple device designs as shown in Figure 6–2, to maintain an effective resistance of 100 ohms.

You should also simulate your multiple device design to obtain the optimum drive strength settings and ensure correct operation.

Figure 6–1 shows the main signal connections between the FPGA and a single LPDDR2 SDRAM component.

Figure 6–1. Configuration with a Single LPDDR2 SDRAM Component

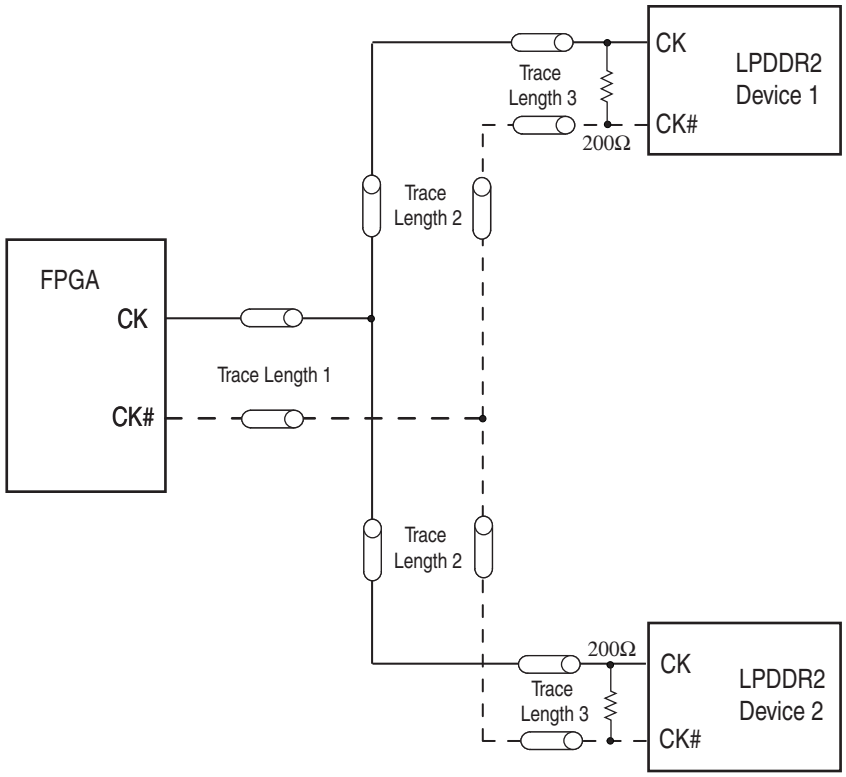


Notes to Figure 6–1:

(1) Use external discrete termination, as shown for CKE, but you may require a pull-down resistor to GND. Refer to the LPDDR2 SDRAM device data sheet for more information about LPDDR2 SDRAM power-up sequencing.

Figure 6–2 shows the differential resistor placement for CK/CK# for multi-point designs.

Figure 6–2. CK Differential Resistor Placement for Multi Point Design

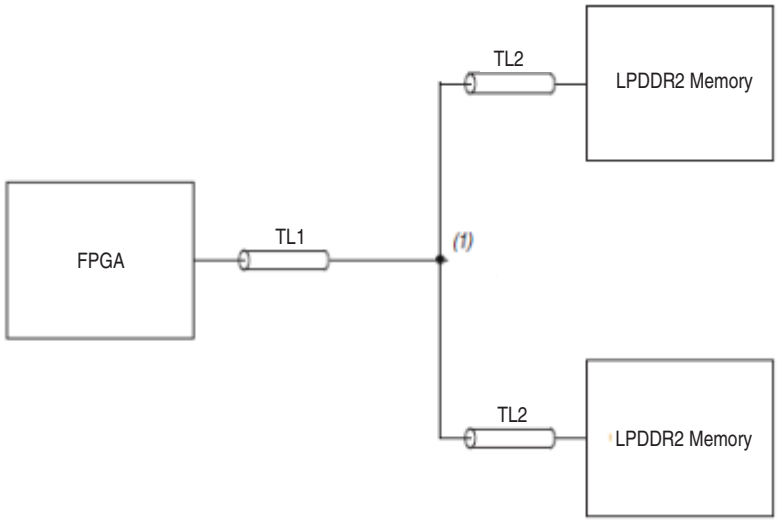


Notes to Figure 6–2:

(1) Place 200-ohm differential resistors near the memory devices at the end of the last board trace segments.

Figure 6–3 shows the detailed balanced topology recommended for the address and command signals in the multi-point design.

Figure 6–3. Address Command Balanced-T Topology



Notes for Figure 6–3:

(1) Split the trace close to the memory devices to minimize signal reflections and impedence nonuniformity.
(2) Keep the TL2 traces as short as possible, so that the memory devices appear as a single load.

Signal Terminations

Arria V and Cyclone V devices offer OCT technology. Table 6–1 lists the extent of OCT support for each device.

Table 6–1. On-Chip Termination Schemes

Termination Scheme	I/O Standard	Arria V and Cyclone V
On-Chip Series Termination without Calibration	HSUL-12	34/40/48/60/80
On-Chip Series Termination with Calibration	HSUL-12	34/40/48/60/80

On-chip series (R_S) termination supports output buffers, and bidirectional buffers only when they are driving output signals. LPDDR2 SDRAM interfaces have bidirectional data paths. The UniPHY IP uses series OCT for memory writes but no parallel OCT for memory reads because Arria V and Cyclone V support only on-chip series termination in the HSUL-12 I/O standard.

For Arria V and Cyclone V devices, the HSUL-12 I/O calibrated terminations are calibrated against 240 ohm 1% resistors connected to the R_{ZQ} pins in an I/O bank with the same V_{CCIO} as the LPDDR2 interface.

Calibration occurs at the end of the device configuration.

LPDDR2 SDRAM memory components have a ZQ pin which connects through a resistor R_{ZQ} (240 ohm) to ground. The output signal impedances for LPDDR2 SDRAM are 34.3 ohm, 40 ohm, 48 ohm, 60 ohm, 80 ohm, and 120 ohm. The output signal impedance is set by mode register during initialization. Refer to the LPDDR2 SDRAM device data sheet for more information.

For information about OCT, refer to the *I/O Features in Arria V Devices* chapter in the *Arria V Device Handbook*, or the *I/O Features in Cyclone V Devices* chapter in the *Cyclone V Device Handbook*.

The following section shows HyperLynx simulation eye diagrams to demonstrate signal termination options. Altera strongly recommends signal terminations to optimize signal integrity and timing margins, and to minimize unwanted emissions, reflections, and crosstalk.

All of the eye diagrams shown in this section are for a 50 ohm trace with a propagation delay of 509 ps which is approximately a 2.8-inch trace on a standard FR4 PCB. The signal I/O standard is HSUL-12.

The eye diagrams in this section show the best case achievable and do not take into account PCB vias, crosstalk, and other degrading effects such as variations in the PCB structure due to manufacturing tolerances.

Simulate your design to ensure correct operation.

Outputs from the FPGA to the LPDDR2 Component

The following output signals are from the FPGA to the LPDDR2 SDRAM component:

- write data (`DQ`)
- data mask (`DM`)
- data strobe (`DQS/DQS#`)
- command address
- command (`CS`, and `CKE`)
- clocks (`CK/CK#`)

No far-end memory termination is needed when driving output signals from FPGA to LPDDR2 SDRAM. Cyclone V and Arria V devices offer the OCT series termination for impedance matching.

The HyperLynx simulation eye diagrams show simulation cases of write data, address, and chip-select signals with the OCT settings. All eye diagrams are shown at the connection to the receiver device die.

Figure 6–4 shows the double data rate write data using an Arria V HSUL-12 with calibrated series 34 ohm OCT output driver.

Figure 6–4. Write Data Simulation at 400 MHz

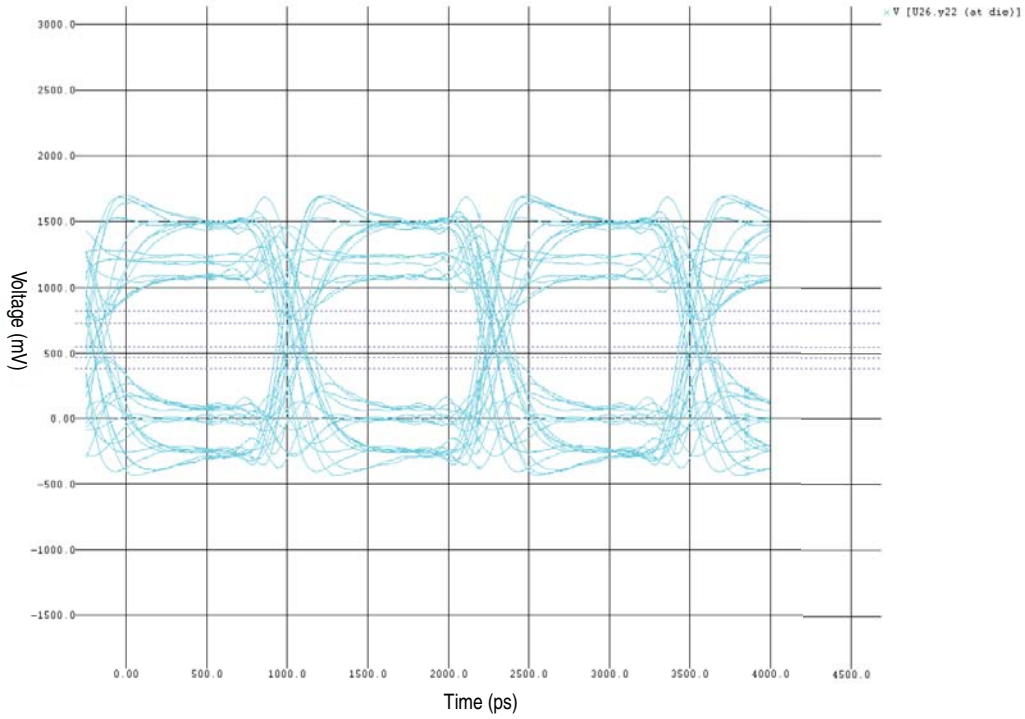


Figure 6–5 shows an address command signal at a frequency of 400 MHz using Arria V HSUL-12 with calibrated series 34 ohm OCT. Address command signals are also double data rate so they are running at 400 MHz.

Figure 6–5. Address Command at 400 MHz Termination

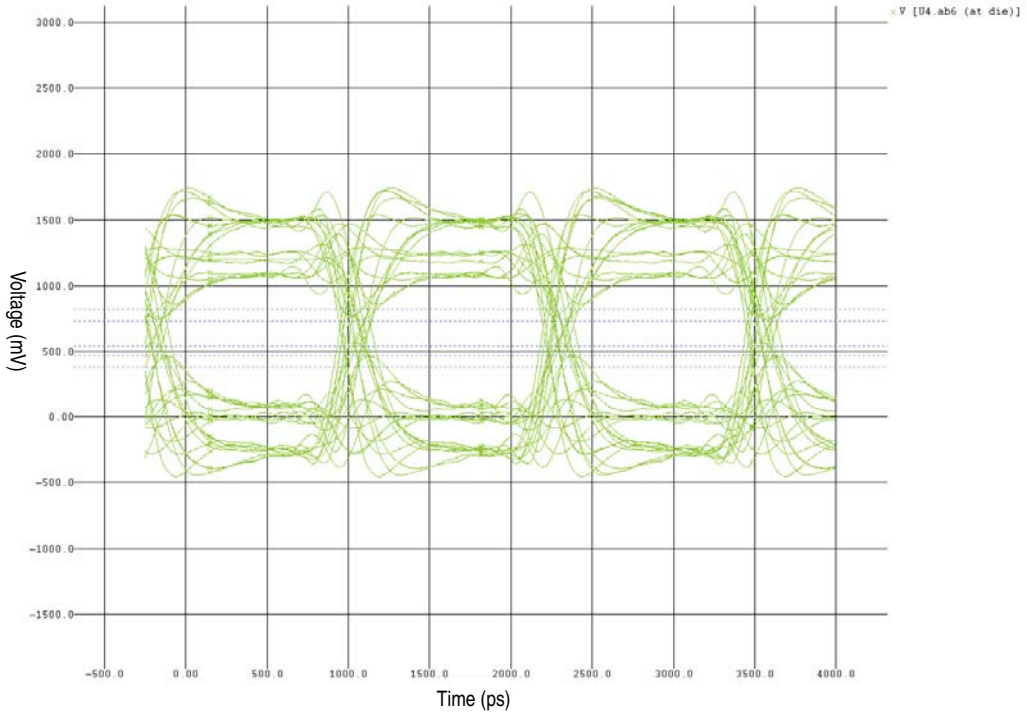
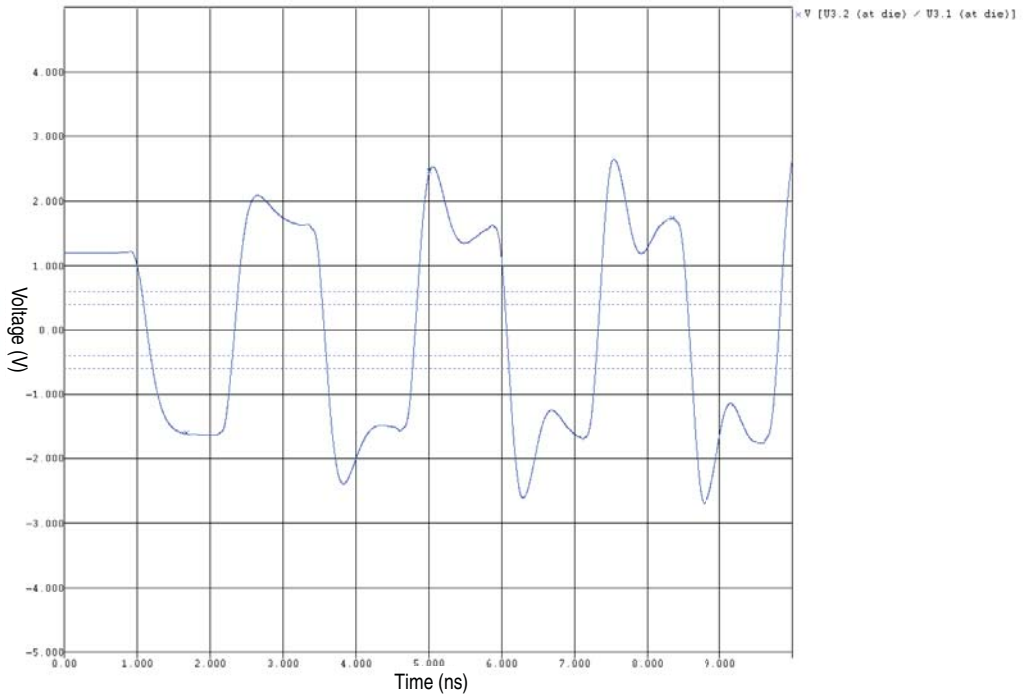


Figure 6–6 shows a memory clock signal at a frequency of 400 MHz using Arria V HSUL-12 with calibrated series 34 ohm OCT.

Figure 6–6. Memory Clock Simulation at 400MHz



Input to the FPGA from the LPDDR2 SDRAM Component

The LPDDR2 SDRAM component drives the following input signals into the FPGA:

- read data
- DQS

LPDDR2 SDRAM provides the flexibility to adjust drive strength to match the impedance of the memory bus, eliminating the need for termination voltage (VTT) and series termination resistors.

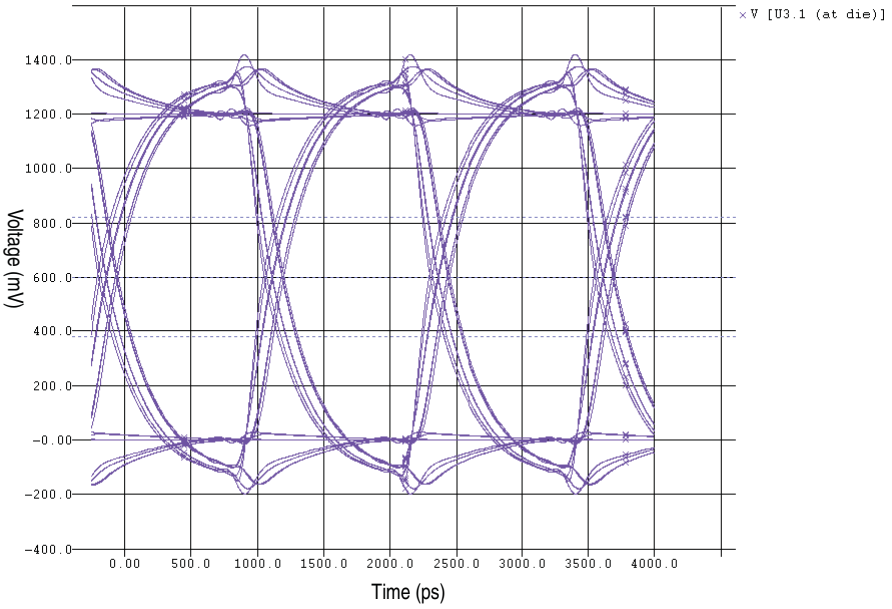
The programmable drive strength options are 34.3 ohms, 40 ohms (default), 48 ohms, 60 ohms, 80 ohms, and 120 ohms. You must perform board simulation to determine the best option for your board layout.

By default, Altera LPDDR2 SDRAM UniPHY IP uses 40 ohm drive strength.

The eye diagrams are shown at the FPGA die pin, and the LPDDR2 SDRAM output driver is HSUL-12 with ZQ calibration of 40 ohms. The LPDDR2 SDRAM read data is double data rate.

Figure 6–7 shows read data simulation at 400 MHz with 40 W drive strength on an Arria V device.

Figure 6–7. Read Data Simulation at 400 MHz with 40 Ω Drive Strength on an Arria V Device



Termination Schemes

Table 6–2 lists the recommended termination schemes for major LPDDR2 SDRAM memory interface signals, which include data (DQ), data strobe (DQS), data mask (DM), clocks (CK, and CK#), command address (CA), and control (CS#, and CKE).

Table 6–2. Termination Recommendations for Arria V and Cyclone V Devices (Part 1 of 2)

Signal Type	HSUL-12 Standard (1), (2)	Memory End Termination
DQS/DQS#	R34 CAL	ZQ40
Data (Write)	R34 CAL	–
Data (Read)	–	ZQ40
Data Mask (DM)	R34 CAL	–
CK/CK# Clocks	R34 CAL	×1 = – (4) ×2 = 200 Ω Differential (5)
Command Address (CA),	R34 CAL	–
Chip Select (CS#)	R34 CAL	–

Table 6–2. Termination Recommendations for Arria V and Cyclone V Devices (Part 2 of 2)

Signal Type	HSUL-12 Standard (1), (2)	Memory End Termination
Clock Enable (CKE) (3)	R34 CAL	4.7 KΩ parallel to GND

Notes to Table 6–2:

(1) R is effective series output impedance.

(2) CAL is OCT with calibration.

(3) Altera recommends that you use a 4.7 KΩ parallel to GND if your design meets the power sequencing requirements of the LPDDR2 SDRAM component. Refer to the LPDDR2 SDRAM data sheet for further information.

(4) ×1 is a single-device load.

(5) ×2 is a double-device load. An alternative option is to use a 100 Ω differential termination at the trace split.

The recommended termination schemes of Table 6–2 are based on 2.8 inch maximum trace length analysis. You may add the external termination resistor or adjust the drive strength to improve signal integrity for longer trace lengths. Recommendations for external termination are as follows:

- Class I termination (50 ohms parallel to VTT at the memory end) — Unidirectional signal (Command Address, control, and CK/CK# signals)
- Class II termination (50 ohms parallel to VTT at both ends) — Bidirectional signal (DQ and DQS/DQS# signal)

Altera recommends that you simulate your design to ensure good signal integrity.

PCB Layout Guidelines

Table 6–3 lists the LPDDR2 SDRAM general routing layout guidelines.

The following layout guidelines include several +/- length-based rules. These length-based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristics of your PCB implementation. They do not include any margin for crosstalk.

Altera recommends that you get accurate time base skew numbers when you simulate your specific implementation.

Table 6–3. LPDD2 Layout Guidelines (Part 1 of 3)

Parameter	Guidelines
Impedance	■ All signal planes must be 50 Ω, single-ended, ±10%. ■ All signal planes must be 100 Ω, differential ±10%. ■ Remove all unused via pads, because they cause unwanted capacitance.
Decoupling Parameter	■ Use 0.1 µF in 0402 size to minimize inductance. ■ Verify your capacitive decoupling using the Altera Power Distribution Network (PDN) Design tool.
Power	■ Route GND, 1.2 V and 1.8 V as planes. ■ Route V_{CCIO} for memories in a single split plane with at least a 20-mil (0.020 inches or 0.508 mm) gap of separation. ■ Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces.

Table 6–3. LPDD2 Layout Guidelines (Part 2 of 3)

Parameter	Guidelines
General Routing	■ All specified delay matching requirements include PCB trace delays, different layer propagation, velocity variance, and crosstalk. To minimize PCB layer propagation variance, Altera recommends that signals from the same net group always be routed on the same layer. If you must route signals of the same net group on different layers with the same impedance characteristic, simulate your worst case PCB trace tolerances to ascertain actual propagation delay differences. Typical layer to layer trace delay variations are of 15 ps/inch order. ■ Use 45° angles (not 90° corners). ■ Avoid T-Junctions for critical nets or clocks. ■ Avoid T-junctions greater than 75 ps (approximately 25 mils, 6.35 mm). ■ Disallow signals across split planes. ■ Restrict routing other signals close to system reset signals. ■ Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks. ■ Match all signals within a given DQ group with a maximum skew of ±10 ps or approximately ±50 mils (0.254 mm) and route on the same layer.
Clock Routing	■ Route clocks on inner layers with outer-layer run lengths held to under 150 ps (approximately 500 mils, 12.7 mm). ■ These signals should maintain a 10-mil (0.254 mm) spacing from other nets. ■ Clocks should maintain a length-matching between clock pairs of ±5 ps or approximately ±25 mils (0.635 mm). ■ Differential clocks should maintain a length-matching between `P` and `N` signals of ±2 ps or approximately ±10 mils (0.254 mm). ■ Space between different clock pairs should be at least three times the space between the traces of a differential pair.
Address and Command Routing	■ To minimize crosstalk, route address, and command signals on a different layer than the data and data mask signals. ■ Do not route the differential clock (CK/CK#) and clock enable (CKE) signals close to the address signals.

Table 6–3. LPDD2 Layout Guidelines (Part 3 of 3)

Parameter	Guidelines
External Memory Routing Rules	■ Apply the following parallelism rules for the LPDDR2 SDRAM data groups: ■ 4 mils for parallel runs < 0.1 inch (approximately 1× spacing relative to plane distance). ■ 5 mils for parallel runs < 0.5 inch (approximately 1× spacing relative to plane distance). ■ 10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance). ■ 15 mils for parallel runs between 1.0 and 2.8 inch (approximately 3× spacing relative to plane distance). ■ Apply the following parallelism rules for the address/command group and clocks group: ■ 4 mils for parallel runs < 0.1 inch (approximately 1× spacing relative to plane distance) ■ 10 mils for parallel runs < 0.5 inch (approximately 2× spacing relative to plane distance) ■ 15 mils for parallel runs between 0.5 and 1.0 inches (approximately 3× spacing relative to plane distance) ■ 20 mils for parallel runs between 1.0 and 2.8 inches (approximately 4× spacing relative to plane distance)
Maximum Trace Length	■ Keep traces as short as possible. The maximum trace length of all signals from the FPGA to the LPDDR2 SDRAM components should be less than 509 ps (approximately 28 mils, 47.4 mm). Altera recommends that you simulate your design to ensure good signal integrity.

Altera recommends the following layout approach, based on the layout guidelines in Table 6–3:

1. Route the differential clocks (CK/CK#) and data strobe (DQS/DQS#) with a length-matching between P and N signals of ±2 ps.
2. Route the DQS/DQS# associated with a DQ group on the same PCB layer. Match these DQS pairs to within ±5 ps.
3. Set the DQS/DQS# as the target trace propagation delay for the associated data and data mask signals.
4. Route the data and data mask signals for the DQ group ideally on the same layer as the associated DQS/DQS# to within ±10 ps skew of the target DQS/DQS#.
5. Route the CK/CK# clocks and set as the target trace propagation delays for the DQ group. Match the CK/CK# clock to within ±50 ps of all the DQS/DQS#.
6. Route the address/control signal group (address, CS, CKE) ideally on the same layer as the CK/CK# clocks, to within ±20 ps skew of the CK/CK# traces.

This layout approach provides a good starting point for a design requirement of the highest clock frequency supported for the LPDDR2 SDRAM interface.

Altera recommends that you create your project in the Quartus® II software with a fully implemented LPDDR2 SDRAM Controller with UniPHY interface, and observe the interface timing margins to determine the actual margins for your design.

Although the recommendations in this chapter are based on simulations, you can apply the same general principles when determining the best termination scheme, drive strength setting, and loading style to any board designs. Even armed with this knowledge, it is still critical that you perform simulations, either using IBIS or HSPICE models, to determine the quality of signal integrity on your designs.

Document Revision History

Table 6–4 lists the revision history for this document.

Table 6–4. Document Revision History

Date	Version	Changes
November 2012	1.0	Initial release.

7. RLDRAM II and RLDRAM 3 Board Design Guidelines

EMI_DG_006-3.2

This chapter provides layout guidelines for you to improve your system's signal integrity and to successfully implement an RLDRAM II or RLDRAM 3 interface.

The RLDRAM II Controller with UniPHY intellectual property (IP) enables you to implement common I/O (CIO) RLDRAM II interfaces with Arria® V, Stratix® III, Stratix IV, and Stratix V devices. The RLDRAM 3 UniPHY IP enables you to implement CIO RLDRAM 3 interfaces with Stratix V and Arria V GZ devices. You can implement separate I/O (SIO) RLDRAM II or RLDRAM 3 interfaces with the ALTDQ_DQS or ALTDQ_DQS2 megafunctions.

This chapter focuses on the following key factors that affect signal integrity:

- I/O standards
- RLDRAM II and RLDRAM 3 configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

I/O Standards

RLDRAM II interface signals use one of the following JEDEC I/O signalling standards:

- HSTL-15—provides the advantages of lower power and lower emissions.
- HSTL-18—provides increased noise immunity with slightly greater output voltage swings.

RLDRAM 3 interface signals use the following JEDEC I/O signalling standards: HSTL 1.2 V and SSTL-12.

To select the most appropriate standard for your interface, refer to the following:

- *Device Data Sheet for Arria II Devices* chapter in the *Arria II Device Handbook*
- *Device Data Sheet for Arria V Devices* chapter in the *Arria V Device Handbook*
- *Stratix III Device Data Sheet: DC and Switching Characteristics* chapter in the *Stratix III Device Handbook*
- *DC and Switching Characteristics for Stratix IV Devices* chapter in the *Stratix IV Device Handbook*
- *DC and Switching Characteristics for Stratix V Devices* chapter in the *Stratix V Device Handbook*

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

The RLDRAM II Controller with UniPHY IP defaults to HSTL 1.8 V Class I outputs and HSTL 1.8 V inputs. The RLDRAM 3 UniPHY IP defaults to HSTL 1.2 V Class I outputs and HSTL 1.2 V inputs.

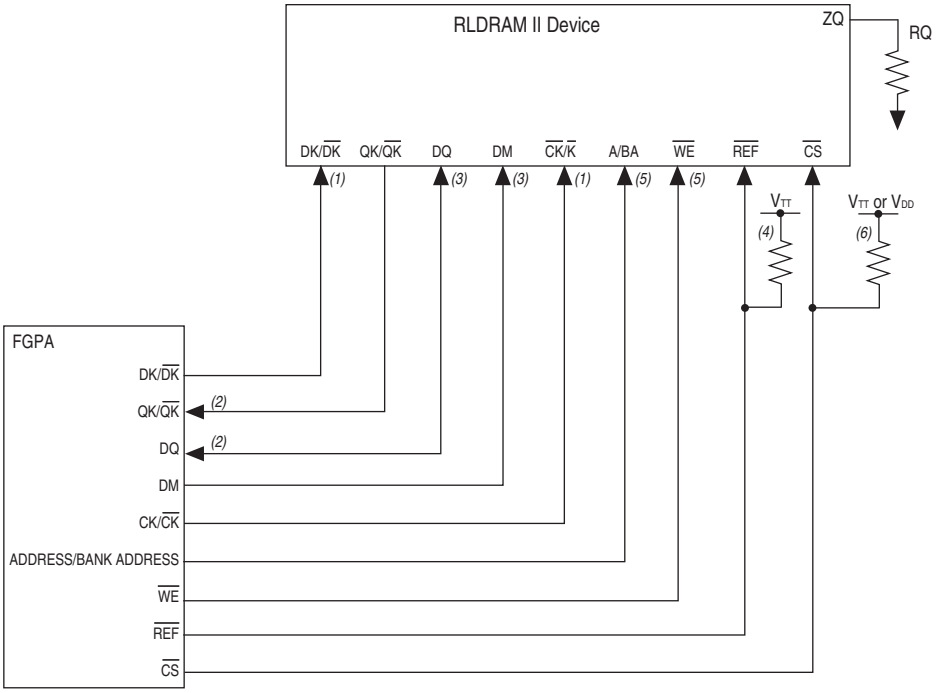
The default for RLDRAM 3 changes from Class I to Class II, supporting up to 933 MHz, with the release of the Quartus II software version 12.1 SP1.

RLDRAM II Configurations

The RLDRAM II Controller with UniPHY IP supports CIO RLDRAM II interfaces with one or two devices. With two devices, the interface supports a width expansion configuration up to 72-bits. The termination and layout principles for SIO RLDRAM II interfaces are similar to CIO RLDRAM II, except that SIO RLDRAM II interfaces have unidirectional data buses.

Figure 7–1 shows the main signal connections between the FPGA and a single CIO RLDRAM II component.

Figure 7–1. Configuration with a Single CIO RLDRAM II Component



Notes to Figure 7–1:

(1) Use external differential termination on DK/DK# and CK/CK#.

(2) Use FPGA parallel on-chip termination (OCT) for terminating QK/QK# and DQ on reads.

(3) Use RLDRAM II component on-die termination (ODT) for terminating DQ and DM on writes.

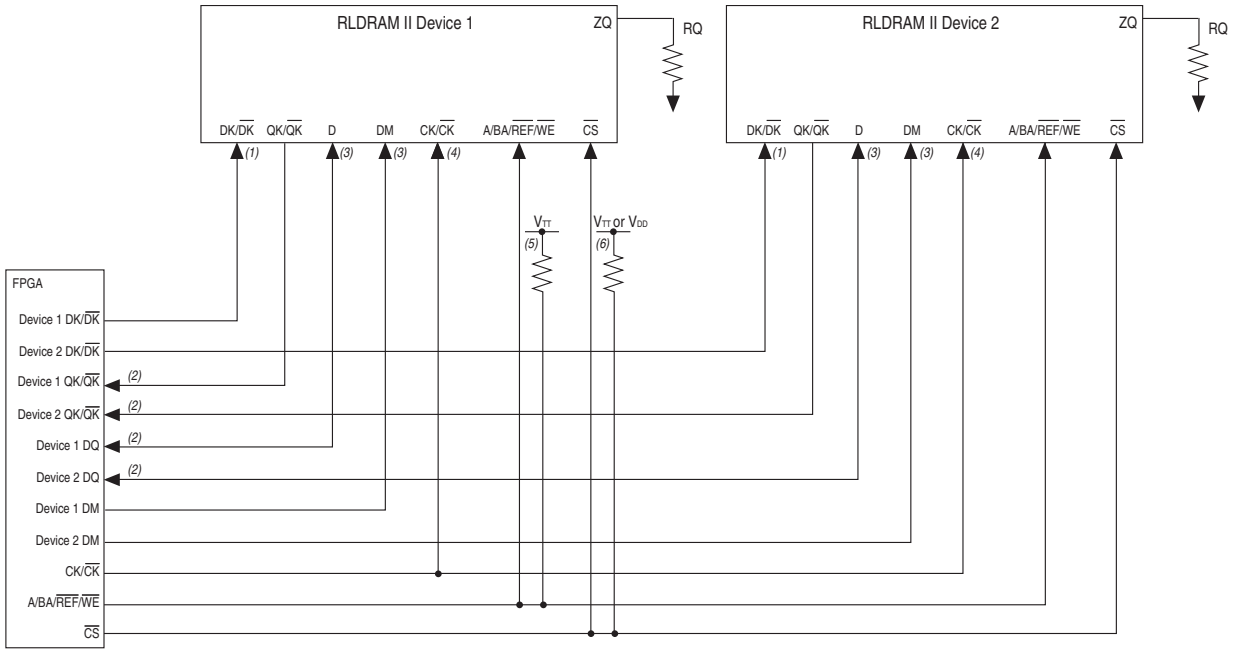
(4) Use external discrete termination with fly-by placement to avoid stubs.

(5) Use external discrete termination for this signal, as shown for REF.

(6) Use external discrete termination, as shown for REF, but you may require a pull-up resistor to V_{DD} as an alternative option. Refer to the RLDRAM II device data sheet for more information about RLDRAM II power-up sequencing.

Figure 7–2 shows the main signal connections between the FPGA and two CIO RLDRAM II components in a width expansion configuration.

Figure 7–2. Configuration with Two CIO RLDRAM II Components in a Width Expansion Configuration



Notes to Figure 7–2:

(1) Use external differential termination on DK/DK#.

(2) Use FPGA parallel OCT for terminating QK/QK# and DQ on reads.

(3) Use RLDRAM II component ODT for terminating DQ and DM on writes.

(4) Use external dual 200 Ω differential termination.

(5) Use external discrete termination at the trace split of the balanced T or Y topology.

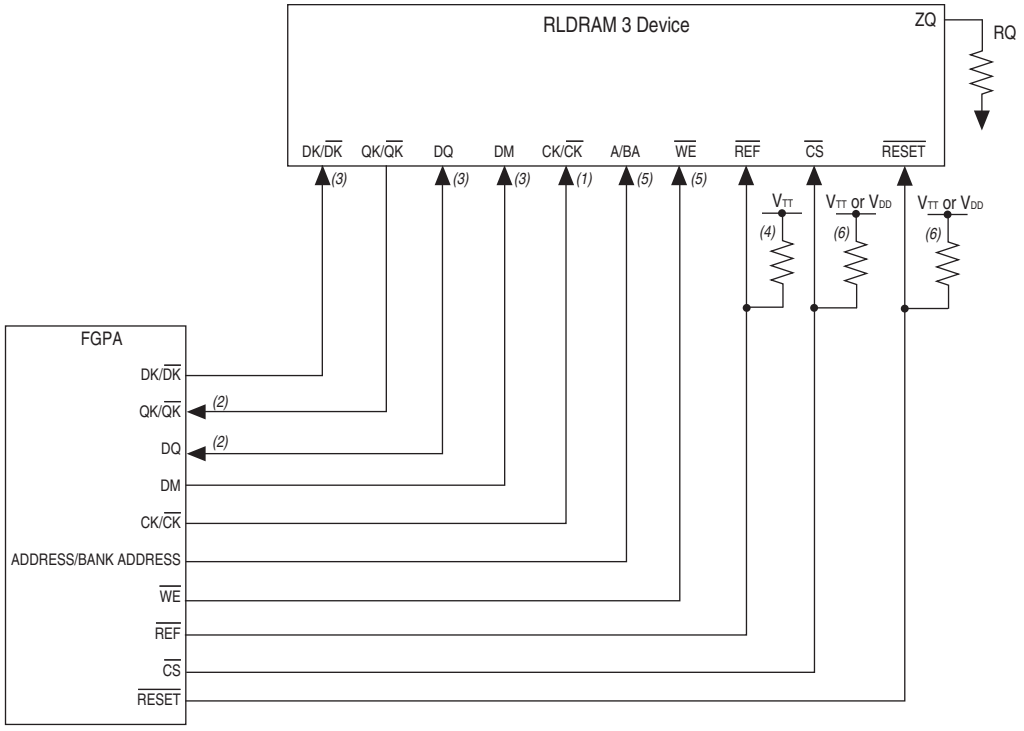
(6) Use external discrete termination at the trace split of the balanced T or Y topology, but you may require a pull-up resistor to V_{DD} as an alternative option. Refer to the RLDRAM II device data sheet for more information about RLDRAM II power-up sequencing.

RLDRAM 3 Configurations

The RLDRAM 3 UniPHY IP supports interfaces for CIO RLDRAM 3 with one or two devices. With two devices, the interface supports a width expansion configuration up to 72-bits. The termination and layout principles for SIO RLDRAM 3 interfaces are similar to CIO RLDRAM 3, except that SIO RLDRAM 3 interfaces have unidirectional data buses.

Figure 7–3 shows the main signal connections between the FPGA and a single CIO RLDRAM 3 component.

Figure 7–3. Configuration with a Single CIO RLDRAM 3 Component



Notes to Figure 7–3:

(1) Use external differential termination on CK/CK#.

(2) Use FPGA parallel on-chip termination (OCT) for terminating QK/QK# and DQ on reads.

(3) Use RLDRAM 3 component on-die termination (ODT) for terminating DQ and DM on writes.

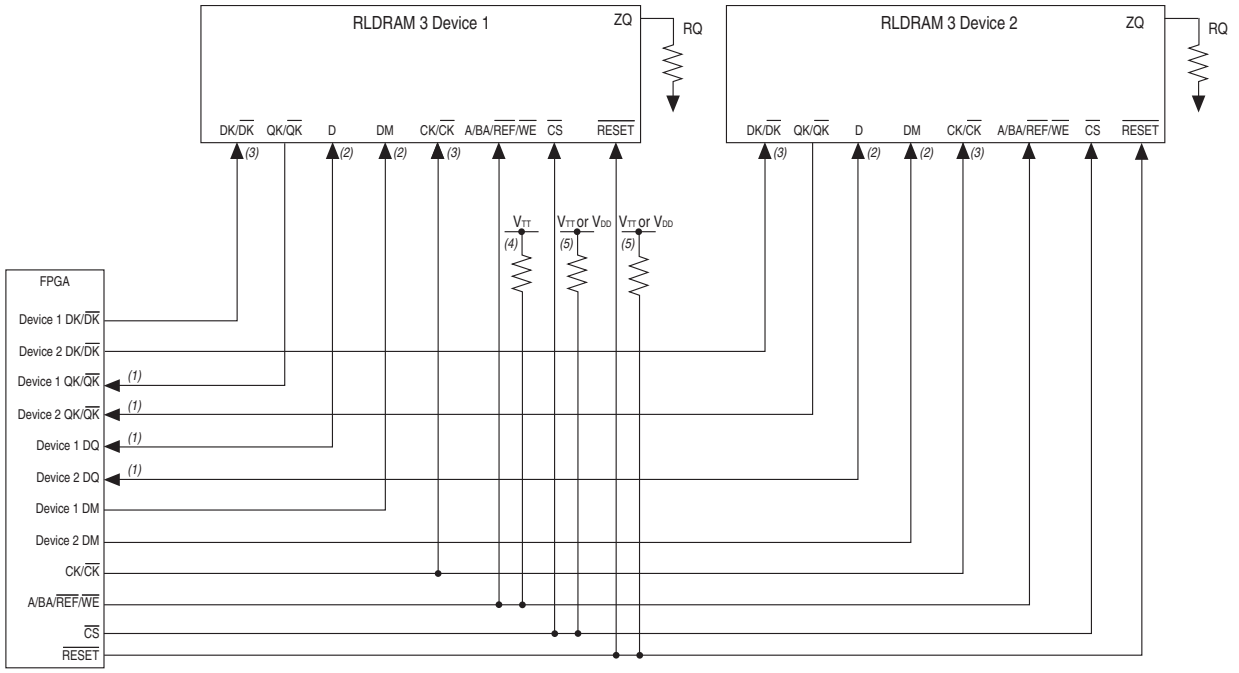
(4) Use external discrete termination with fly-by placement to avoid stubs.

(5) Use external discrete termination for this signal, as shown for REF.

(6) Use external discrete termination, as shown for REF, but you may require a pull-up resistor to V_{DD} as an alternative option. Refer to the RLDRAM 3 device data sheet for more information about RLDRAM 3 power-up sequencing.

Figure 7–4 shows the main signal connections between the FPGA and two CIO RLDRAM 3 components in a width expansion configuration.

Figure 7–4. Configuration with Two CIO RLDRAM 3 Components in a Width Expansion Configuration



Notes to Figure 7–4:

(1) Use FPGA parallel OCT for terminating QK/QK# and DQ on reads.

(2) Use RLDRAM 3 component ODT for terminating DQ, DM, and DK on writes.

(3) Use external dual 200 Ω differential termination.

(4) Use external discrete termination at the trace split of the balanced T or Y topology.

(5) Use external discrete termination at the trace split of the balanced T or Y topology, but you may require a pull-up resistor to V_{DD} as an alternative option. Refer to the RLDRAM 3 device data sheet for more information about RLDRAM 3 power-up sequencing.

Signal Terminations

Table 7–1 lists the on-chip series termination (R_S OCT) and on-chip parallel termination (R_T OCT) schemes for supported devices.

For RLDRAM 3, the default output termination resistance (R_S) changes from 50 ohm to 25 ohm with the release of the Quartus II software version 12.1 SP1.

Table 7–1. On-Chip Termination Schemes

Termination Scheme	Class I Signal Standards	FPGA Device	
		Arria II GZ, Stratix III, and Stratix IV	Arria V and Stratix V
		Row/Column I/O	Row/Column I/O
R_S OCT without Calibration	RLDRAM II - HSTL-15 and HSTL-18 RLDRAM 3 - HSTL 1.2 V	50	50
R_S OCT with Calibration	RLDRAM II - HSTL-15 and HSTL-18 RLDRAM 3 - HSTL 1.2 V	50	50 (1)
R_T OCT with Calibration	RLDRAM II - HSTL-15 and HSTL-18 RLDRAM 3 - HSTL 1.2 V	50	50 (1)

Note to Table 7–1:

(1) Although 50 Ω is the recommended option, Stratix V devices offer a wider range of calibrated termination impedances.

RLDRAM II and RLDRAM 3 CIO interfaces have bidirectional data paths. The UniPHY IP uses dynamic OCT on the datapath, which switches between series OCT for memory writes and parallel OCT for memory reads. The termination schemes also follow these characteristics:

- R_S OCT supports output buffers.
- R_T OCT supports input buffers.
- R_S OCT supports bidirectional buffers only when they are driving output signals.
- R_T OCT bidirectional buffers only when they are input signals.

For Arria II GZ, Stratix III, and Stratix IV devices, the HSTL Class I I/O calibrated terminations are calibrated against 50 Ω 1% resistors connected to the R_{UP} and R_{DN} pins in an I/O bank with the same V_{CCIO} as the RLDRAM II interface. For Arria V and Stratix V devices, the HSTL Class I I/O calibrated terminations are calibrated against 100 Ω 1% resistors connected to the R_{ZQ} pins in an I/O bank with the same V_{CCIO} as the RLDRAM II and RLDRAM 3 interfaces.

The calibration occurs at the end of the device configuration.

RLDRAM II and RLDRAM 3 memory components have a ZQ pin that connects through a resistor RQ to ground. Typically the RLDRAM II and RLDRAM 3 output signal impedance is a fraction of RQ. Refer to the RLDRAM II and RLDRAM 3 device data sheets for more information.

For information about OCT, refer to the following:

- *I/O Features in Arria II Devices* chapter in the *Arria II Device Handbook*
- *I/O Features in Arria V Devices* chapter in the *Arria V Device Handbook*
- *Stratix III Device I/O Features* chapter in the *Stratix III Device Handbook*
- *I/O Features in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*
- *I/O Features in Stratix V Devices* chapter in the *Stratix V Device Handbook*

Altera strongly recommends signal terminations to optimize signal integrity and timing margins, and to minimize unwanted emissions, reflections, and crosstalk.

Simulate your design to check your termination scheme.

Input to the FPGA from the RLDRAM Components

The RLDRAM II or RLDRAM 3 component drives the following input signals into the FPGA:

- Read data (DQ on the bidirectional data signals for CIO RLDRAM II and CIO RLDRAM 3).
- Read clocks (QK/QK#).

Altera recommends that you use the FPGA parallel OCT to terminate the data on reads and read clocks.

Outputs from the FPGA to the RLDRAM II and RLDRAM 3 Components

The following output signals are from the FPGA to the RLDRAM II and RLDRAM 3 components:

- Write data (DQ on the bidirectional data signals for CIO RLDRAM II and RLDRAM 3)
- Data mask (DM)
- Address, bank address
- Command (CS, WE, and REF)
- Clocks (CK/CK# and DK/DK#)

For point-to-point single-ended signals requiring external termination, Altera recommends that you place a fly-by termination by terminating at the end of the transmission line after the receiver to avoid unterminated stubs. The guideline is to place the fly-by termination within 100 ps propagation delay of the receiver.

Although not recommended, you can place the termination before the receiver, which leaves an unterminated stub. The stub delay is critical because the stub between the termination and the receiver is effectively unterminated, causing additional ringing and reflections. Stub delays should be less than 50 ps.

Altera recommends that the differential clocks, CK, CK# and DK, DK# (RLDRAM II) and CK, CK# (RLDRAM 3), use a differential termination at the end of the trace at the external memory component. Alternatively, you can terminate each clock output with a parallel termination to V_{TT}.

RLDRAM II Termination Schemes

Table 7–2 lists the recommended termination schemes for major CIO RLDRAM II memory interface signals, which include data (DQ), data mask (DM), clocks (CK, CK#, DK, DK#, QK, and QK#), address, bank address, and command (WE#, REF#, and CS#).

Table 7–2. RLDRAM II Termination Recommendations for Arria II GZ, Arria V, Stratix III, Stratix IV, and Stratix V Devices

Signal Type	HSTL 15/18 Standard (1), (2), (3), (4)	Memory End Termination
DK/DK# Clocks	Class I R50 NO CAL	100 Ω Differential
QK/QK# Clocks	Class I P50 CAL	ZQ50
Data (Write)	Class I R50 CAL	ODT
Data (Read)	Class I P50 CAL	ZQ50
Data Mask	Class I R50 CAL	ODT
CK/CK# Clocks	Class I R50 NO CAL	×1 = 100 Ω Differential (9) ×2 = 200 Ω Differential (10)
Address/Bank Address (5), (6)	Class I Max Current	50 Ω Parallel to V_{TT}
Command (WE#, REF#) (5), (6)	Class I Max Current	50 Ω Parallel to V_{TT}
Command (CS#) (5), (6), (7)	Class I Max Current	50 Ω Parallel to V_{TT} or Pull-up to V_{DD}
QVLD (8)	Class I P50 CAL	ZQ50

Notes to Figure 7–2:

(1) R is effective series output impedance.

(2) P is effective parallel input impedance.

(3) CAL is OCT with calibration.

(4) NO CAL is OCT without calibration.

(5) For width expansion configuration, the address and control signals are routed to 2 devices. Recommended termination is 50 Ω parallel to V_{TT} at the trace split of a balanced T or Y routing topology. Use a clamshell placement of the two RLDRAM II components to achieve minimal stub delays and optimum signal integrity. Clamshell placement is when two devices overlay each other by being placed on opposite sides of the PCB.

(6) The UniPHY default IP setting for this output is Max Current. A Class I 50 Ω output with calibration output is typically optimal in single load topologies.

(7) Altera recommends that you use a 50 Ω parallel termination to V_{TT} if your design meets the power sequencing requirements of the RLDRAM II component. Refer to the RLDRAM II data sheet for further information.

(8) QVLD is not used in the RLDRAM II Controller with UniPHY implementations.

(9) ×1 is a single-device load.

(10) ×2 is a double-device load. An alternative option is to use a 100 Ω differential termination at the trace split.

Altera recommends that you simulate your specific design for your system to ensure good signal integrity.

RLDRAM 3 Termination Schemes

Table 7–3 lists the recommended termination schemes for major CIO RLDRAM 3 memory interface signals, which include data (DQ), data mask (DM), clocks (CK, CK#, DK, DK#, QK, and QK#), address, bank address, and command (WE#, REF#, and CS#).

Table 7–3. RLDRAM 3 Termination Recommendations for Arria V GZ, and Stratix V Devices

Signal Type	Memory End Termination
DK/DK# Clocks	ODT (selectable 40, 60, & 120 ohms)
QK/QK# Clocks	100 Ω Differential or Ron (selectable 40 & 60 ohms)
Data (Write)	ODT (selectable 40, 60, & 120 ohms)
Data (Read)	Ron (selectable 40 & 60 ohms)
Data Mask	ODT (selectable 40, 60, & 120 ohms)
CK/CK# Clocks	×1 = 100 Ω Differential (5) ×2 = 100 Ω Differential (6)
Address/Bank Address (1), (2)	50 Ω Parallel to V_{TT}
Command (WE#, REF#) (1), (2)	50 Ω Parallel to V_{TT}
Command (CS#) (1), (2), (3)	50 Ω Parallel to V_{TT} or Pull-up to V_{DD}
QVLD (4)	Ron (selectable 40 & 60 ohms)

Notes to Table 7–3:

(1) For width expansion configuration, the address and control signals are routed to 2 devices. Recommended termination is 50 Ω parallel to V_{TT} at the trace split of a balanced T or Y routing topology. Use a clamshell placement of the two RLDRAM 3 components to achieve minimal stub delays and optimum signal integrity. Clamshell placement is when two devices overlay each other by being placed on opposite sides of the PCB.

(2) The UniPHY default IP setting for this output is Max Current. A Class I 50 Ω output with calibration output is typically optimal in single load topologies.

(3) Altera recommends that you use a 50 Ω parallel termination to V_{TT} if your design meets the power sequencing requirements of the RLDRAM 3 component. Refer to the RLDRAM 3 data sheet for further information.

(4) QVLD is not used in the RLDRAM 3 Controller with UniPHY implementations.

(5) ×1 is a single-device load.

(6) ×2 is a double-device load. An alternative option is to use a 100 Ω differential termination at the trace split.

Altera recommends that you simulate your specific design for your system to ensure good signal integrity.

PCB Layout Guidelines

Altera recommends that you create your project in the Quartus® II software with a fully implemented RLDRAM II Controller with UniPHY interface, or RLDRAM 3 with UniPHY IP, and observe the interface timing margins to determine the actual margins for your design.

Although the recommendations in this chapter are based on simulations, you can apply the same general principles when determining the best termination scheme, drive strength setting, and loading style to any board designs. Altera recommends that you perform simulations, either using IBIS or HSPICE models, to determine the quality of signal integrity on your designs, and that you get accurate time base skew numbers when you simulate your specific implementation.

The following layout guidelines include several +/- length-based rules. These length-based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristics of your PCB implementation. They do not include any margin for crosstalk.

Table 7–4 lists the RLDRAM II and RLDRAM 3 general routing layout guidelines.

Table 7–4. RLDRAM II and RLDRAM 3 Layout Guidelines (Part 1 of 2)

Parameter	Guidelines
Impedance	■ All signal planes must be 50 Ω, single-ended, ±10%. ■ All signal planes must be 100 Ω, differential ±10%. ■ Remove all unused via pads, because they cause unwanted capacitance.
Decoupling Parameter	■ Use 0.1 µF in 0402 size to minimize inductance. ■ Make V_{TT} voltage decoupling close to pull-up resistors. ■ Connect decoupling caps between V_{TT} and ground. ■ Use a 0.1 µF cap for every other V_{TT} pin. ■ Verify your capacitive decoupling using the Altera Power Distribution Network (PDN) Design tool.
Power	■ Route GND, 1.5 V/1.8 V as planes. ■ Route V_{CCIO} for memories in a single split plane with at least a 20-mil (0.020 inches or 0.508 mm) gap of separation. ■ Route V_{TT} as islands or 250-mil (6.35-mm) power traces. ■ Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces.
General Routing	■ All specified delay matching requirements include PCB trace delays, different layer propagation, velocity variance, and crosstalk. To minimize PCB layer propagation variance, Altera recommends that signals from the same net group always be routed on the same layer. If you must route signals of the same net group on different layers with the same impedance characteristic, simulate your worst case PCB trace tolerances to ascertain actual propagation delay differences. Typical layer to layer trace delay variations are of 15 ps/inch order. ■ Use 45° angles (not 90° corners). ■ Avoid T-Junctions for critical nets or clocks. ■ Avoid T-junctions greater than 150 ps (approximately 500 mils, 12.7 mm). ■ Disallow signals across split planes. ■ Restrict routing other signals close to system reset signals. ■ Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks. ■ Match all signals within a given DQ group with a maximum skew of ±10 ps or approximately ±50 mils (0.254 mm) and route on the same layer.

Table 7–4. RLDRAM II and RLDRAM 3 Layout Guidelines (Part 2 of 2)

Parameter	Guidelines
Clock Routing	■ Route clocks on inner layers with outer-layer run lengths held to under 150 ps (approximately 500 mils, 12.7 mm). ■ These signals should maintain a 10-mil (0.254 mm) spacing from other nets. ■ Clocks should maintain a length-matching between clock pairs of ±5 ps or approximately ±25 mils (0.635 mm). ■ Differential clocks should maintain a length-matching between `P` and `N` signals of ±2 ps or approximately ±10 mils (0.254 mm). ■ Space between different clock pairs should be at least three times the space between the traces of a differential pair.
Address and Command Routing	■ To minimize crosstalk, route address, bank address, and command signals on a different layer than the data and data mask signals. ■ Do not route the differential clock signals close to the address signals. ■ Keep the distance from the pin on the RLDRAM II or RLDRAM 3 component to the stub termination resistor (V_{TT}) to less than 50 ps (approximately 250 mils, 6.35 mm) for the address/command signal group. ■ Keep the distance from the pin on the RLDRAM II or RLDRAM 3 component to the fly-by termination resistor (V_{TT}) to less than 100 ps (approximately 500 mils, 12.7 mm) for the address/command signal group.

External Memory Routing Rules	■ Apply the following parallelism rules for the RLDRAM II or RLDRAM 3 data/address/command groups:
	  ■ 4 mils for parallel runs < 0.1 inch (approximately 1× spacing relative to plane distance).
	  ■ 5 mils for parallel runs < 0.5 inch (approximately 1× spacing relative to plane distance).
	  ■ 10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance).
	  ■ 15 mils for parallel runs between 1.0 and 3.3 inch (approximately 3× spacing relative to plane distance).

Maximum Trace Length	■ Keep the maximum trace length of all signals from the FPGA to the RLDRAM II or RLDRAM 3 components to 600 ps (approximately 3,300 mils, 83.3 mm).

RLDRAM II and RLDRAM 3 Layout Approach

Using the layout guidelines in Table 7–4, Altera recommends the following layout approach:

1. If the RLDRAM II interface has multiple DQ groups (×18 or ×36 RLDRAM II/RLDRAM 3 component or width expansion configuration), match all the `DK/DK#` and `QK, QK#` clocks as tightly as possible to optimize the timing margins in your design.
2. Route the `DK/DK#` write clock and `QK/QK#` read clock associated with a DQ group on the same PCB layer. Match these clock pairs to within ±5 ps.
3. Set the `DK/DK#` or `QK/QK#` clock as the target trace propagation delay for the associated data and data mask signals.
4. Route the data and data mask signals for the DQ group ideally on the same layer as the associated `QK/QK#` and `DK/DK#` clocks to within ±10 ps skew of the target clock.

5. Route the CK/CK# clocks and set as the target trace propagation delays for the address/command signal group. Match the CK/CK# clock to within ±50 ps of all the DK/DK# clocks.
6. Route the address/control signal group (address, bank address, CS, WE, and REF) ideally on the same layer as the CK/CK# clocks, to within ±20 ps skew of the CK/CK# traces.

It is important to match the delays of CK vs. DK, and CK vs. Addr-cmd as much as possible.

This layout approach provides a good starting point for a design requirement of the highest clock frequency supported for the RLDRAM II and RLDRAM 3 interfaces.

For details on pin planning, refer to the Planning Pin and FPGA Resources chapter in the *External Memory Interface Handbook*.

Document Revision History

Table 7–5 lists the revision history for this document.

Table 7–5. Document Revision History

Date	Version	Changes
November 2012	3.2	Added content supporting RLDRAM 3 and updated RLDRAM II standards.
June 2012	3.1	Added Feedback icon.
November 2011	3.0	Added Arria V information.
June 2011	2.0	Added Stratix V information.
December 2010	1.0	Initial release.

8. QDR II SRAM Board Design Guidelines

EMI_DG_007-4.2

This chapter provides guidelines for you to improve your system's signal integrity and layout guidelines to help successfully implement a QDR II or QDR II+ SRAM interface in your system.

The QDR II and QDR II+ SRAM Controller with UniPHY intellectual property (IP) enables you to implement QDR II and QDR II+ interfaces with Arria® II GX, Arria V, Stratix® III, Stratix IV, and Stratix V devices.

In this chapter, QDR II SRAM refers to both QDR II and QDR II+ SRAM unless stated otherwise.

This chapter focuses on the following key factors that affect signal integrity:

- I/O standards
- QDR II SRAM configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

I/O Standards

QDR II SRAM interface signals use one of the following JEDEC I/O signalling standards:

- HSTL-15—provides the advantages of lower power and lower emissions.
- HSTL-18—provides increased noise immunity with slightly greater output voltage swings.

To select the most appropriate standard for your interface, refer to the *Arria II GX Devices Data Sheet: Electrical Characteristics* chapter in the *Arria II Device Handbook*, *Stratix III Device Datasheet: DC and Switching Characteristics* chapter in the *Stratix III Device Handbook*, or the *Stratix IV Device Datasheet DC and Switching Characteristics* chapter in the *Stratix IV Device Handbook*.

Altera® QDR II SRAM Controller with UniPHY IP defaults to HSTL 1.5 V Class I outputs and HSTL 1.5 V inputs.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

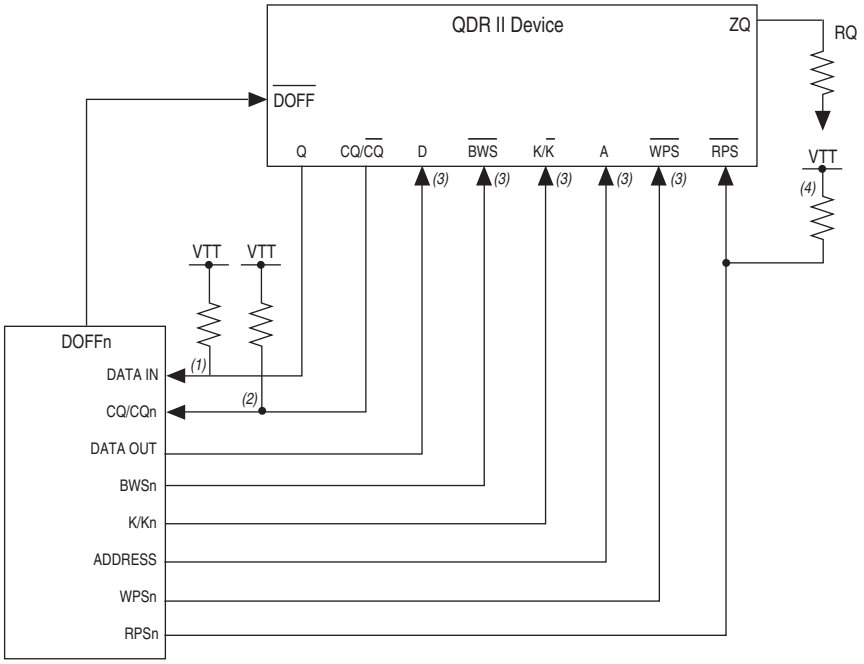
ISO 9001:2008 Registered

QDR II SRAM Configurations

The QDR II SRAM Controller with UniPHY IP supports interfaces with a single device, and two devices in a width expansion configuration up to maximum width of 72 bits.

Figure 8–1 shows the main signal connections between the FPGA and a single QDR II SRAM component.

Figure 8–1. Configuration With A Single QDR II SRAM Component



Notes to Figure 8–1:

(1) Use external discrete termination only for data inputs targeting Arria II GX devices that do not support parallel OCT. For Stratix III and Stratix IV devices, use parallel OCT.

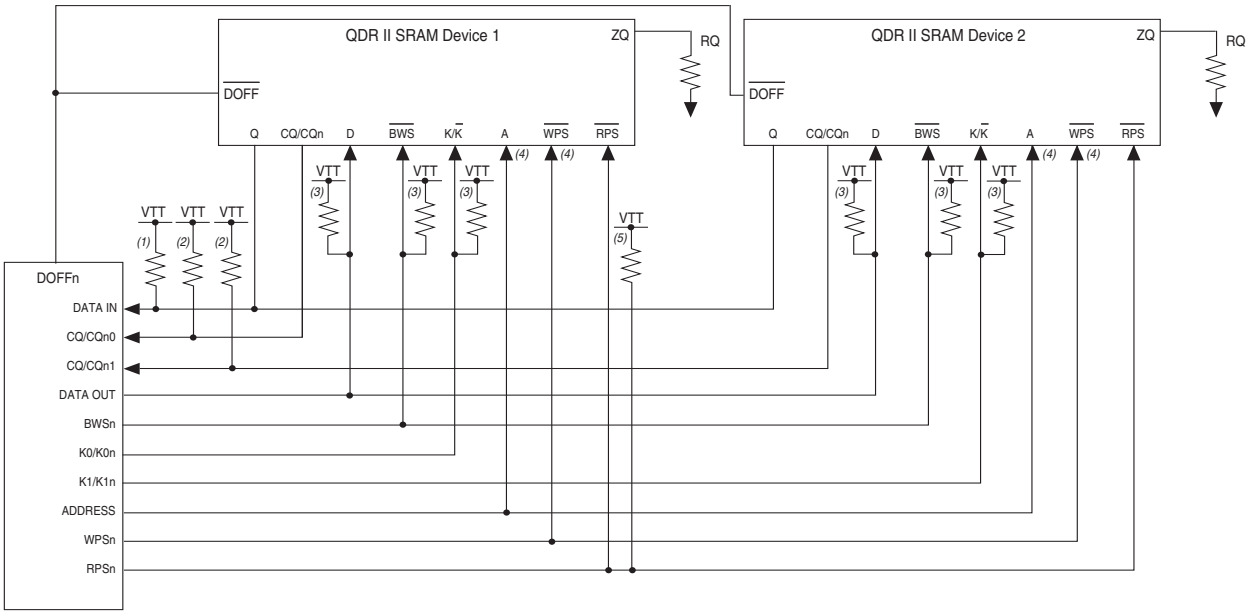
(2) Use external discrete termination only for CQ/CQ# targeting Arria II GX devices, or for any device using ×36 emulated mode.

(3) Use external discrete termination for this signal, as shown for RPS.

(4) Use external discrete termination with fly-by placement to avoid stubs.

Figure 8–2 shows the main signal connections between the FPGA and two QDR II SRAM components in a width expansion configuration.

Figure 8–2. Configuration With Two QDR II SRAM Components In A Width Expansion Configuration



Notes to Figure 8–2:

(1) Use external discrete termination only for data inputs targeting Arria II GX devices that do not support parallel OCT. For Stratix III and Stratix IV devices, use parallel OCT.

(2) Use external discrete termination only for CQ/CQ# targeting Arria II GX devices, or for any device using ×36 emulated mode.

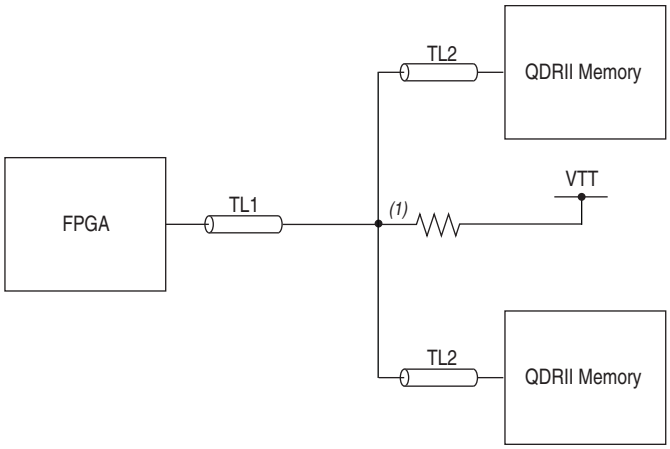
(3) Use external discrete termination for data outputs, BWSn, and K/K# clocks with fly-by placement to avoid stubs.

(4) Use external discrete termination for this signal, as shown for RPS.

(5) Use external discrete termination at the trace split of the balanced T or Y topology.

Figure 8–3 shows the detailed balanced topology recommended for the address and command signals in the width expansion configuration.

Figure 8–3. External Parallel Termination for Balanced Topology



Note to Figure 8–3:

(1) To minimize the reflections and parallel impedance discontinuity seen by the signal, place the trace split close to the QDR II SRAM memory components. Keep TL2 short so that the QDR II SRAM components appear as a lumped load.

Signal Terminations

Arria II GX, Stratix III and Stratix IV devices offer on-chip termination (OCT) technology.

Table 8–1 summarizes the extent of OCT support for each device.

Table 8–1. On-Chip Termination Schemes *(1)*

Termination Scheme	HSTL-15 and HSTL-18	FPGA Device					
		Arria II GX		Arria II GZ, Stratix III, and Stratix IV		Arria V and Stratix V	
		Column I/O	Row I/O	Column I/O	Row I/O	Column I/O	Row I/O
On-Chip Series Termination without Calibration	Class I	50	50	50	50	—	—
On-Chip Series Termination with Calibration	Class I	50	50	50	50	—	—
On-Chip Parallel Termination with Calibration	Class I	—	—	50	50	50	50

Note to Table 8–1:

(1) This table provides information about HSTL-15 and HSTL-18 standards because these are the supported I/O standards for QDR II SRAM memory interfaces by Altera FPGAs.

On-chip series (R_S) termination is supported only on output and bidirectional buffers, while on-chip parallel (R_T) termination is supported only on input and bidirectional buffers. Because QDR II SRAM interfaces have unidirectional data paths, dynamic OCT is not required.

For Arria II GX, Stratix III and Stratix IV devices, the HSTL Class I I/O calibrated terminations are calibrated against 50 Ω 1% resistors connected to the R_{UP} and R_{DN} pins in an I/O bank with the same VCCIO as the QDRII SRAM interface. The calibration occurs at the end of the device configuration.

QDR II SRAM controllers have a ZQ pin which is connected via a resistor RQ to ground. Typically the QDR II SRAM output signal impedance is 0.2 × RQ. Refer to the QDR II SRAM device data sheet for more information.

For information about OCT, refer to the *I/O Features in Arria II GX Devices* chapter in the *Arria II GX Device Handbook*, *I/O Features in Arria V Devices* chapter in the *Arria V Device Handbook*, *Stratix III Device I/O Features* chapter in the *Stratix III Device Handbook*, *I/O Features in Stratix IV Devices* chapter in the *Stratix IV Device Handbook, and the I/O Features in Stratix V Devices* chapter in the *Stratix V Device Handbook*

The following section shows HyperLynx simulation eye diagrams to demonstrate signal termination options. Altera strongly recommends signal terminations to optimize signal integrity and timing margins, and to minimize unwanted emissions, reflections, and crosstalk.

All of the eye diagrams shown in this section are for a 50 Ω trace with a propagation delay of 720 ps which is approximately a 4-inch trace on a standard FR4 PCB. The signal I/O standard is HSTL-15.

For point-to-point signals, Altera recommends that you place a fly-by termination by terminating at the end of the transmission line after the receiver to avoid unterminated stubs. The guideline is to place the fly-by termination within 100 ps propagation delay of the receiver.

Although not recommended, you can place the termination before the receiver, which leaves an unterminated stub. The stub delay is critical because the stub between the termination and the receiver is effectively unterminated, causing additional ringing and reflections. Stub delays should be less than 50 ps.

The eye diagrams shown in this section show the best case achievable and do not take into account PCB vias, crosstalk and other degrading effects such as variations in the PCB structure due to manufacturing tolerances.

Simulate your design to ensure correct functionality.

Output from the FPGA to the QDR II SRAM Component

The following output signals are from the FPGA to the QDR II SRAM component:

- write data
- byte write select (`BWSn`)
- address
- control (`WPSn` and `RPSn`)
- clocks, `K/K#`

Altera recommends that you terminate the write clocks, K and K#, with a single-ended fly-by 50 Ω parallel termination to V_{TT}. However, simulations show that you can consider a differential termination if the clock pair is well matched and routed differentially.

The HyperLynx simulation eye diagrams show simulation cases of write data and address signals with termination options. The QDR II SRAM write data is double data rate. The QDR II SRAM address is either double data rate (burst length of 2) or single data rate (burst length of 4).

Simulations show that lowering the drive strength does not make a significant difference to the eye diagrams. All eye diagrams are shown at the QDR II SRAM device receiver pin.

Figure 8–4 shows the fly-by terminated signal using Stratix IV Class I HSTL-15 with calibrated 50 Ω OCT output driver.

Figure 8–4. Write Data Simulation at 400 MHz with Fly-By 50 Ω Parallel Termination to V_{TT}

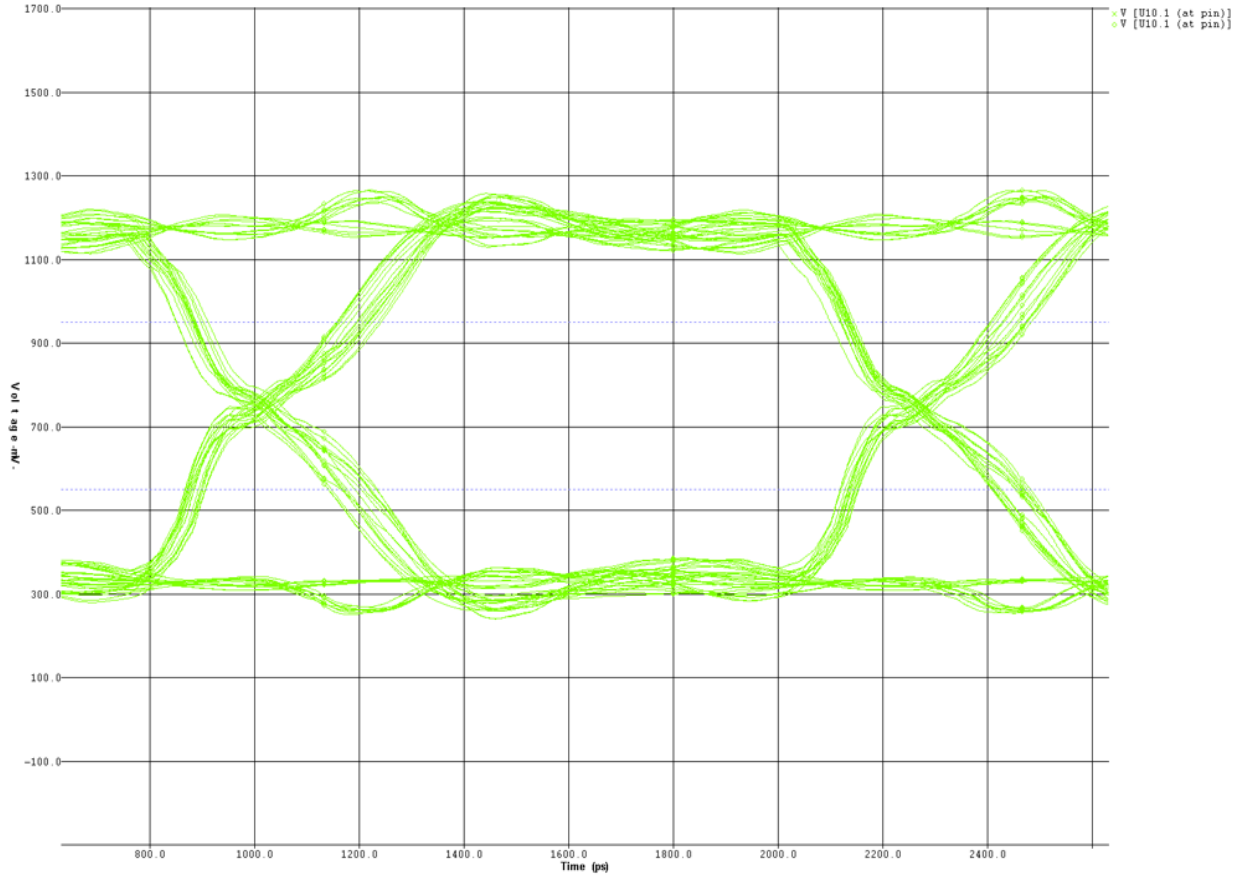


Figure 8–5 shows an unterminated signal using Stratix IV Class I HSTL-15 with a calibrated 50 Ω OCT output driver. This unterminated solution is not recommended.

Figure 8–5. Write Data Simulation at 400 MHz with No Far-End Termination

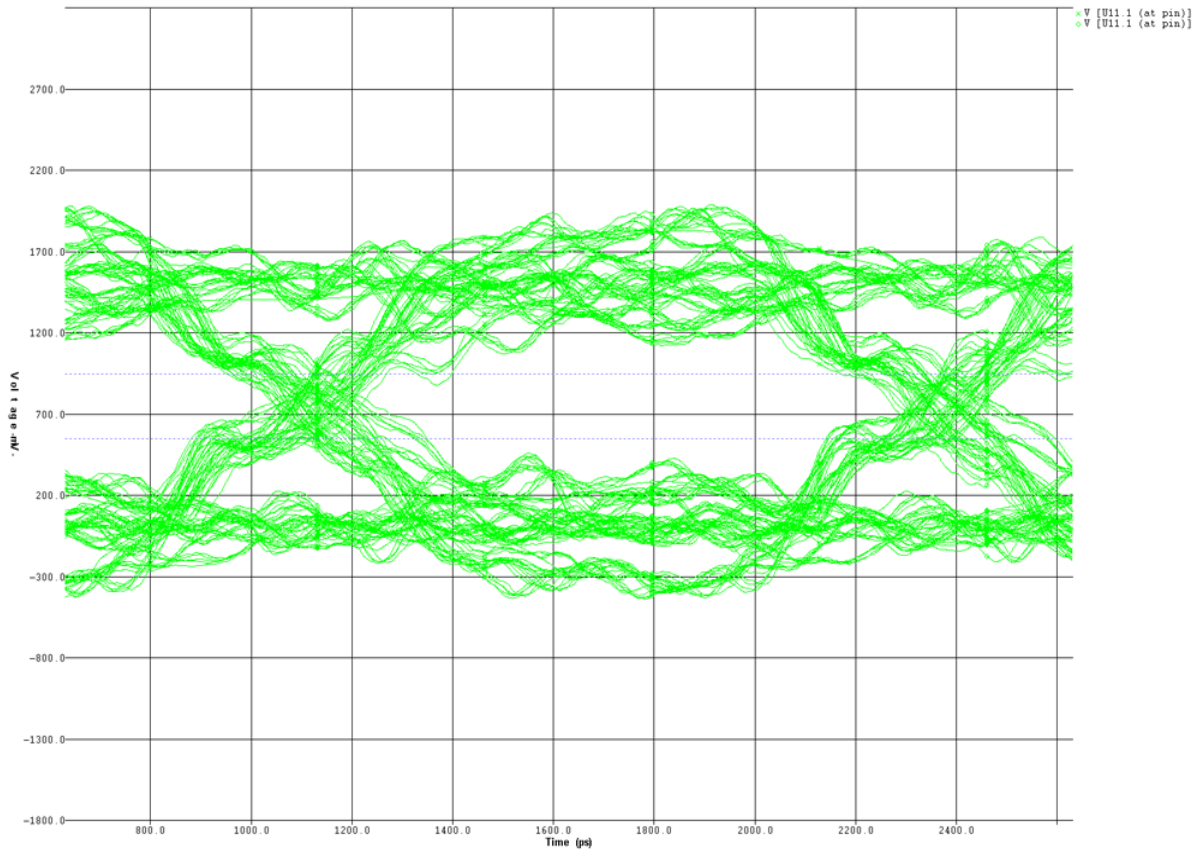


Figure 8–6 shows an unterminated signal at a lower frequency of 250 MHz using Arria II GX Class I HSTL-15 with calibrated 50 Ω OCT output driver. This unterminated solution may be passable for some systems, but is shown so that you can compare against the superior quality of the terminated signal in Figure 8–4.

Figure 8–6. Write Data Simulation at 250 MHz with No Far-End Termination

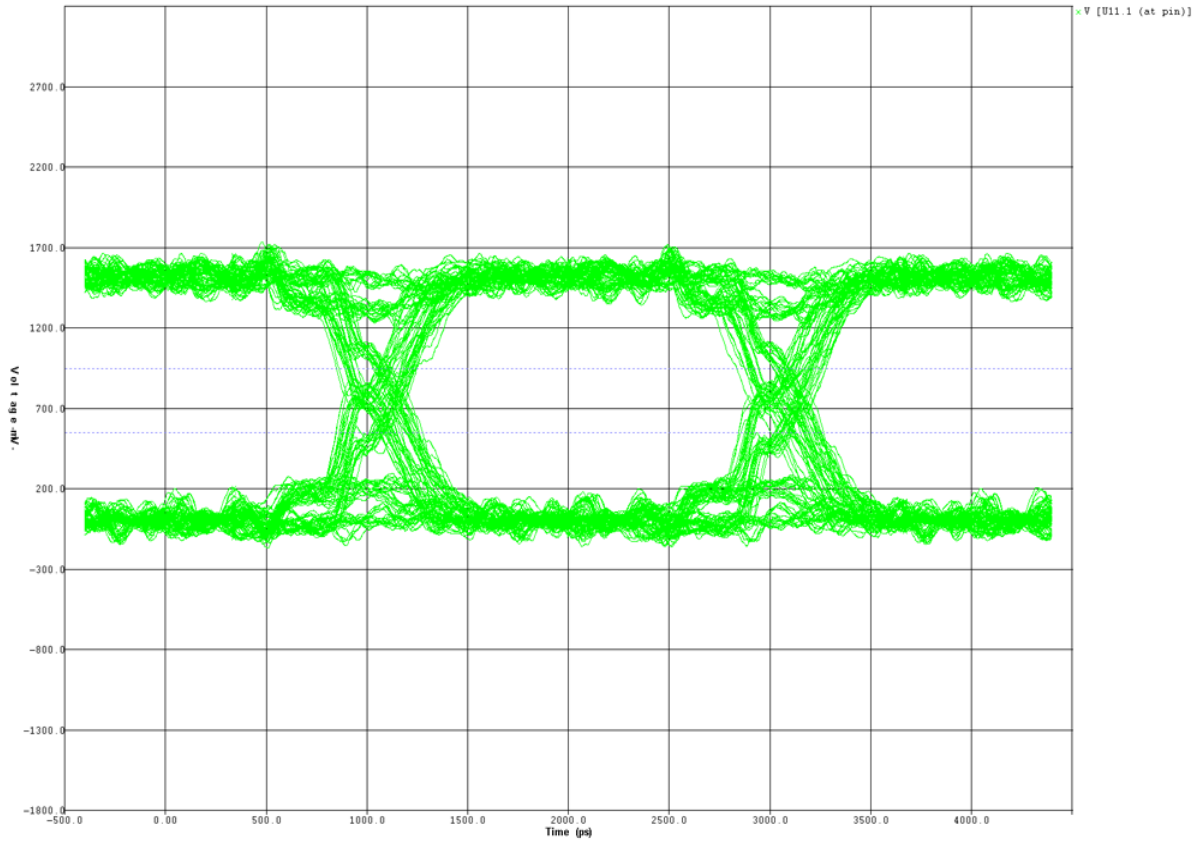


Figure 8–7 shows an unterminated signal at a frequency of 175 MHz with a point-to-point connection. QDR II SRAM interfaces using Stratix IV devices have a maximum supported frequency of 350 MHz. For QDR II SRAM with burst length of four interfaces, the address signals are effectively single date rate at 175 MHz. This unterminated solution is not recommended but can be considered. The FPGA output driver is Class I HSTL-15 with a calibrated 50 Ω OCT.

Figure 8–7. Address Simulation for QDR II SRAM Burst Length of 4 at 175 MHz with No Far-End Termination

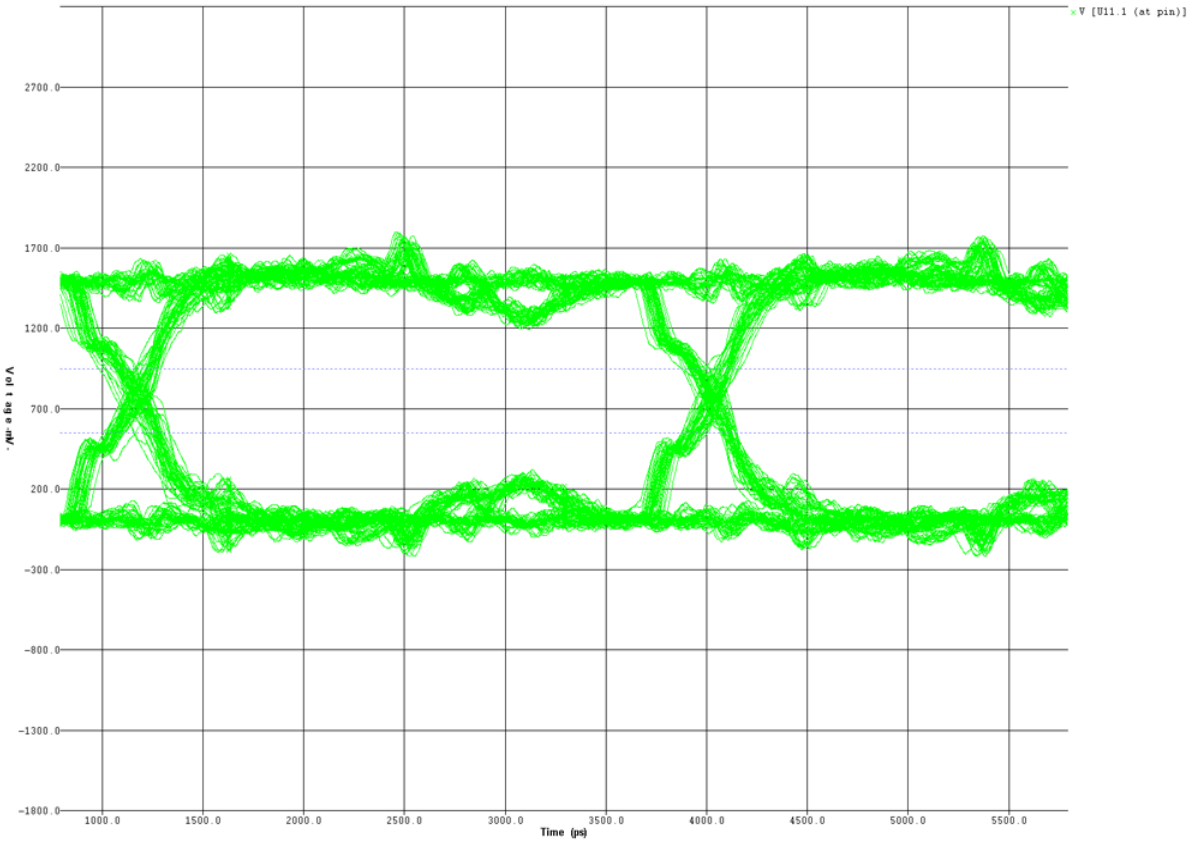
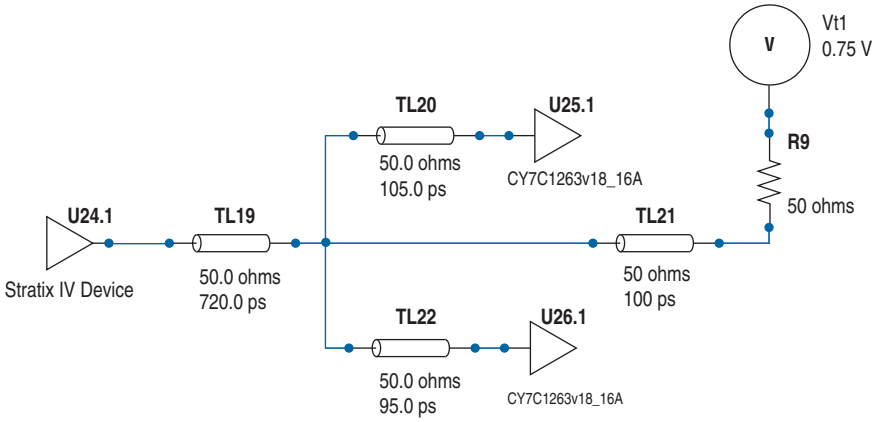


Figure 8–8 shows a typical topology, which are used for two components in width expansion mode. Altera recommends that you match the stubs TL20 and TL22, but you can allow small differences allowed to achieve acceptable signal integrity.

Figure 8–8. Address for QDR II SRAM Burst Length of 2 in Width Expansion Mode Topology



The eye diagrams in Figure 8–9 and Figure 8–10 use the topology shown in Figure 8–8. The eye diagram in Figure 8–11 uses the topology shown in Figure 8–8 without the V_{TT} termination, R9 and TL21.

Figure 8–9 shows an address signal at a frequency of 400 MHz with parallel 50 Ω termination to V_{TT} for QDR II SRAM burst length of 2 width expansion using Stratix IV Class I HSTL-15 12 mA driver and fly-by 50 Ω parallel termination to V_{TT}.

Figure 8–9. Address Simulation Using Stratix IV Class I HSTL-15 12 mA Driver and Fly-by 50 Ω Parallel Termination to V_{TT}

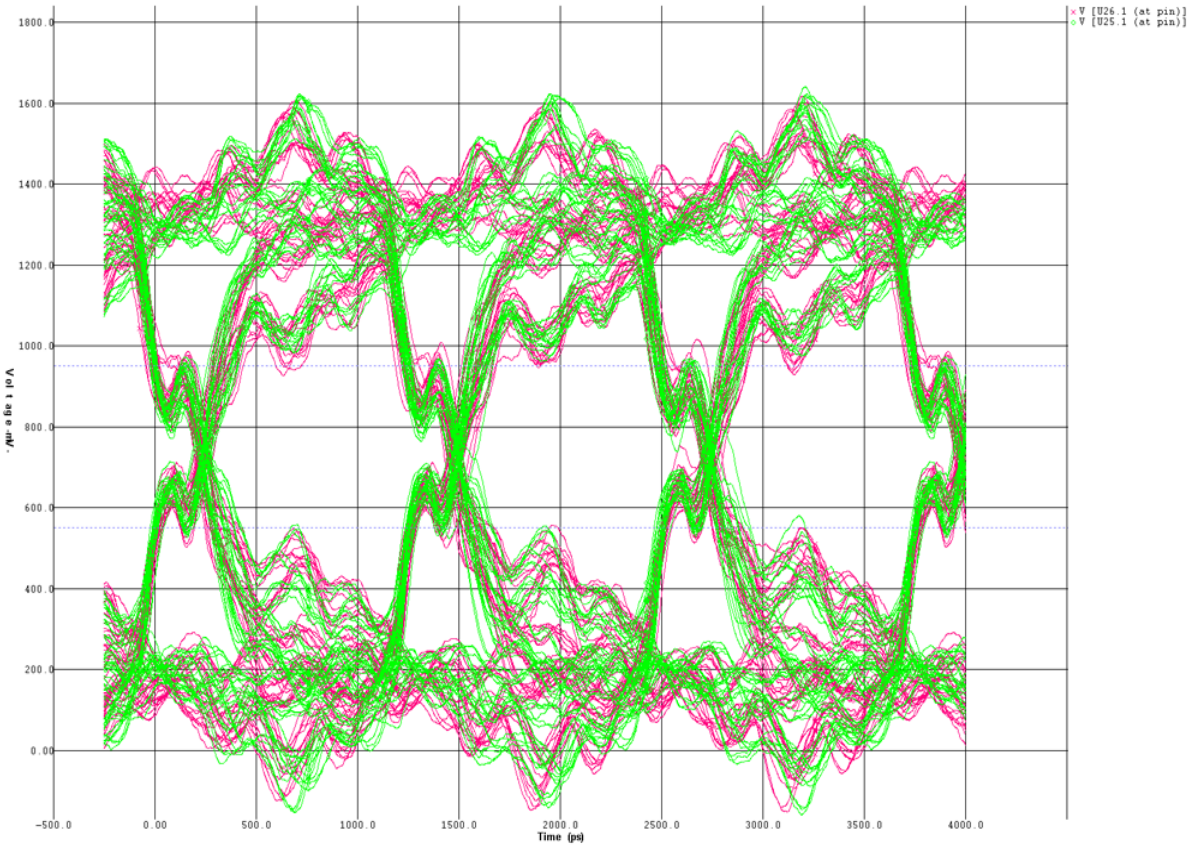


Figure 8–10 shows an address signal at a frequency of 400 MHz with parallel 50 Ω termination to V_{TT} for QDR II SRAM burst length of 2 width expansion using Stratix IV Class I HSTL-15 with 50 Ω calibration driver and fly-by 50 Ω parallel termination to V_{TT}. The waveform eye is significantly improved compared to the maximum (12mA) drive strength case.

Figure 8–10. Address Simulation Using Stratix IV Class I HSTL-15 50 Ω Calibration Driver and Fly-by 50 Ω Parallel Termination to V_{TT}

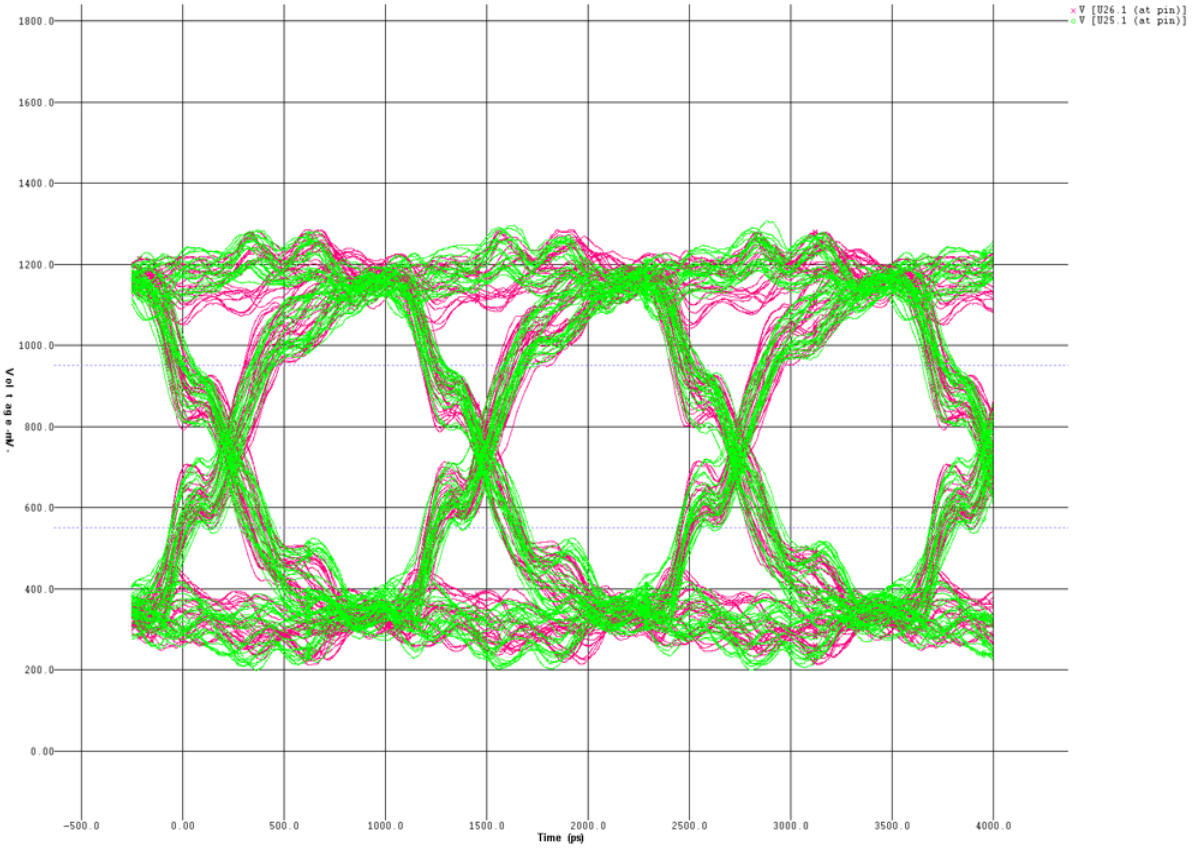
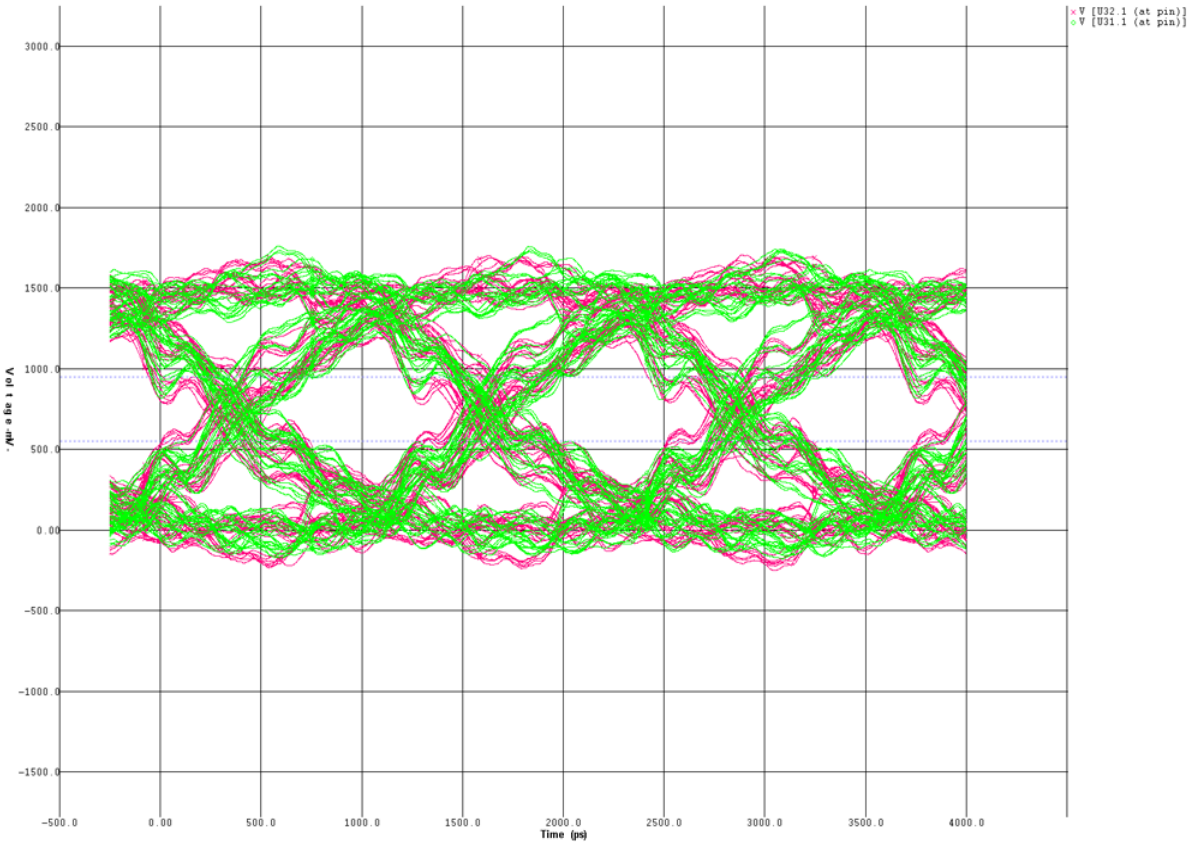


Figure 8–11 shows an unterminated address signal at a frequency of 400 MHz for QDR II SRAM burst length of 2 width expansion using Stratix IV Class I HSTL-15 with 50 Ω calibration driver. This unterminated address has small eye and is not recommended.

Figure 8–11. Address Simulation Using Stratix IV Class I HSTL-15 50 Ω Calibration Driver and No Termination



Input to the FPGA from the QDR II SRAM Component

The QDR II SRAM component drives the following input signals into the FPGA:

- read data
- echo clocks, CQ/CQ#

For point-to-point signals, Altera recommends that you use the FPGA parallel OCT wherever possible. For devices that do not support parallel OCT (Arria II GX), and for ×36 emulated configuration CQ/CQ# termination, Altera recommends that you use a fly-by 50 Ω parallel termination to V_{TT}. Although not recommended, you can use parallel termination with a short stub of less that 50 ps propagation delay as an alternative option. The input echo clocks, CQ and CQ# must not use a differential termination.

The eye diagrams are shown at the FPGA receiver pin, and the QDR II SRAM output driver is Class I HSTL-15 using its ZQ calibration of 50 Ω The QDR II SRAM read data is double data rate.

Figure 8–12 shows the ideal case of a fly-by terminated signal using 50 Ω calibrated parallel OCT with Stratix IV device.

Figure 8–12. Read Data Simulation at 400 MHz with 50 Ω Parallel OCT Termination

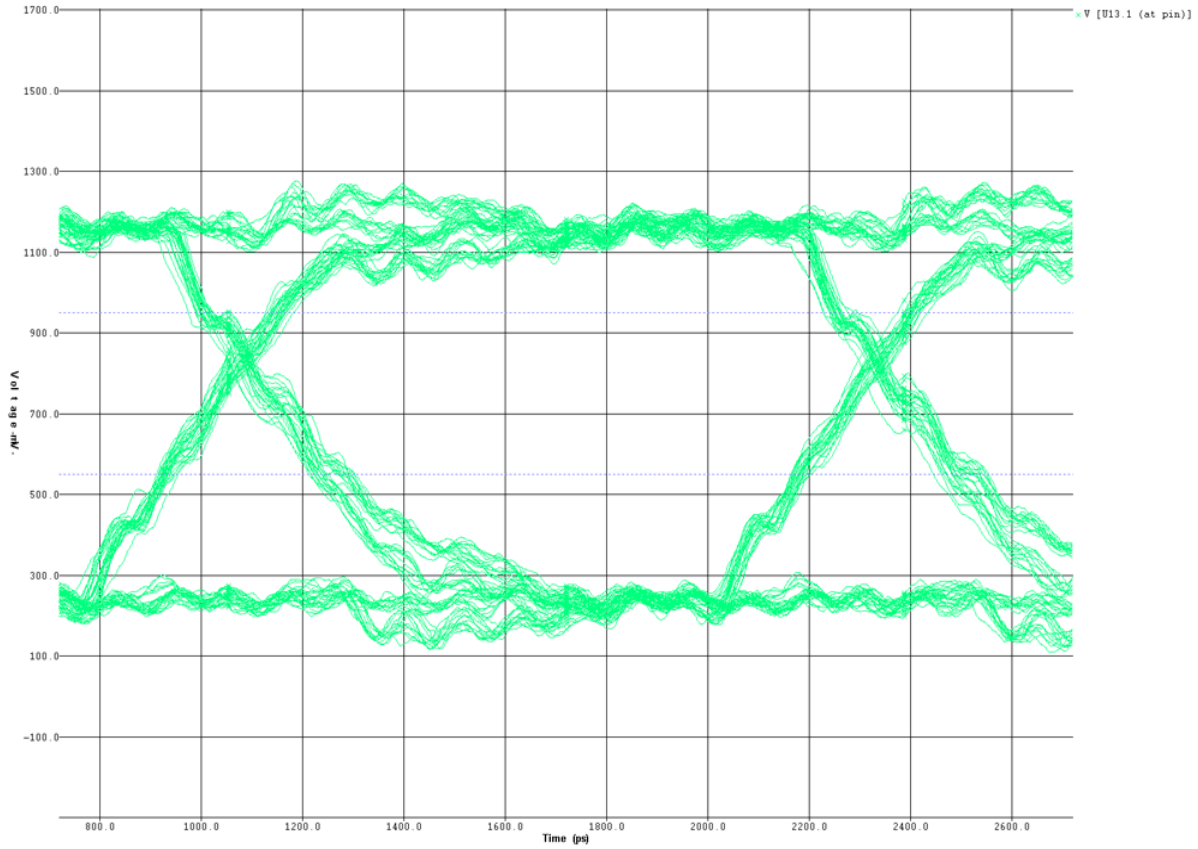


Figure 8–13 shows an external discrete component fly-by terminated signal at a lower frequency of 250 MHz using an Arria II GX device.

Figure 8–13. Read Data Simulation at 250 MHz with Fly-By Parallel 50 Ω Termination

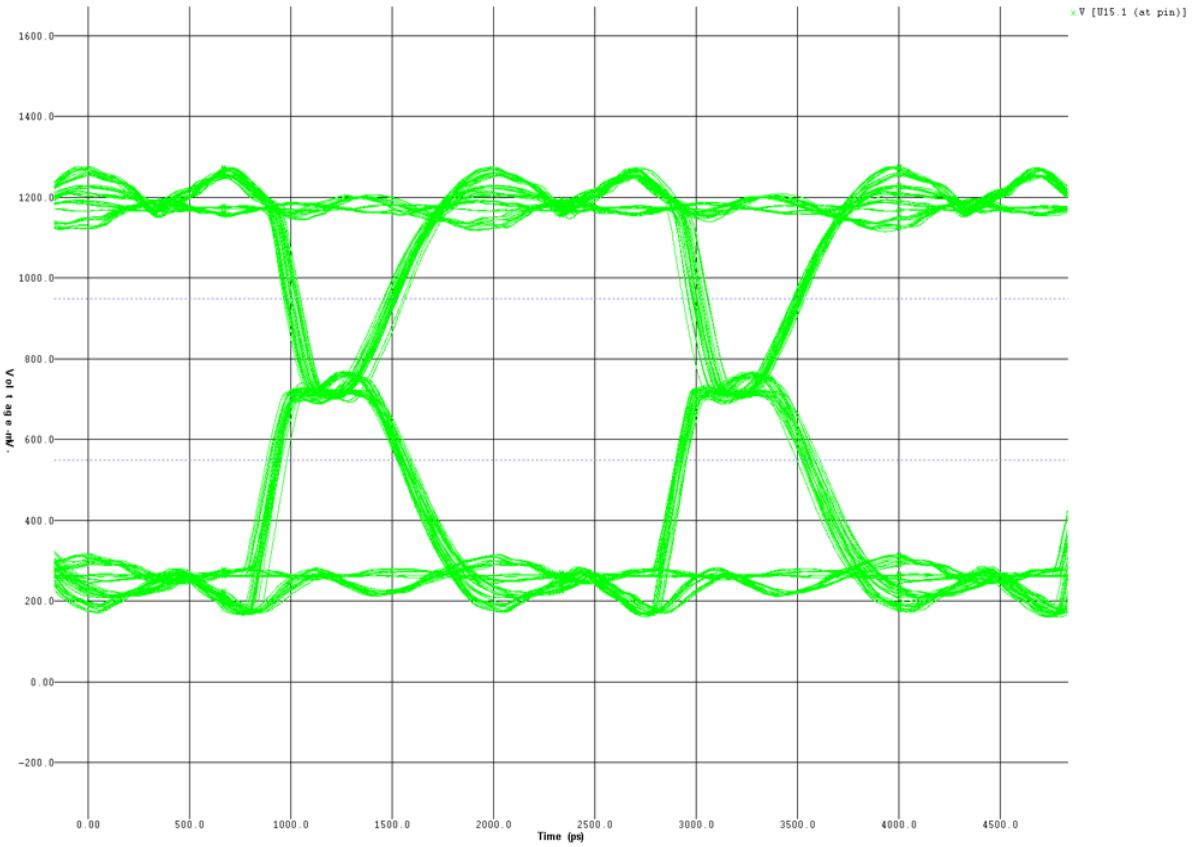
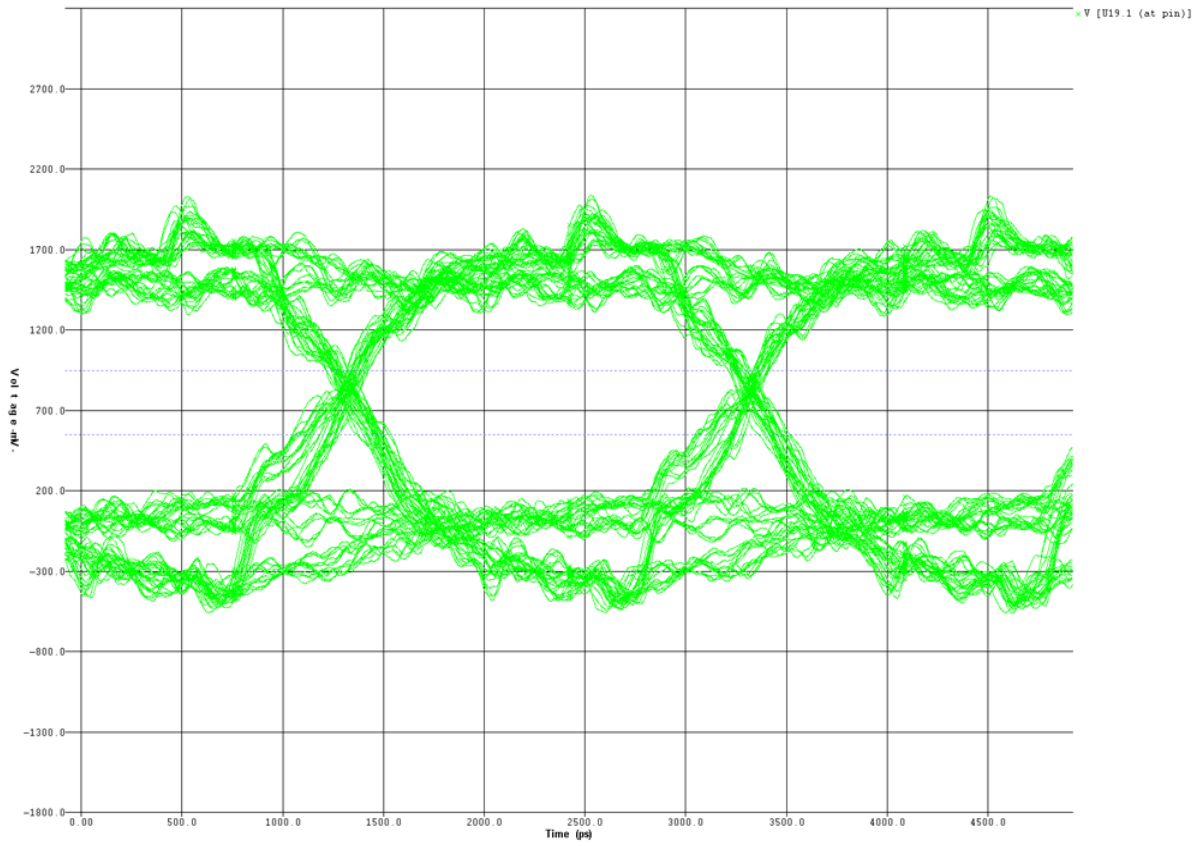


Figure 8–14 shows an unterminated signal at a lower frequency of 250 MHz using an Arria II GX device. This unterminated solution is not recommended but is shown so that you can compare against the superior quality of the terminated signal in Figure 8–13.

Figure 8–14. Read Data Simulation at 250 MHz with No Far-End Termination



Termination Schemes

Table 8–2 and Table 8–3 list the recommended termination schemes for major QDR II SRAM memory interface signals, which include write data (D), byte write select (BWS), read data (Q), clocks (K, K#, CQ, and CQ#), address and command (WPS and RPS).

Table 8–2. Termination Recommendations for Arria II GX Devices

Signal Type	HSTL 15/18 Standard (1), (2)	FPGA End Discrete Termination	Memory End Termination
K/K# Clocks	Class I R50 CAL	—	50 Ω Parallel to V_{TT}
Write Data	Class I R50 CAL	—	50 Ω Parallel to V_{TT}
BWS	Class I R50 CAL	—	50 Ω Parallel to V_{TT}
Address (3), (4)	Class I Max Current	—	50 Ω Parallel to V_{TT}
WPS, RPS (3), (4)	Class I Max Current	—	50 Ω Parallel to V_{TT}
CQ/CQ#	Class I	50Ω Parallel to V_{TT}	ZQ50
CQ/CQ# ×36 emulated (5)	Class I	50 Ω Parallel to V_{TT}	ZQ50
Read Data (Q)	Class I	50 Ω Parallel to V_{TT}	ZQ50
QVLD (6)	—	—	ZQ50

Notes to Table 8–2:

(1) R is effective series output impedance.

(2) CAL is calibrated OCT.

(3) For width expansion configuration, the address and control signals are routed to 2 devices. Recommended termination is 50 Ω parallel to V_{TT} at the trace split of a balanced T or Y routing topology. For 400 MHz burst length 2 configurations where the address signals are double data rate, it is recommended to use a clamshell placement of the two QDR II SRAM components to achieve minimal stub delays and optimum signal integrity. Clamshell placement is when two devices overlay each other by being placed on opposite sides of the PCB.

(4) A Class I 50 Ω output with calibration output is typically optimal in double load topologies.

(5) For ×36 emulated mode, the recommended termination for the CQ/CQ# signals is a 50 Ω parallel termination to V_{TT} at the trace split, refer to Figure 8–15. Altera recommends that you use this termination when ×36 DQ/DQS groups are not supported in the FPGA.

(6) QVLD is not used in the QDR II or QDR II+ SRAM with UniPHY implementations.

Table 8–3. Termination Recommendations for Arria V, Stratix III, Stratix IV, and Stratix V Devices (Part 1 of 2)

Signal Type	HSTL 15/18 Standard (1), (2), (3)	FPGA End Discrete Termination	Memory End Termination
K/K# Clocks	Class I R50 CAL	—	50 Ω Parallel to V_{TT}
Write Data	Class I R50 CAL	—	50 Ω Parallel to V_{TT}
BWS	Class I R50 CAL	—	50 Ω Parallel to V_{TT}
Address (4), (5)	Class I Max Current	—	50 Ω Parallel to V_{TT}
WPS, RPS (4), (5)	Class I Max Current	—	50 Ω Parallel to V_{TT}
CQ/CQ#	Class I P50 CAL	—	ZQ50
CQ/CQ# ×36 emulated (6)	—	50 Ω Parallel to V_{TT}	ZQ50
Read Data (Q)	Class I P50 CAL	—	ZQ50

Table 8–3. Termination Recommendations for Arria V, Stratix III, Stratix IV, and Stratix V Devices (Part 2 of 2)

Signal Type	HSTL 15/18 Standard (1), (2), (3)	FPGA End Discrete Termination	Memory End Termination
QVLD (7)	Class I P50 CAL	—	ZQ50

Notes to Table 8–3:

(1) R is effective series output impedance.

(2) P is effective parallel input impedance.

(3) CAL is calibrated OCT.

(4) For width expansion configuration, the address and control signals are routed to 2 devices. Recommended termination is 50 Ω parallel to V_{TT} at the trace split of a balanced T or Y routing topology. For 400 MHz burst length 2 configurations where the address signals are double data rate, it is recommended to use a "clam shell" placement of the two QDR II SRAM components to achieve minimal stub delays and optimum signal integrity. "Clam shell" placement is when two devices overlay each other by being placed on opposite sides of the PCB.

(5) The UniPHY default IP setting for this output is Max Current. A Class 1 50 Ω output with calibration output is typically optimal in single load topologies.

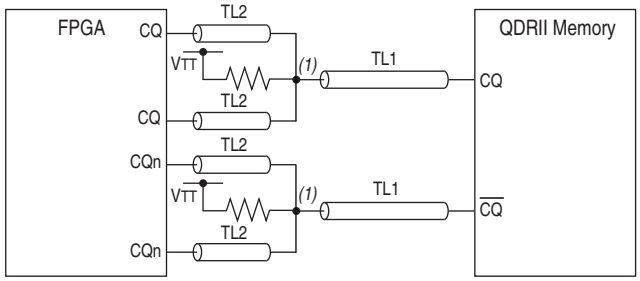
(6) For ×36 emulated mode, the recommended termination for the CQ/CQ# signals is a 50 Ω parallel termination to V_{TT} at the trace split, refer to Figure 8–15. Altera recommends that you use this termination when ×36 DQ/DQS groups are not supported in the FPGA.

(7) QVLD is not used in the QDR II or QDR II+ SRAM Controller with UniPHY implementations.

Altera recommends that you simulate your specific design for your system to ensure good signal integrity.

For a ×36 QDR II SRAM interface that uses an emulated mode of two ×18 DQS groups in the FPGA, there are two CQ/CQ# connections at the FPGA and a single CQ/CQ# output from the QDR II SRAM device. Altera recommends that you use a balanced T topology with the trace split close to the FPGA and a parallel termination at the split, as shown in Figure 8–15.

Figure 8–15. Emulated ×36 Mode CQ/CQn Termination Topology



Note to Figure 8–15:

(1) To minimize the reflections and parallel impedance discontinuity seen by the signal, place the trace split close to the FPGA device. Keep TL2 short so that the FPGA inputs appear as a lumped load.

For more information about ×36 emulated modes, refer to the "Exceptions for ×36 Emulated QDR II and QDR II+ SRAM Interfaces in Arria II GX, Stratix III, and Stratix IV Devices" section in the *Planning Pin and Resource* chapter.

PCB Layout Guidelines

Table 8–4 summarizes QDR II and QDR II SRAM general routing layout guidelines.

The following layout guidelines include several +/- length based rules. These length based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristics of your PCB implementation. They do not include any margin for crosstalk.

Altera recommends that you get accurate time base skew numbers when you simulate your specific implementation.

Table 8–4. QDR II and QDR II+ SRAM Layout Guidelines (Part 1 of 2)

Parameter	Guidelines
Impedance	■ All signal planes must be 50 Ω, single-ended, ±10%. ■ All signal planes must be 100 Ω, differential ±10%. ■ Remove all unused via pads, because they cause unwanted capacitance.
Decoupling Parameter	■ Use 0.1 µF in 0402 size to minimize inductance. ■ Make V_{TT} voltage decoupling close to pull-up resistors. ■ Connect decoupling caps between V_{TT} and ground. ■ Use a 0.1 µF cap for every other V_{TT} pin. ■ Verify your capacitive decoupling using the Altera Power Distribution Network (PDN) Design tool.
Power	■ Route GND, 1.5 V/1.8 V as planes. ■ Route V_{CCIO} for memories in a single split plane with at least a 20-mil (0.020 inches or 0.508 mm) gap of separation. ■ Route V_{TT} as islands or 250-mil (6.35-mm) power traces. ■ Route all oscillators and PLL power as islands or 100-mil (2.54-mm) power traces.
General Routing	■ All specified delay matching requirements include PCB trace delays, different layer propagation, velocity variance, and crosstalk. To minimize PCB layer propagation variance, Altera recommends that signals from the same net group always be routed on the same layer. If signals of the same net group must be routed on different layers with the same impedance characteristic, you must simulate your worst case PCB trace tolerances to ascertain actual propagation delay differences. Typical later to later trace delay variations are of 15 ps/inch order. ■ Use 45° angles (not 90° corners). ■ Avoid T-Junctions for critical nets or clocks. ■ Avoid T-junctions greater than 150 ps (approximately 500 mils, 12.7 mm). ■ Disallow signals across split planes. ■ Restrict routing other signals close to system reset signals. ■ Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks.

Table 8–4. QDR II and QDR II+ SRAM Layout Guidelines (Part 2 of 2)

Parameter	Guidelines
Clock Routing	■ Route clocks on inner layers with outer-layer run lengths held to under 150 ps (approximately 500 mils, 12.7 mm). ■ These signals should maintain a 10-mil (0.254 mm) spacing from other nets. ■ Clocks should maintain a length-matching between clock pairs of ±5 ps or approximately ±25 mils (0.635 mm). ■ Complementary clocks should maintain a length-matching between P and N signals of ±2 ps or approximately ±10 mils (0.254 mm). ■ Keep the distance from the pin on the QDR II SRAM component to stub termination resistor (V_{TT}) to less than 50 ps (approximately 250 mils, 6.35 mm) for the K, K# clocks. ■ Keep the distance from the pin on the QDR II SRAM component to fly-by termination resistor (V_{TT}) to less than 100 ps (approximately 500 mils, 12.7 mm) for the K, K# clocks. ■ Keep the distance from the pin on the FPGA component to stub termination resistor (V_{TT}) to less than 50 ps (approximately 250 mils, 6.35 mm) for the echo clocks, CQ, CQ#, if they require an external discrete termination. ■ Keep the distance from the pin on the FPGA component to fly-by termination resistor (V_{TT}) to less than 100 ps (approximately 500 mils, 12.7 mm) for the echo clocks, CQ, CQ#, if they require an external discrete termination.

External Memory Routing Rules	■ Keep the distance from the pin on the QDR II SRAM component to stub termination resistor (V_{TT}) to less than 50 ps (approximately 250 mils, 6.35 mm) for the write data, byte write select and address/command signal groups.
	■ Keep the distance from the pin on the QDR II SRAM component to fly-by termination resistor (V_{TT}) to less than 100 ps (approximately 500 mils, 12.7 mm) for the write data, byte write select and address/command signal groups.
	■ Keep the distance from the pin on the FPGA (Arria II GX) to stub termination resistor (V_{TT}) to less than 50 ps (approximately 250 mils, 6.35 mm) for the read data signal group.
	■ Keep the distance from the pin on the FPGA (Arria II GX) to fly-by termination resistor (V_{TT}) to less than 100 ps (approximately 500 mils, 12.7 mm) for the read data signal group.
	■ Parallelism rules for the QDR II SRAM data/address/command groups are as follows:
	  ■ 4 mils for parallel runs < 0.1 inch (approximately 1× spacing relative to plane distance).
	  ■ 5 mils for parallel runs < 0.5 inch (approximately 1× spacing relative to plane distance).
	  ■ 10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance).
	  ■ 15 mils for parallel runs between 1.0 and 6.0 inch (approximately 3× spacing relative to plane distance).

Maximum Trace Length	■ Keep the maximum trace length of all signals from the FPGA to the QDR II SRAM components to 6 inches.

Using the layout guidelines in Table 8–4, Altera recommends the following layout approach:

1. Route the K/K# clocks and set the clocks as the target trace propagation delays for the output signal group.
2. Route the write data output signal group (write data, byte write select), ideally on the same layer as the K/K# clocks, to within ±10 ps skew of the K/K# traces.

3. Route the address/control output signal group (address, RPS, WPS), ideally on the same layer as the K/K# clocks, to within ±20 ps skew of the K/K# traces.
4. Route the CQ/CQ# clocks and set the clocks as the target trace propagation delays for the input signal group.
5. Route the read data output signal group (read data), ideally on the same layer as the CQ/CQ# clocks, to within ±10 ps skew of the CQ/CQ# traces.
6. The output and input groups do not need to have the same propagation delays, but they must have all the signals matched closely within the respective groups.

Table 8–5 and Table 8–6 list the typical margins for QDR II and QDR II+ SRAM interfaces, with the assumption that there is zero skew between the signal groups.

Table 8–5. Typical Worst Case Margins for QDR II SRAM Interfaces of Burst Length 2

Device	Speed Grade	Frequency (MHz)	Typical Margin Address/Command (ps)	Typical Margin Write Data (ps)	Typical Margin Read Data (ps)
Arria II GX	I5	250	± 240	± 80	± 170
Arria II GX ×36 emulated	I5	200	± 480	± 340	± 460
Stratix IV	—	350	—	—	—
Stratix IV ×36 emulated	C2	300	± 320	± 170	± 340

Table 8–6. Typical Worst Case Margins for QDR II+ SRAM Interfaces of Burst Length 4

Device	Speed Grade	Frequency (MHz)	Typical Margin Address/Command (ps) (1)	Typical Margin Write Data (ps)	Typical Margin Read Data (ps)
Arria II GX	I5	250	± 810	± 150	± 130
Arria II GX ×36 emulated	I5	200	± 1260	± 410	± 420
Stratix IV	C2	400	± 550	± 10	± 80
Stratix IV ×36 emulated	C2	300	± 860	± 180	± 300

Note to Table 8–6:

(1) The QDR II+ SRAM burst length of 4 designs have greater margins on the address signals because they are single data rate.

Other devices and speed grades typically show higher margins than the ones in Table 8–5 and Table 8–6.

Altera recommends that you create your project with a fully implemented QDR II or QDR II+ SRAM Controller with UniPHY interface, and observe the interface timing margins to determine the actual margins for your design.

Although the recommendations in this chapter are based on simulations, you can apply the same general principles when determining the best termination scheme, drive strength setting, and loading style to any board designs. Even armed with this knowledge, it is still critical that you perform simulations, either using IBIS or HSPICE models, to determine the quality of signal integrity on your designs.

Document Revision History

Table 8–7 lists the revision history for this document.

Table 8–7. Document Revision History

Date	Version	Changes
November 2012	4.2	Changed chapter number from 7 to 8.
June 2012	4.1	Changed note to Table 8–2. Added Feedback icon.
November 2011	4.0	Added Arria V information.
June 2011	3.0	Added Stratix V information.
December 2010	2.0	Maintenance update.
July 2010	1.0	Initial release.

9. Implementing and Parameterizing Memory IP

EMI_DG_008-6.0

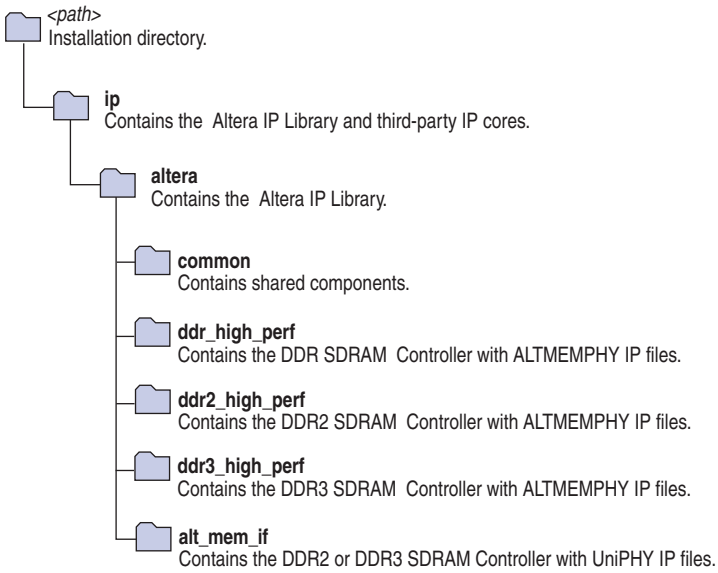
This chapter describes the general overview of the Altera® IP core design flow to help you quickly get started with any Altera IP core. The Altera IP Library is installed as part of the Quartus® II installation process.You can select and parameterize any Altera IP core from the library. Altera provides an integrated parameter editor that allows you to customize IP cores to support a wide variety of applications. The parameter editor guides you through the setting of parameter values and selection of optional ports. The following section describes the general design flow and use of Altera IP cores.

Installation and Licensing

The Altera IP Library is distributed, downloaded, and installed with the Quartus II software from the Altera website (www.altera.com).

Figure 9–1 shows the directory structure after you install the memory controller with the memory IP, where <*path*> is the installation directory. The default installation directory on Windows is **c:\altera**\<*version*>; on Linux it is **/opt/altera**<*version*>.

Figure 9–1. Directory Structure



© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

You can evaluate an IP core in simulation and in hardware until you are satisfied with its functionality and performance. Some IP cores require that you purchase a license for the IP core when you want to take your design to production. After you purchase a license for an Altera IP core, you can generate a license file from the Altera Licensing page of the Altera website and install the license on your computer. For additional information, refer to *Altera Software Installation and Licensing*.

Free Evaluation

Altera's OpenCore Plus evaluation feature is only applicable to the DDR, DDR2 and DDR3 SDRAM memory controllers. No license key or fee is required for Altera memory controllers implemented as hard IP in Altera devices. With the OpenCore Plus evaluation feature, you can perform the following actions:

- Simulate the behavior of a megafunction (Altera MegaCore® function or AMPP^{SM} megafunction) within your system.
- Verify the functionality of your design, as well as evaluate its size and speed quickly and easily.
- Generate time-limited device programming files for designs that include MegaCore functions.
- Program a device and verify your design in hardware.

You need to purchase a license for the MegaCore only when you are completely satisfied with its functionality and performance, and want to take your design to production.

OpenCore Plus Time-Out Behavior

OpenCore Plus hardware evaluation can support the following two modes of operation:

- Untethered—the design runs for a limited time
- Tethered—requires a connection between your board and the host computer. If tethered mode is supported by all megafunctions in a design, the device can operate for a longer time or indefinitely

All megafunctions in a device time out simultaneously when the most restrictive evaluation time is reached. If there is more than one megafunction in a design, a specific megafunction's time-out behavior may be masked by the time-out behavior of the other megafunctions.

For MegaCore functions, the untethered time-out is 1 hour; the tethered time-out value is indefinite.

Your design stops working after the hardware evaluation time expires and the `local_ready` output goes low.

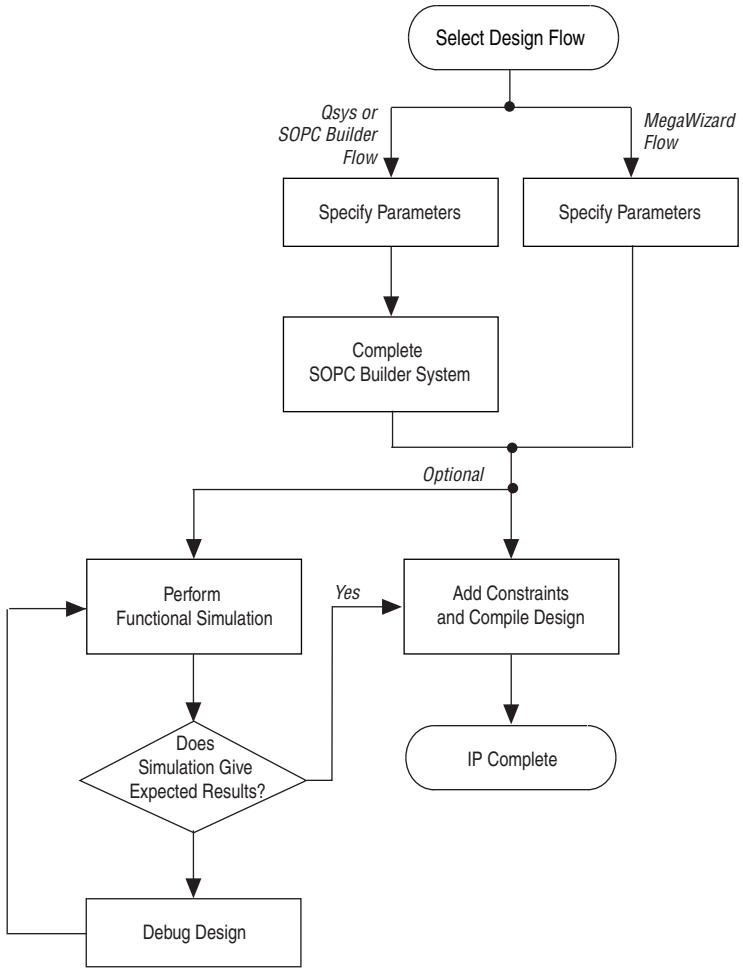
Design Flow

You can implement the memory controllers with ALTMEMPHY IP or UniPHY IP using any of the following flows:

- MegaWizard™ Plug-In Manager flow
- SOPC Builder flow
- Qsys Flow

Figure 9–2 shows the stages for creating a system in the Quartus II software using the available flows.

Figure 9–2. Design Flows *(1)*



Note to Figure 9–2:

(1) Altera IP cores may or may not support the Qsys and SOPC Builder design flows.

The MegaWizard Plug-In Manager flow offers the following advantages:

- Allows you to parameterize an IP core variant and instantiate into an existing design
- For some IP cores, this flow generates a complete example design and testbench

The SOPC Builder flow offers the following advantages:

- Generates simulation environment
- Allows you to integrate Altera-provided custom components
- Uses Avalon® memory-mapped (Avalon-MM) interfaces

The Qsys flow offers the following additional advantages over SOPC Builder:

- Provides visualization of hierarchical designs
- Allows greater performance through interconnect elements and pipelining
- Provides closer integration with the Quartus II software

MegaWizard Plug-In Manager Flow

The MegaWizard Plug-In Manager flow allows you to customize the memory controller with ALTMEMPHY or UniPHY IP, and manually integrate the function into your design.

Specifying Parameters

To specify parameters using the MegaWizard Plug-In Manager flow, perform the following steps:

1. Create a Quartus II project using the **New Project Wizard** available from the File menu.
2. In the Quartus II software, launch the **MegaWizard Plug-in Manager** from the Tools menu, and follow the prompts in the MegaWizard Plug-In Manager interface to create or edit a custom IP core variation.
3. Select a memory controller with the memory IP in the **Installed Plug-Ins** list in the **External Memory** folder.
4. Specify the parameters on all pages in the **Parameter Settings** tab.

   For detailed explanation of the parameters, refer to "Parameterizing Memory Controllers with ALTMEMPHY IP" on page 9–46 and "Parameterizing Memory Controllers with UniPHY IP" on page 9–65.

   The UniPHY IP cores provide preset parameters for specific applications. You may need to modify some of the preset parameters depending on the frequency of operation. A typical list of parameters which need to be change are—Memory Parameters: Memory CAS latency setting and Memory CAS write latency setting. Memory Timing: tWTR, tFAW, tRRD, and tRTP.

5. Modify preset parameters, if required.
   a. Click the arrow to expand the **Presets** list.
   b. Select the desired preset, and then click **Apply**.
6. To create, modify, or remove your own custom presets, click **New**, **Update**, or **Delete** at the bottom of the **Presets** list.

7. If the IP core provides a simulation model, specify appropriate options in the wizard to generate a simulation model.

   Altera IP supports a variety of simulation models, including simulation-specific IP functional simulation models and encrypted RTL models, and plain text RTL models. These are all cycle-accurate models. The models allow for fast functional simulation of your IP core instance using industry-standard VHDL or Verilog HDL simulators. For some cores, only the plain text RTL model is generated, and you can simulate that model.

   For more information about functional simulation models for Altera IP cores, refer to *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

   CAUTION: Use the simulation models only for simulation and not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

8. This step applies to memory controllers with ALTMEMPHY IP. If the parameter editor includes **EDA** and **Summary** tabs, follow these steps:

   a. Some third-party synthesis tools can use a netlist that contains the structure of an IP core but no detailed logic to optimize timing and performance of the design containing it. To use this feature if your synthesis tool and IP core support it, turn on **Generate netlist**.

      When targeting a VHDL simulation model, the MegaWizard Plug-In Manager still generates the <*variation_name*>**_alt_mem_phy.v** for the Quartus II synthesis. Do not use this file for simulation. Use the <*variation_name*>**.vho** for simulation instead.

      The ALTMEMPHY megafunction only supports functional simulation. You cannot perform timing or gate-level simulation when using the ALTMEMPHY megafunction.

   b. On the **Summary** tab, if available, select the files you want to generate. A gray checkmark indicates a file that is automatically generated. All other files are optional.

      If file selection is supported for your IP core, after you generate the core, a generation report (<*variation name*>**.html)** appears in your project directory. This file contains information about the generated files.

9. Click the **Finish** button, the parameter editor generates the top-level HDL code for your IP core, and a simulation directory which includes files for simulation.

   The **Finish** button may be unavailable until all parameterization errors listed in the messages window are corrected.

10. Click **Yes** if you are prompted to add the **.qip** to the current Quartus II project. You can also turn on **Automatically add Quartus II IP Files to all projects**.

11. This step applies to memory controllers with ALTMEMPHY IP. Set the <*variation name*>**_example_top.v** or **.vhd** to be the project top-level design file.
    a. On the File menu, click **Open**.
    b. Browse to <*variation name*>**_example_top** and click **Open**.
    c. On the Project menu, click **Set as Top-Level Entity**.

You can now integrate your custom IP core instance in your design, simulate, and compile. While integrating your IP core instance into your design, you must make appropriate pin assignments. You can create a virtual pin to avoid making specific pin assignments for top-level signals while you are simulating and not ready to map the design to hardware.

For IP cores, the generation process also creates complete example designs. An example design for hardware testing is located in the **<***variation_name***>_example_design/example_project/** directory. An example design for RTL simulation is located in the **<***variation_name***>_example_design/simulation/** directory.

For information about the Quartus II software, including virtual pins and the MegaWizard Plug-In Manager, refer to Quartus II Help.

Constraining the Design

After you have generated the memory IP MegaCore function, you may need to set timing constraints and perform timing analysis using the Quartus II TimeQuest Timing Analyzer. When you generate the MegaCore function, the MegaWizard Plug-In Manager also generates a Synopsis Design Constraint File (**.sdc**), <*variation_name*>**.sdc**, and a pin assignment script, <*variation_name*>**_pin_assignments.tcl**. Both the **.sdc** and the <*variation name*>**_pin_assignments.tcl** scripts support multiple instances. These scripts iterate through all instances of the core and apply the same constraints to all of them. You can derive the timing constraints from the external device data sheet and tolerances from the board layout.

For more information about timing constraints and analysis, refer to the *Analyzing Timing of Memory IP* chapter.

Add Pins and DQ Group Assignments

The <*variation_name*>**_pin_assignments.tcl** script, sets up the I/O standards and the input/output termination for the memory IP. This script also helps to relate the `DQ` pin groups together for the Quartus II Fitter to place them correctly.

The pin assignment script does not create a PLL reference clock for the design. You must create a clock for the design and provide pin assignments for the signals of both the example driver and testbench that the MegaCore variation generates.

Run the <*variation_name*>**_pin_assignments.tcl** script to add the input and output termination, I/O standards, and DQ group assignments to the example design. To run the pin assignment script, follow these steps:

1. On the Processing menu, point to **Start**, and click **Start Analysis and Synthesis**.
2. On the Tools menu click **Tcl Scripts**.

3. Specify the **pin_assignments.tcl** and click **Run**.

If the PLL input reference clock pin does not have the same I/O standard as the memory interface I/Os, a no-fit might occur because incompatible I/O standards cannot be placed in the same I/O bank.

If you are upgrading your memory IP from an earlier Quartus II version, follow these steps:

- For UniPHY IP, rerun the **pin_assignments.tcl** script in the later Quartus II revision.
- For ALTMEMPHY IP, delete all the memory non-location I/O assignments and rerun the **pin_assignments.**tcl script.

Compiling the Design

After constraining your design, compile your design in the Quartus II software to generate timing reports to verify whether timing has been met.

To compile the design, on the Processing menu, click **Start Compilation**.

After you have compiled the top-level file, you can perform RTL simulation or program your targeted Altera device to verify the top-level file in hardware.

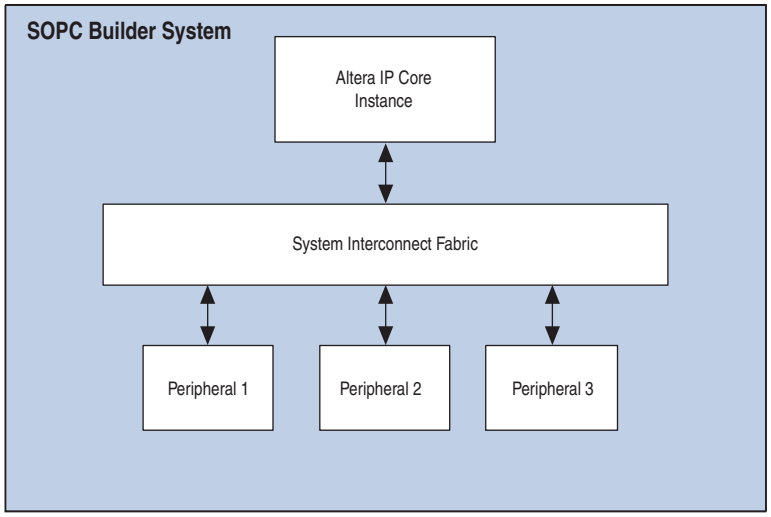
For more information about simulating the memory IP, refer to the *Simulating Memory IP* chapter.

SOPC Builder Flow

You can use SOPC Builder to build a system that includes your customized IP core. You can easily add other components and quickly create an SOPC Builder system. SOPC Builder automatically generates HDL files that include all of the specified components and interconnections. SOPC Builder defines default connections, which you can modify. The HDL files are ready to be compiled by the Quartus II software to produce output files for programming an Altera device.

Figure 9–3 shows a block diagram of an example SOPC Builder system.

Figure 9–3. SOPC Builder System



For more information about system interconnect fabric, refer to the *System Interconnect Fabric for Memory-Mapped Interfaces* and *System Interconnect Fabric for Streaming Interfaces* chapters in the *SOPC Builder User Guide* and to the *Avalon Interface Specifications*.

For more information about SOPC Builder and the Quartus II software, refer to the *SOPC Builder Features* and *Building Systems with SOPC Builder* sections in the *SOPC Builder User Guide* and to Quartus II Help.

Specifying Parameters

Specify the required parameters in the parameter editor, For detailed explanations of these parameters, refer to To specify IP core parameters in the SOPC Builder flow, follow these steps:

1. Create a new Quartus II project using the **New Project Wizard** available from the File menu.
2. On the Tools menu, click **SOPC Builder**.
3. For a new system, specify the system name and language.
4. On the **System Contents** tab, double-click the name of your IP core to add it to your system. The relevant parameter editor appears.

5. Specify the required parameters in the parameter editor. For detailed explanations of these parameters, refer to "Parameterizing Memory Controllers with ALTMEMPHY IP" on page 9–46 and "Parameterizing Memory Controllers with UniPHY IP" on page 9–65.

   ☞ The UniPHY IP cores provide preset parameters for specific applications. You may need to modify some of the preset parameters depending on the frequency of operation. A typical list of parameters which need to be change are—Memory Parameters: Memory CAS latency setting and Memory CAS write latency setting. Memory Timing: tWTR, tFAW, tRRD, and tRTP.

6. To use preset parameters, click the arrow to expand the **Presets** list, select the desired preset, and then click **Apply**.

   ☞ You can also add and modify your own presets.

7. To create, modify, or remove your own custom preset, click **New**, **Update**, or **Delete** at the bottom of the **Presets** list.

   ☞ You must also turn on **Generate SOPC Builder compatible resets** on the **Controller Settings** tab when parameterizing those cores.

8. Click **Finish** to complete the IP core instance and add it to the system.

The **Finish** button may be unavailable until all parameterization errors listed in the messages window are corrected.

Completing the SOPC Builder System

To complete the SOPC Builder system, follow these steps:

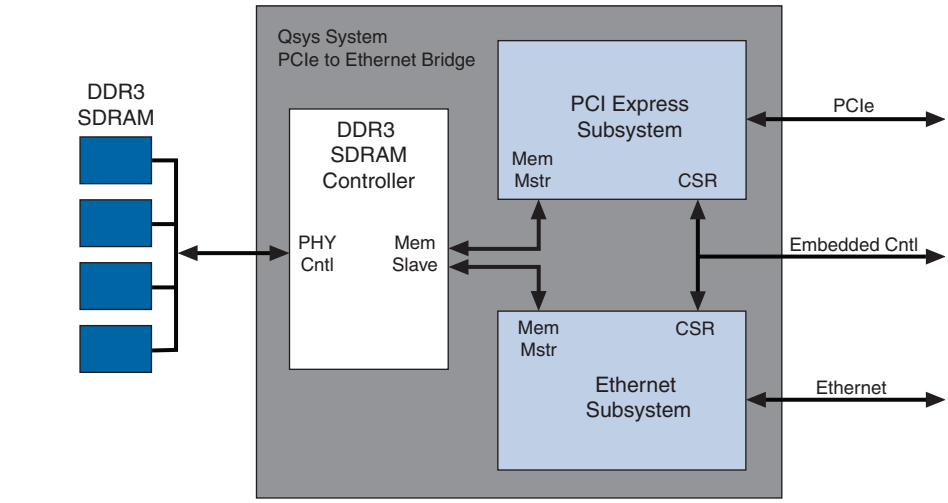
1. Add and parameterize any additional components. Some IP cores include a complete SOPC Builder system design example.
2. Use the Connection panel on the **System Contents** tab to connect the components.
3. By default, clock names are not displayed. To display clock names in the **Module Name** column and the clocks in the **Clock** column in the **System Contents** tab, click **Filters** to display the **Filters** dialog box. In the **Filter** list, click **All.**
4. Click **Generate** to generate the system. SOPC Builder generates the system and produces the <*system name*>**.qip** that contains the assignments and information required to process the IP core or system in the Quartus II Compiler.
5. In the Quartus II software, click **Add/Remove Files in Project** and add the **.qip** to the project.
6. Compile your design in the Quartus II software.

Qsys System Integration Tool Design Flow

You can use the Qsys system integration tool to build a system that includes your customized IP core. You easily can add other components and quickly create a Qsys system. Qsys automatically generates HDL files that include all of the specified components and interconnections. In Qsys, you specify the connections you want. The HDL files are ready to be compiled by the Quartus II software to produce output files for programming an Altera device. Qsys generates Verilog HDL simulation models for the IP cores that comprise your system.

Figure 9–4 shows a high level block diagram of an example Qsys system.

Figure 9–4. Example Qsys System



For more information about the Qsys system interconnect, refer to the *Qsys Interconnect* chapter in volume 1 of the *Quartus II Handbook* and to the *Avalon Interface Specifications*.

For more information about the Qsys tool and the Quartus II software, refer to the *System Design with Qsys* section in volume 1 of the *Quartus II Handbook* and to Quartus II Help.

Specify Parameters

To specify parameters for your IP core using the Qsys flow, follow these steps:

1. Create a new Quartus II project using the **New Project Wizard** available from the File menu.
2. On the Tools menu, click **Qsys**.
3. In the **Component Library** window, double-click the name of your IP core to add it to your system. The relevant parameter editor appears.

Specify the required parameters in all tabs in the Qsys tool. For detailed explanations of these parameters, refer to "Parameterizing Memory Controllers with ALTMEMPHY IP" on page 9–46 and "Parameterizing Memory Controllers with UniPHY IP" on page 9–65.

If your design includes external memory interface IP cores, you must turn on **Generate power-of-2 bus widths for SOPC Builder** on the **Controller Settings** tab when parameterizing those cores.

The UniPHY IP cores provide preset parameters for specific applications. You may need to modify some of the preset parameters depending on the frequency of operation. A typical list of parameters which need to be change are—Memory Parameters: Memory CAS latency setting and Memory CAS write latency setting. Memory Timing: tWTR, tFAW, tRRD, and tRTP.

4. To use preset parameters, click the arrow to expand the **Presets** list, select the desired preset, and then click **Apply**.

   You can also add or modify your own presets.

5. To create, modify, or remove your own custom presets, click **New**, **Update**, or **Delete** at the bottom of the **Presets** list.
6. Click **Finish** to complete the IP core instance and add it to the system.

   The **Finish** button may be unavailable until all parameterization errors listed in the messages window are corrected.

Complete the Qsys System

To complete the Qsys system, follow these steps:

1. Add and parameterize any additional components.
2. Connect the components using the Connection panel on the **System Contents** tab.
3. In the **Export** column, enter the name of any connections that should be a top-level Qsys system port.
4. If you intend to simulate your Qsys system, on the **Generation** tab, select either **Create testbench Qsys system** to **Standard, BFMs for standard Avalon interfaces** to create a testbench with bus functional models (BFMs) attached to all exported interfaces or **Simple, BFMs for clocks and resets** to create a testbench with BFMs driving only clocks and reset interfaces.
5. To generate a simulation model for the testbench Qsys system at the same time, set **Create testbench simulation model** to **Verilog** or **VHDL**. Set this option to **None** to view or modify the generated testbench system before generating its simulation model.
6. If your system is not part of a Quartus II project and you want to generate synthesis register transfer language (RTL) or high-level hardware description language (HDL) files, turn on **Create HDL design files for synthesis**.

7. Click **Generate** to generate the system. Qsys generates the system and produces the <*system name*>**.qip** that contains the assignments and information required to process the IP core or system in the Quartus II Compiler.
8. In the Quartus II software, click **Add/Remove Files in Project** and add the **.qip** to the project.
9. Compile your project in the Quartus II software.

To ensure that the **memory** and **oct** interfaces are exported to the top-level RTL file, be careful not to accidentally rename or delete either of these interfaces in the **Export** column of the **System Contents** tab.

Qsys and SOPC Builder Interfaces

Table 9–1 and Table 9–2 list the DDR2 and DDR3 SDRAM with UniPHY signals available for each interface in Qsys and SOPC Builder and provide a description and guidance on how to connect those interfaces.

Table 9–1. DDR2 SDRAM Controller with UniPHY Interfaces (Part 1 of 6)

Signals in Interface	Interface Type	Description/How to Connect
pll_ref_clk interface		
pll_ref_clk	Clock input	PLL reference clock input.
global_reset interface		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
soft_reset interface		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
afi_reset interface		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. The controller asserts this interface when the PLL loses lock or the PHY is reset.
afi_reset_in interface		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the `afi_reset_n` output of an identically configured memory interface in PLL master mode.
afi_clk interface		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.

Table 9–1. DDR2 SDRAM Controller with UniPHY Interfaces (Part 2 of 6)

Signals in Interface	Interface Type	Description/How to Connect
afi_clk_in interface		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, you must connect this `afi_clk` input to the `afi_clk` output of an identically configured memory interface in PLL master mode.
afi_half_clk interface		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of `afi_clk`. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_half_clk_in interface		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of `afi_clk`. When the interface is in PLL slave mode, this is a clock input that you must connect to the `afi_half_clk` output of an identically configured memory interface in PLL master mode.
memory interface (DDR2 SDRAM)		
mem_a	Conduit	Interface signals between the PHY and the memory device.
mem_ba		
mem_ck		
mem_ck_n		
mem_cke		
mem_cs_n		
mem_dm		
mem_ras_n		
mem_cas_n		
mem_we_n		
mem_dq		
mem_dqs		
mem_dqs_n		
mem_odt		
mem_ac_parity		
mem_err_out_n		
mem_parity_error_n		
memory interface (LPDDR2)		

Table 9–1. DDR2 SDRAM Controller with UniPHY Interfaces (Part 3 of 6)

Signals in Interface	Interface Type	Description/How to Connect
mem_ca	Conduit	Interface signals between the PHY and the memory device.
mem_ck		
mem_ck_n		
mem_cke		
mem_cs_n		
mem_dm		
mem_dq		
mem_dqs		
mem_dqs_n		
avl interface		
avl_ready	Avalon-MM Slave	Avalon-MM interface signals between the memory interface and user logic.
avl_burst_begin		
avl_addr		
avl_rdata_valid		
avl_rdata		
avl_wdata		
avl_be		
avl_read_req		
avl_write_req		
avl_size		
status interface		
local_init_done	Conduit	Memory interface status signals.
local_cal_success		
local_cal_fail		
oct interface		
rup (Stratix® III/IV, Arria® II GZ)	Conduit	OCT reference resistor pins for **rup**/`rdn` or `rzqin`.
rdn (Stratix III/IV, Arria II GZ)		
rzq (Stratix V, Arria V, Cyclone V)		
local_powerdown interface		
local_powerdn_ack	Conduit	This powerdown interface for the controller is enabled only when you turn on **Enable Auto Powerdown**.
pll_sharing interface		

Table 9–1. DDR2 SDRAM Controller with UniPHY Interfaces (Part 4 of 6)

Signals in Interface	Interface Type	Description/How to Connect
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set **PLL sharing mode** to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_hr_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
pll_dr_clk		
dll_sharing interface		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set **DLL sharing mode** to master or slave.
dll_pll_locked		
oct_sharing interface		
seriesterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set **OCT sharing mode** to master or slave.
parallelterminationcontrol		
hcx_dll_reconfig interface		
dll_offset_ctrl_addnsub	Conduit	This DLL reconfiguration interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to enable DLL reconfiguration.
dll_offset_ctrl_offset		
dll_offset_ctrl_addnsub *(1)*		
dll_offset_ctrl_offset *(1)*		
dll_offset_ctrl_offsetctrlout *(1)*		
dll_offset_ctrl_b_offsetctrlout *(1)*		
hcx_pll_reconfig interface		
configupdate	Conduit	This PLL reconfiguration interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to enable PLL reconfiguration.
phasecounterselect		
phasestep		
phaseupdown		
scanclk		
scanclkena		
scandata		
phasedone		
scandataout		
scandone		
hcx_rom_reconfig interface		

Table 9–1. DDR2 SDRAM Controller with UniPHY Interfaces (Part 5 of 6)

Signals in Interface	Interface Type	Description/How to Connect
hc_rom_config_clock	Conduit	This ROM loader interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to control the loading of the sequencer ROM.
hc_rom_conig_datain		
hc_rom_config_rom_data_ready		
hc_rom_config_init		
hc_rom_config_init_busy		
hc_rom_config_rom_rden		
hc_rom_config_rom_address		
autoprecharge_req interface		
local_autopch_req	Conduit	Precharge interface for connection to a custom control block. This interface is enabled only when you turn on **Auto-precharge Control**.
user_refresh interface		
local_refresh_req	Conduit	User refresh interface for connection to a custom control block. This interface is enabled only when you turn on **User Auto-Refresh Control**.
local_refresh_chip		
local_refresh_ack		
self_refresh interface		
local_self_rfsh_req	Conduit	Self refresh interface for connection to a custom control block. This interface is enabled only when you turn on **Self-refresh Control**.
local_self_rfsh_chip		
local_self_rfsh_ack		
ecc_interrupt interface		
ecc_interrupt	Conduit	ECC interrupt signal for connection to a custom control block. This interface is enabled only when you turn on **Error Detection and Correction Logic**.
csr interface		
csr_write_req	Avalon-MM Slave	Configuration and status register signals for the memory interface, for connection to an Avalon_MM master. This interface is enabled only when you turn on **Configuration and Status Register**.
csr_read_req		
csr_waitrequest		
csr_addr		
csr_be		
csr_wdata		
csr_rdata		
csr_rdata_valid		
Hard Memory Controller MPFE FIFO Clock Interface		

Table 9–1. DDR2 SDRAM Controller with UniPHY Interfaces (Part 6 of 6)

Signals in Interface	Interface Type	Description/How to Connect
mp_cmd_clk	Conduit	When you enable the Hard Memory Interface, three FIFO buffers (command, read data, and write data) are created in the MPFE. Each FIFO buffer has its own clock and reset port. This interface is enabled when you turn on the Enable Hard Memory Interface.
mp_rfifo_clk		
mp_wfifo_clk		
mp_cmd_reset		
mp_rfifo_reset		
mp_wfifo_reset		
Hard Memory Controller Bonding Interface		
bonding_in_1	Conduit	Bonding interface to bond two controllers to expand the bandwidth. This interface is enabled when you turn on the Export bonding interface.
bonding_in_2		
bonding_in_3		
bonding_out_1		
bonding_out_2		
bonding_out_3		
Note to Table 9–1: (1) Signals available only in DLL master mode.		

Table 9–2. DDR3 SDRAM Controller with UniPHY Interfaces (Part 1 of 6)

Signals in Interface	Interface Type	Description/How to Connect
pll_ref_clk interface		
pll_ref_clk	Clock input	PLL reference clock input.
global_reset interface		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
soft_reset interface		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
afi_reset interface		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. This interface is asserted when the PLL loses lock or the PHY is reset.
afi_reset_in interface		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the `afi_reset_n` output of an identically configured memory interface in PLL master mode.

Table 9–2. DDR3 SDRAM Controller with UniPHY Interfaces (Part 2 of 6)

Signals in Interface	Interface Type	Description/How to Connect
afi_clk interface		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_clk_in interface		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, this is a clock input that you must connect to the `afi_clk` output of an identically configured memory interface in PLL master mode.
afi_half_clk interface		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of `afi_clk`. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_half_clk_in interface		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of the `afi_clk`. When the interface is in PLL slave mode, you must connect this `afi_half_clk` input to the `afi_half_clk` output of an identically configured memory interface in PLL master mode.

Table 9–2. DDR3 SDRAM Controller with UniPHY Interfaces (Part 3 of 6)

Signals in Interface	Interface Type	Description/How to Connect
memory interface		
mem_a	Conduit	Interface signals between the PHY and the memory device.
mem_ba		
mem_ck		
mem_ck_n		
mem_cke		
mem_cs_n		
mem_dm		
mem_ras_n		
mem_cas_n		
mem_we_n		
mem_dq		
mem_dqs		
mem_dqs_n		
mem_odt		
mem_reset_n		
mem_ac_parity		
mem_err_out_n		
mem_parity_error_n		
avl interface		
avl_ready	Avalon-MM Slave	Avalon-MM interface signals between the memory interface and user logic.
avl_burst_begin		
avl_addr		
avl_rdata_valid		
avl_rdata		
avl_wdata		
avl_be		
avl_read_req		
avl_write_req		
avl_size		
status interface		
local_init_done	Conduit	Memory interface status signals.
local_cal_success		
local_cal_fail		
oct interface		
rup (Stratix III/IV, Arria II GZ)	Conduit	OCT reference resistor pins for `rup`/`rdn` or `rzqin`.
rdn (Stratix III/IV, Arria II GZ)		
rzq (Stratix V, Arria v, Cyclone V)		

Table 9–2. DDR3 SDRAM Controller with UniPHY Interfaces (Part 4 of 6)

Signals in Interface	Interface Type	Description/How to Connect
local_powerdown interface		
local_powerdn_ack	Conduit	This powerdown interface for the controller is enabled only when you turn on **Enable Auto Power Down**.
pll_sharing interface		
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set **PLL sharing mode** to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_hr_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
pll_dr_clk		
dll_sharing interface		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set **DLL sharing mode** to master or slave.
dll_pll_locked		
oct_sharing interface		
seriesterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set **OCT sharing mode** to master or slave.
parallelterminationcontrol		
hcx_dll_reconfig interface		
dll_offset_ctrl_addnsub	Conduit	This DLL reconfiguration interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to enable DLL reconfiguration.
dll_offset_ctrl_offset		
dll_offset_ctrl_addnsub *(1)*		
dll_offset_ctrl_offset *(1)*		
dll_offset_ctrl_offsetctrlout *(1)*		
dll_offset_ctrl_b_offsetctrlout *(1)*		

Table 9–2. DDR3 SDRAM Controller with UniPHY Interfaces (Part 5 of 6)

Signals in Interface	Interface Type	Description/How to Connect
hcx_pll_reconfig interface		
configupdate	Conduit	This PLL reconfiguration interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to enable PLL reconfiguration.
phasecounterselect		
phasestep		
phaseupdown		
scanclk		
scanclkena		
scandata		
phasedone		
scandataout		
scandone		
hcx_rom_reconfig interface		
hc_rom_config_clock	Conduit	This ROM loader interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to control loading of the sequencer ROM.
hc_rom_conig_datain		
hc_rom_config_rom_data_ready		
hc_rom_config_init		
hc_rom_config_init_busy		
hc_rom_config_rom_rden		
hc_rom_config_rom_address		
autoprecharge_req interface		
local_autopch_req	Conduit	Precharge interface for connection to a custom control block. This interface is enabled only when you turn on **Auto-precharge Control.**
user_refresh interface		
local_refresh_req	Conduit	User refresh interface for connection to a custom control block. This interface is enabled only when you turn on **User Auto-Refresh Control**.
local_refresh_chip		
local_refresh_ack		
self_refresh interface		
local_self_rfsh_req	Conduit	Self refresh interface for connection to a custom control block. This interface is enabled only when you turn on **Self-refresh Control**.
local_self_rfsh_chip		
local_self_rfsh_ack		
ecc_interrupt interface		
ecc_interrupt	Conduit	ECC interrupt signal for connection to a custom control block. This interface is enabled only when you turn on **Error Detection and Correction Logic**.

Table 9–2. DDR3 SDRAM Controller with UniPHY Interfaces (Part 6 of 6)

Signals in Interface	Interface Type	Description/How to Connect
csr interface		
csr_write_req	Avalon-MM Slave	Configuration and status register signals for the memory interface, for connection to an Avalon_MM master. This interface is enabled only when you turn on **Configuration and Status Register**.
csr_read_req		
csr_waitrequest		
csr_addr		
csr_be		
csr_wdata		
csr_rdata		
csr_rdata_valid		
Hard Memory Controller MPFE FIFO Clock Interface		
mp_cmd_clk	Conduit	When you enable the Hard Memory Interface, three FIFO buffers (command, read data, and write data) are created in the MPFE. Each FIFO buffer has its own clock and reset port. This interface is enabled when you turn on the Enable Hard Memory Interface.
mp_rfifo_clk		
mp_wfifo_clk		
mp_cmd_reset_n		
mp_rfifo_reset_n		
mp_wfifo_reset_n		
Hard Memory Controller Bonding Interface		
bonding_in_1	Conduit	Use bonding interface to bond two controllers to expand the bandwidth. This interface is enabled when you turn on the Export bonding interface.
bonding_in_2		
bonding_in_3		
bonding_out_1		
bonding_out_2		
bonding_out_3		
Note to Table 9–2 (1) Signals available only in DLL master mode.		

Table 9–3 lists the LPDDR2 SDRAM signals available for each interface in Qsys and SOPC Builder and provides a description and guidance on how to connect those interfaces.

Table 9–3. LPDDR2 SDRAM Controller with UniPHY Interfaces (Part 1 of 5)

Signals in Interface	Interface Type	Description/How to Connect
pll_ref_clk interface		
pll_ref_clk	Clock input	PLL reference clock input.
global_reset interface		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
soft_reset interface		

Table 9–3. LPDDR2 SDRAM Controller with UniPHY Interfaces (Part 2 of 5)

Signals in Interface	Interface Type	Description/How to Connect
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
afi_reset interface		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. The controller asserts this interface when the PLL loses lock or the PHY is reset.
afi_reset_in interface		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the afi_reset output of an identically configured memory interface in PLL master mode.
afi_clk interface		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_clk_in interface		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, you must connect this afi_clk input to the afi_clk output of an identically configured memory interface in PLL master mode.
afi_half_clk interface		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_half_clk_in interface		

Table 9–3. LPDDR2 SDRAM Controller with UniPHY Interfaces (Part 3 of 5)

Signals in Interface	Interface Type	Description/How to Connect
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL slave mode, this is a clock input that you must connect to the afi_half_clk output of an identically configured memory interface in PLL master mode.
memory interface		
mem_ca	Conduit	Interface signals between the PHY and the memory device.
mem_ck		
mem_ck_n		
mem_cke		
mem_cs_n		
mem_dm		
mem_dq		
mem_dqs		
mem_dqs_n		
avl interface		
avl_ready	Avalon-MM Slave	Avalon-MM interface signals between the memory interface and user logic.
avl_burst_begin		
avl_addr		
avl_rdata_valid		
avl_rdata		
avl_wdata		
avl_be		
avl_read_req		
avl_write_req		
avl_size		
status interface		
local_init_done	Conduit	Memory interface status signals.
local_cal_success		
local_cal_fail		
oct interface		
rzq	Conduit	OCT reference resistor pins for rzqin.
local_powerdown interface		
local_powerdn_ack	Conduit	This powerdown interface for the controller is enabled only when you turn on Enable Auto Powerdown.

Table 9–3. LPDDR2 SDRAM Controller with UniPHY Interfaces (Part 4 of 5)

Signals in Interface	Interface Type	Description/How to Connect
local_deep_powerdn interface		
local_deep_powerdn_ack	Conduit	Deep power down interface for the controller to enable deep power down. This interface is enable when turn on Enable Deep Power-Down Controls.
local_deep_powerdn_chip		
local_deep_powerdn_req		
pll_sharing interface		
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set PLL sharing mode to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_mem_phy_clk		
afi_phy_clk		
pll_write_clk_pre_phy_clk		
dll_sharing interface		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set DLL sharing mode to master or slave.
dll_pll_locked		
oct_sharing interface		
seriesterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set OCT sharing mode to master or slave.
parallelterminationcontrol		
autoprecharge_req interface		
local_autopch_req	Conduit	Precharge interface for connection to a custom control block. This interface is enabled only when you turn on Auto-precharge Control.
user_refresh interface		
local_refresh_req	Conduit	User refresh interface for connection to a custom control block. This interface is enabled only when you turn on User Auto-Refresh Control.
local_refresh_chip		
local_refresh_ack		
self_refresh interface		
local_self_rfsh_req	Conduit	Self refresh interface for connection to a custom control block. This interface is enabled only when you turn on Self-refresh Control.
local_self_rfsh_chip		
local_self_rfsh_ack		

Table 9–3. LPDDR2 SDRAM Controller with UniPHY Interfaces (Part 5 of 5)

Signals in Interface	Interface Type	Description/How to Connect
ecc_interrupt interface		
ecc_interrupt	Conduit	ECC interrupt signal for connection to a custom control block. This interface is enabled only when you turn on Error Detection and Correction Logic.
csr interface		
csr_write_req	Avalon-MM Slave	Configuration and status register signals for the memory interface, for connection to an Avalon_MM master. This interface is enabled only when you turn on Configuration and Status Register.
csr_read_req		
csr_waitrequest		
csr_addr		
csr_be		
csr_wdata		
csr_rdata		
csr_rdata_valid		
Local_rdata_error interface		
Local_rdata_error	Conduit	Indicates read data error when Error Detection and Correction logic is enabled.
Hard Memory Controller MPFE FIFO Clock Interface		
mp_cmd_clk	Conduit	When you enable the Hard Memory Interface, three FIFO buffers (command, read data, and write data) are created in the MPFE. Each FIFO buffer has its own clock and reset port. This interface is enabled when you turn on the Enable Hard Memory Interface.
mp_rfifo_clk		
mp_wfifo_clk		
mp_cmd_reset_n		
mp_rfifo_reset_n		
mp_wfifo_reset_n		
Hard Memory Controller Bonding Interface		
bonding_in_1	Conduit	Bonding interface to bond two controllers to expand the bandwidth. This interface is enabled when you turn on the Export bonding interface.
bonding_in_2		
bonding_in_3		
bonding_out_1		
bonding_out_2		
bonding_out_3		

Table 9–4 lists the QDR II and QDR II+ SRAM signals available for each interface in Qsys and SOPC Builder and provides a description and guidance on how to connect those interfaces.

Table 9–4. QDR II and QDR II+ SRAM Controller with UniPHY Interfaces (Part 1 of 4)

Signals in Interface	Interface Type	Description/How to Connect
pll_ref_clk interface		
pll_ref_clk	Clock input	PLL reference clock input.
global_reset interface		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
soft_reset interface		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
afi_reset interface		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. This interface is asserted when the PLL loses lock or the PHY is reset.
afi_reset_in interface		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the `afi_reset` output of an identically configured memory interface in PLL master mode.
afi_clk interface		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.

Table 9–4. QDR II and QDR II+ SRAM Controller with UniPHY Interfaces (Part 2 of 4)

Signals in Interface	Interface Type	Description/How to Connect
afi_clk_in interface		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, this is a clock input that you must connect to the `afi_clk` output of an identically configured memory interface in PLL master mode.
afi_half_clk interface		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of `afi_clk`. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_half_clk_in interface		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of `afi_clk`. When the interface is in PLL slave mode, you must connect this `afi_half_clk` input to the `afi_half_clk` output of an identically configured memory interface in PLL master mode.
memory interface		
mem_a	Conduit	Interface signals between the PHY and the memory device.
mem_cqn		
mem_bws_n		
mem_cq		
mem_d		
mem_k		
mem_k_n		
mem_q		
mem_wps_n		
mem_rps_n		
mem_doff_n		

Table 9–4. QDR II and QDR II+ SRAM Controller with UniPHY Interfaces (Part 3 of 4)

Signals in Interface	Interface Type	Description/How to Connect
avl_r interface		
avl_r_read_req	Avalon-MM Slave	Avalon-MM interface between memory interface and user logic for read requests.
avl_r_ready		
avl_r_addr		
avl_r_size		
avl_r_rdata_valid		
avl_r_rdata		
avl_w interface		
avl_w_write_req	Avalon-MM Slave	Avalon-MM interface between memory interface and user logic for write requests.
avl_w_ready		
avl_w_addr		
avl_w_size		
avl_w_wdata		
avl_w_be		
status interface		
local_init_done	Conduit	Memory interface status signals.
local_cal_success		
local_cal_fail		
oct interface		
rup (Stratix III/IV, Arria II GZ, Arria II GX)	Conduit	OCT reference resistor pins for `rup`/`rdn` or `rzqin`.
rdn (Stratix III/IV, Arria II GZ, Arria II GX)		
rzq (Stratix V, Arria V, Cyclone V)		
pll_sharing interface		
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set **PLL sharing mode** to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_hr_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
pll_dr_clk		
dll_sharing interface		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set **DLL sharing mode** to master or slave.
dll_pll_locked		

Table 9–4. QDR II and QDR II+ SRAM Controller with UniPHY Interfaces (Part 4 of 4)

Signals in Interface	Interface Type	Description/How to Connect
oct_sharing interface		
seriesterminationcontrol (Stratix III/IV/V, Arria II GZ, Arria V, Cyclone V)	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set **OCT sharing mode** to master or slave.
parallelterminationcontrol (Stratix III/IV/V, Arria II GZ, Arria V, Cyclone V)		
terminationcontrol (Arria II GX)		
hcx_dll_reconfig		
dll_offset_ctrl_addnsub	Conduit	This DLL reconfiguration interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to enable DLL reconfiguration.
dll_offset_ctrl_offset		
dll_offset_ctrl_addnsub (1)		
dll_offset_ctrl_offset (1)		
dll_offset_ctrl_offsetctrlout (1)		
dll_offset_ctrl_b_offsetctrlout (1)		
hcx_pll_reconfig		
configupdate	Conduit	This PLL reconfiguration interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to enable PLL reconfiguration.
phasecounterselect		
phasestep		
phaseupdown		
scanclk		
scanclkena		
scandata		
phasedone		
scandataout		
scandone		
hcx_rom_reconfig		
hc_rom_config_clock	Conduit	This ROM loader interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to control loading of the sequencer ROM.
hc_rom_config_datain		
hc_rom_config_rom_data_ready		
hc_rom_config_init		
hc_rom_config_init_busy		
hc_rom_config_rom_rden		
hc_rom_config_rom_address		

Note to Table 9–4

(1) Signals available only in DLL master mode.

Table 9–5 lists the RLDRAM II signals available for each interface in Qsys and SOPC Builder and provides a description and guidance on how to connect those interfaces.

Table 9–5. RLDRAM II Controller with UniPHY Interfaces (Part 1 of 4)

Interface Name	Interface Type	Description
pll_ref_clk interface		
pll_ref_clk	Clock input.	PLL reference clock input.
global_reset interface		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
soft_reset interface		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
afi_reset interface		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. This interface is asserted when the PLL loses lock or the PHY is reset.
afi_reset_in interface		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the `afi_reset` output of an identically configured memory interface in PLL master mode.
afi_clk interface		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_clk_in interface		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, you must connect this `afi_clk` input to the `afi_clk` output of an identically configured memory interface in PLL master mode.
afi_half_clk interface		

Table 9–5. RLDRAM II Controller with UniPHY Interfaces (Part 2 of 4)

Interface Name	Interface Type	Description
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of the `afi_clk`. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_half_clk_in interface		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of the `afi_clk`. When the interface is in PLL slave mode, you must connect this `afi_half_clk` input to the `afi_half_clk` output of an identically configured memory interface in PLL master mode.
memory interface		
mem_a	Conduit	Interface signals between the PHY and the memory device.
mem_ba		
mem_ck		
mem_ck_n		
mem_cs_n		
mem_dk		
mem_dk_n		
mem_dm		
mem_dq		
mem_qk		
mem_qk_n		
mem_ref_n		
mem_we_n		
avl interface		
avl_size	Avalom-MM Slave	Avalon-MM interface between memory interface and user logic.
avl_wdata		
avl_rdata_valid		
avl_rdata		
avl_ready		
avl_write_req		
avl_read_req		
avl_addr		
status interface		
local_init_done	Conduit	Memory interface status signals.
local_cal_success		
local_cal_fail		

Table 9–5. RLDRAM II Controller with UniPHY Interfaces (Part 3 of 4)

Interface Name	Interface Type	Description
oct interface		
rup (Stratix III/IV, Arria II GZ)	Conduit	OCT reference resistor pins for `rup`/`rdn` or `rzqin`.
rdn (Stratix III/IV, Arria II GZ)		
rzq (Stratix V)		
pll_sharing interface		
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set **PLL sharing mode** to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_hr_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
pll_dr_clk		
dll_sharing interface		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set **DLL sharing mode** to master or slave.
oct_sharing interface		
seriesterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set **OCT sharing mode** to master or slave.
parallelterminationcontrol		
hcx_dll_reconfig interface		
dll_offset_ctrl_addnsub	Conduit	This DLL reconfiguration interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to enable DLL reconfiguration.
dll_offset_ctrl_offset		
dll_offset_ctrl_addnsub *(1)*		
dll_offset_ctrl_offset *(1)*		
dll_offset_ctrl_offsetctrlout *(1)*		
dll_offset_ctrl_b_offsetctrlout *(1)*		

Table 9–5. RLDRAM II Controller with UniPHY Interfaces (Part 4 of 4)

Interface Name	Interface Type	Description
hcx_pll_reconfig interface		
configupdate	Conduit	This PLL reconfiguration interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to enable PLL reconfiguration.
phasecounterselect		
phasestep		
phaseupdown		
scanclk		
scanclkena		
scandata		
phasedone		
scandataout		
scandone		
hcx_rom_reconfig interface		
hc_rom_config_clock	Conduit	This ROM loader interface is enabled only when you turn on **HardCopy Compatibility Mode**. You can connect this interface to user-created custom logic to control loading of the sequencer ROM.
hc_rom_config_datain		
hc_rom_config_rom_data_ready		
hc_rom_config_init		
hc_rom_config_init_busy		
hc_rom_config_rom_rden		
hc_rom_config_rom_adress		
parity_error_interrupt interface		
parity_error	Conduit	Parity error interrupt conduit for connection to custom control block. This interface is enabled only if you turn on **Enable Error Detection Parity**.
user_refresh interface		
ref_req	Conduit	User refresh interface for connection to custom control block. This interface is enabled only if you turn on **Enable User Refresh**.
ref_ba		
ref_ack		
reserved interface		
reserved	Conduit	Reserved interface required for certain pin configurations when you select the Nios® II-based sequencer.
Note to Table 9–5 (1) Signals available only in DLL master mode.		

Table 9–6 lists the RLDRAM 3 signals available for each interface in Qsys and SOPC Builder and provides a description and guidance on how to connect those interfaces.

Table 9–6. RLDRAM 3 UniPHY Interface (Part 1 of 4)

Signals in Interface	Interface Type	Description/How to Connect
pll_ref_clk interface		

Table 9–6. RLDRAM 3 UniPHY Interface (Part 2 of 4)

pll_ref_clk	Clock input	PLL reference clock input.
global_reset interface		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
soft_reset interface		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
afi_reset interface		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. The controller asserts this interface when the PLL loses lock or the PHY is reset.
afi_reset_in interface		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the afi_reset output of an identically configured memory interface in PLL master mode.
afi_clk interface		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_clk_in interface		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, you must connect this afi_clk input to the afi_clk output of an identically configured memory interface in PLL master mode.
afi_half_clk interface		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL master or no sharing modes, this interface is a clock output.
afi_half_clk_in interface		

Table 9–6. RLDRAM 3 UniPHY Interface (Part 3 of 4)

afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL slave mode, this is a clock input that you must connect to the afi_half_clk output of an identically configured memory interface in PLL master mode.
memory interface		
mem_a	Conduit	Interface signals between the PHY and the memory device.
mem_ba		
mem_ck		
mem_ck_n		
mem_cs_n		
mem_dk		
mem_dk_n		
mem_dm		
mem_dq		
mem_qk		
mem_qk_n		
mem_ref_n		
mem_we_n		
mem_reset_n		
afi interface		
afi_addr	Avalon-MM Slave	Altera PHY interface (AFI) signals between the PHY and controller.
afi_ba		
afi_cs_n		
afi_we_n		
afi_ref_n		
afi_wdata_valid		
afi_wdata		
afi_dm		
afi_rdata		
afi_rdata_en		
afi_rdata_en_full		
afi_rdata_valid		
afi_rst_n		
afi_cal_success		
afi_cal_fail		
afi_wlat		
afi_rlat		
oct interface		

Table 9–6. RLDRAM 3 UniPHY Interface (Part 4 of 4)

oct_rzqin	Conduit	OCT reference resistor pins for rzqin.
pll_sharing interface		
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set PLL sharing mode to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_mem_phy_clk		
afi_phy_clk		
pll_write_clk_pre_phy_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
dll_sharing interface		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set DLL sharing mode to master or slave.
dll_pll_locked		
oct_sharing interface		
seriesterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set OCT sharing mode to master or slave.
parallelterminationcontrol		

Generated Files

When you complete the IP generation flow, there are generated files created in your project directory. The directory structure created varies somewhat, depending on the tool used to parameterize and generate the IP.

☞ The PLL parameters are statically defined in the <*variation_name*>**_parameters.tcl** at generation time. To ensure timing constraints and timing reports are correct, when you edit the PLL parameters, apply those changes to the PLL parameters in this file.

The following sections list the generated files for the ALTMEMPHY and UniPHY IP.

Generated Files for Memory Controllers with the ALTMEMPHY IP

Table 9–7 lists the ALTMEMPHY generated directory and key files using the MegaWizard Plug-In Manager.

Table 9–7. ALTMEMPHY Generated Files (Part 1 of 2)

File Name	Description
alt_mem_phy_defines.v	Contains constants used in the interface. This file is always in Verilog HDL regardless of the language you chose in the MegaWizard Plug-In Manager.
<*variation_name*>**.ppf**	Pin planner file for your ALTMEMPHY variation.
<*variation_name*>**.qip**	Quartus II IP file for your ALTMEMPHY variation, containing the files associated with this megafunction.
<*variation_name*>**.v/.vhd**	Top-level file of your ALTMEMPHY variation, generated based on the language you chose in the MegaWizard Plug-In Manager.
<*variation_name*>**.vho**	Contains functional simulation model for VHDL only.
<*variation_name*>**_alt_mem_phy_seq_wrapper.vo/.vho**	A wrapper file, for simulation only, that calls the sequencer file, created based on the language you chose in the MegaWizard Plug-In Manager.
<*variation_name*>**.html**	Lists the top-level files created and ports used in the megafunction.
<*variation_name*>**_alt_mem_phy_seq_wrapper.v/.vhd**	A wrapper file, for compilation only, that calls the sequencer file, created based on the language you chose in the MegaWizard Plug-In Manager.
<*variation_name*>**_alt_mem_phy_seq.vhd**	Contains the sequencer used during calibration. This file is always in VHDL language regardless of the language you chose in the MegaWizard Plug-In Manager.
<*variation_name*>**_alt_mem_phy.v**	Contains all modules of the ALTMEMPHY variation except for the sequencer. This file is always in Verilog HDL language regardless of the language you chose in the MegaWizard Plug-In Manager. The <*variation_name*>**_alt_mem_phy_seq.vhd** includes the DDR3 SDRAM sequencer.
<*variation name*>**_alt_mem_phy_pll**_<*device*>**.ppf**	This XML file describes the MegaCore pin attributes to the Quartus II Pin Planner.
<*variation_name*>**_alt_mem_phy_pll.v/.vhd**	The PLL megafunction file for your ALTMEMPHY variation, generated based on the language you chose in the MegaWizard Plug-In Manager.
<*variation_name*>**_alt_mem_phy_delay.vhd**	Includes a delay module for simulation. This file is only generated if you choose VHDL as the language of your MegaWizard Plug-In Manager output files.
<*variation_name*>**_alt_mem_phy_dq_dqs.vhd** or **.v**	Generated file that contains DQ/DQS I/O atoms interconnects and instance. Only generated when targeting Arria II GX devices.

Table 9–7. ALTMEMPHY Generated Files (Part 2 of 2)

File Name	Description
<*variation_name*>**_alt_mem_phy_dq_dqs_clearbox.txt**	Specification file that generates the <*variation_name*>**_alt_mem_phy_dq_dqs** file using the clearbox flow. Only generated when targeting Arria II GX devices.
<*variation_name*>**_alt_mem_phy_pll.qip**	Quartus II IP file for the PLL that your ALTMEMPHY variation uses that contains the files associated with this megafunction.
<*variation_name*>**_alt_mem_phy_pll_bb.v/.cmp**	Black box file for the PLL used in your ALTMEMPHY variation. Typically unused.
<*variation_name*>**_alt_mem_phy_reconfig . qip**	Quartus II IP file for the PLL reconfiguration block. Only generated when targeting Arria GX, HardCopy® II, Stratix II, and Stratix II GX devices.
<*variation_name*>**_alt_mem_phy_reconfig.v/.vhd**	PLL reconfiguration block module. Only generated when targeting Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.
<*variation_name*>**_alt_mem_phy_reconfig_bb.v/cmp**	Black box file for the PLL reconfiguration block. Only generated when targeting Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.
<*variation_name*>**_bb.v/.cmp**	Black box file for your ALTMEMPHY variation, depending whether you are using Verilog HDL or VHDL language.
<*variation_name*>**_ddr_pins.tcl**	Contains procedures used in the <*variation_name*>**_ddr_timing.sdc** and <*variation_name*>**_report_timing.tcl** files.
<*variation_name*>**_pin_assignments.tcl**	Contains I/O standard, drive strength, output enable grouping, DQ/DQS grouping, and termination assignments for your ALTMEMPHY variation. If your top-level design pin names do not match the default pin names or a prefixed version, edit the assignments in this file.
<*variation_name*>**_ddr_timing.sdc**	Contains timing constraints for your ALTMEMPHY variation.
<*variation_name*>**_report_timing.tcl**	Script that reports timing for your ALTMEMPHY variation during compilation.

Table 9–8 lists the modules that are instantiated in the **<***variation_name***>_alt_mem_phy.v/.vhd** file. A particular ALTMEMPHY variation may or may not use any of the modules, depending on the memory standard that you specify.

Table 9–8. Modules in <*variation_name*>_alt_mem_phy.v File (Part 1 of 2)

Module Name	Usage	Description
<*variation_name*>`_alt_mem_phy_addr_cmd`	All ALTMEMPHY variations	Generates the address and command structures.
<*variation_name*>`_alt_mem_phy_clk_reset`	All ALTMEMPHY variations	Instantiates PLL, DLL, and reset logic.

Table 9–8. Modules in <*variation_name*>_alt_mem_phy.v File (Part 2 of 2)

Module Name	Usage	Description
<*variation_name*>_alt_mem_phy_dp_io	All ALTMEMPHY variations	Generates the DQ, DQS, DM, and QVLD I/O pins.
<*variation_name*>_alt_mem_phy_mimic	DDR2/DDR SDRAM ALTMEMPHY variation	Creates the VT tracking mechanism for DDR and DDR2 SDRAM PHY IPs.
<*variation_name*>_alt_mem_phy_oct_delay	DDR2/DDR SDRAM ALTMEMPHY variation when dynamic OCT is enabled.	Generates the proper delay and duration for the OCT signals.
<*variation_name*>_alt_mem_phy_postamble	DDR2/DDR SDRAM ALTMEMPHY variations	Generates the postamble enable and disable scheme for DDR and DDR2 SDRAM PHY IPs.
<*variation_name*>_alt_mem_phy_read_dp	All ALTMEMPHY variations (unused for Stratix III or Stratix IV devices)	Takes read data from the I/O through a read path FIFO buffer, to transition from the resyncronization clock to the PHY clock.
<*variation_name*>_alt_mem_phy_read_dp_group	DDR2/DDR SDRAM ALTMEMPHY variations (Stratix III and Stratix IV devices only)	A per DQS group version of <*variation_name*>_alt_mem_phy_read_dp.
<*variation_name*>_alt_mem_phy_rdata_valid	DDR2/DDR SDRAM ALTMEMPHY variations	Generates read data valid signal to sequencer and controller.
<*variation_name*>_alt_mem_phy_seq_wrapper	All ALTMEMPHY variations	Generates sequencer for DDR and DDR2 SDRAM.
<*variation_name*>_alt_mem_phy_write_dp	All ALTMEMPHY variations	Generates the demultiplexing of data from half-rate to full-rate DDR data.
<*variation_name*>_alt_mem_phy_write_dp_fr	DDR2/DDR SDRAM ALTMEMPHY variations	A full-rate version of <*variation_name*>_alt_mem_phy_write_dp.

Table 9–9 lists the additional files generated by the HPC II that may be in your project directory.

Table 9–9. Controller-Generated Files (Part 1 of 2)

Filename	Description
alt_mem_ddrx_addr_cmd.v	Decodes internal protocol-related signals into memory address and command signals.
alt_mem_ddrx_addr_cmd_wrap.v	A wrapper that instantiates the **alt_mem_ddrx_addr_cmd.v** file.
alt_mem_ddrx_ddr2_odt_gen.v	Generates the on-die termination (ODT) control signal for DDR2 memory interfaces.
alt_mem_ddrx_ddr3_odt_gen.v	Generates the ODT control signal for DDR3 memory interfaces.
alt_mem_ddrx_odt_gen.v	Wrapper that instantiates **alt_mem_ddrx_ddr2_odt_gen.v** and **alt_mem_ddrx_ddr3_odt_gen.v**. This file also controls the ODT addressing scheme.
alt_mem_ddrx_rdwr_data_tmg.v	Decodes internal data burst related signals to memory data signals.
alt_mem_ddrx_arbiter.v	Contains logic that determines which command to execute based on certain schemes.
alt_mem_ddrx_burst_gen.v	Converts internal DRAM-aware commands to AFI signals.
alt_mem_ddrx_cmd_gen.v	Converts user requests to DRAM-aware commands.

Table 9–9. Controller-Generated Files (Part 2 of 2)

Filename	Description
alt_mem_ddrx_csr.v	Contains configuration registers.
alt_mem_ddrx_buffer.v	Contains buffer for local data.
alt_mem_ddrx_buffer_manager.v	Manages the allocation of buffers.
alt_mem_ddrx_burst_tracking.v	Tracks data received per local burst command.
alt_mem_ddrx_dataid_manager.v	Manages the IDs associated with data stored in buffer.
alt_mem_ddrx_fifo.v	Contains the FIFO buffer to store local data to create a link; is also used in `rdata_path` to store the read address and error address.
alt_mem_ddrx_list.v	Tracks the DRAM commands associated with the data stored internally.
alt_mem_ddrx_rdata_path.v	Contains read data path logic.
alt_mem_ddrx_wdata_path.v	Contains write data path logic.
alt_mem_ddrx_define.v	Defines common parameters used in the RTL files.
alt_mem_ddrx_ecc_decoder.v	Instantiates appropriate width of the ECC decoder logic.
alt_mem_ddrx_ecc_decoder_32_syn.v	Contains synthesizable 32-bit version of the ECC decoder.
alt_mem_ddrx_ecc_decoder_64_syn.v	Contains synthesizable 64-bit version of the ECC decoder.
alt_mem_ddrx_ecc_encoder.v	Instantiates appropriate width of the ECC encoder logic.
alt_mem_ddrx_ecc_encoder_32_syn.v	Contains synthesizable 32-bit version of the ECC decoder.
alt_mem_ddrx_ecc_encoder_64_syn.v	Contains synthesizable 64-bit version of the ECC decoder.
alt_mem_ddrx_ecc_encoder_decoder_wrapper.v	Wrapper that instantiates all ECC logic.
alt_mem_ddrx_input_if.v	Contains local input interface logic.
alt_mem_ddrx_mm_st_converter.v	Contains supporting logic for Avalon-MM interface.
alt_mem_ddrx_rank_timer.v	Contains a timer associated with rank timing.
alt_mem_ddrx_sideband.v	Contains supporting logic for user-controlled refresh and precharge signals.
alt_mem_ddrx_tbp.v	Contains command queue and associated logic for reordering features.
alt_mem_ddrx_timing_param.v	Contains timer logic associated with non-rank timing.
alt_mem_ddrx_controller_st_top.v	Wrapper that instantiates all submodules and configuration registers.
alt_mem_ddrx_controller_top.v	Wrapper that contains memory controller with Avalon-MM interface.
alt_mem_ddrx_controller.v	Wrapper that instantiates all submodules.

Generated Files for Memory Controllers with the UniPHY IP

Table 9–10 lists the generated directory structure and key files created with the MegaWizard Plug-In Manager, SOPC Builder, and Qsys.

Table 9–10. Generated Directory Structure and Key Files (Part 1 of 4)

Directory		File Name	Description
MegaWizard Plug-In Manager			
Synthesis Files	<*working_dir*>/	<*variation_name*>**.qip**	Quartus II IP file which refers to all generated files in the synthesis fileset. Include this file in your Quartus II project.
	<*working_dir*>/	<*variation_name*>**.v** or <*variation_name*>**.vhd**	Top-level wrapper synthesis files. **.v** is IEEE Encrypted Verilog. **.vhd** is generated VHDL.
	<*working_dir*>/<*variation_name*>/	<*variation_name*>**_0002.v**	UniPHY top-level wrapper.
	<*working_dir*>/<*variation_name*>/	***.v**, ***.sv**, ***.tcl**, ***.sdc**, ***.ppf**	RTL and constraints files for synthesis.
	<*working_dir*>/<*variation_name*>/	<*variation_name*>**_p0_pin_assignments.tcl**	Pin constraints script to be run after synthesis.
Simulation Files	<*working_dir*>/<*variation_name*>**_sim**/	<*variation_name*>**.v**	Top-level wrapper simulation files for both Verilog and VHDL.
	<*working_dir*>/<*variation_name*>**_sim**/<*subcomponent_module*>/	***.v**, ***.sv**, ***.vhd**, ***.vho**,***hex**, ***.mif**	RTL and constraints files for simulation. **.v** and **.sv** files are IEEE Encrypted Verilog. **.vhd** and **.vho** are generated VHDL.

Table 9–10. Generated Directory Structure and Key Files (Part 2 of 4)

Directory		File Name	Description
MegaWizard Plug-In Manager—Example Design Fileset			
Synthesis Files	<*variation_name*>**_example_design/example_project/**	<*variation_name*>**_example.qip**	Quartus II IP file that refers to all generated files in the synthesizable project.
	<*variation_name*>**_example_design/example_project/**	<*variation_name*>**_example.qpf**	Quartus II project for synthesis flow.
	<*variation_name*>**_example_design/example_project/**	<*variation_name*>**_example.qsf**	Quartus II project for synthesis flow.
	<*variation_name*>**_example_design/example_project/**<*variation_name*>**_example/**	<*variation_name*>**_example.v**	Top-level wrapper.
	<*variation_name*>**_example_design/example_project/**<*variation_name*>**_example/submodules/**	***.v**, ***.sv**, ***.tcl**, ***.sdc**, ***.ppf**	RTL and constraints files.
	<*variation_name*>**_example_design/example_project/**<*variation_name*>**_example/submodules/**	<*variation_name*>**_example_if0_p0_pin_assignments.tcl**	Pin constraints script to be run after synthesis. _if0 and _p0 are instance names. For more information, refer to Table 9–11 on page 9–45.

Table 9–10. Generated Directory Structure and Key Files (Part 3 of 4)

Directory		File Name	Description
Simulation Files	<*variation_name*>**_example_design/simulation/**	**generate_sim_verilog_example_design**.tcl	Run this file to generate the Verilog simulation example design.
	<*variation_name*>**_example_design/simulation/**	**generate_sim_vhdl_example_design.tcl**	Run this file to generate the VHDL simulation example design.
	<*variation_name*>**_example_design/simulation/**	**README.txt**	A text file with instructions about how to generate and run the simulation example design.
	<*variation_name*>**_example_design/simulation/verilog/mentor**	**run.do**	ModelSim script to simulate the generated Verilog example design.
	<*variation_name*>**_example_design/simulation/vhdl/mentor**	**run.do**	ModelSim script to simulate the generated VHDL example design.
	<*variation_name*>**_example_design/simulation/verilog/** <*variation_name*>**_sim/**	<*variation_name*>**_example_sim.v**	Top-level wrapper (Testbench) for Verilog.
	<*variation_name*>**_example_design/simulation/vhdl/** <*variation_name*>**_sim/**	<*variation_name*>**_example_sim.vhd**	Top-level wrapper (Testbench) for VHDL.
	<*variation_name*>**_example_design/simulation/** <*variation_name*>**_sim/verilog/submodules/**	***.v**, ***.sv**, ***.hex**, *.**mif**	RTL and ROM data for Verilog.
	<*variation_name*>**_example_design/simulation/** <*variation_name*>**_sim/vhdl/submodules/**	***.vhd**, ***.vho**, ***.hex**, ***.mif**	RTL and ROM data for VHDL.
SOPC Builder			
<*working_dir*>/		<*system_name*>**.qip**	Quartus II IP file that refers to all the generated files in the SOPC Builder project.
<*working_dir*>/		<*system_name*>**.v**	System top-level RTL.
<*working_dir*>/		<*module_name*>**.v**	Module wrapper RTL.
<*working_dir*>/<*module_name*>/		***.v**, ***.sv**, ***.tcl**, ***.sdc**, ***.ppf**	Subdirectory of TL and constraints for each system module.
Qsys			
<*working_dir*>/<*system_name*>**/synthesis/**		<*system_name*>**.qip**	Quartus II IP file that refers to all the generated files in the synthesis fileset.
<*working_dir*>/<*system_name*>**/synthesis/**		<*system_name*>**.v**	System top-level RTL for synthesis.

Table 9–10. Generated Directory Structure and Key Files (Part 4 of 4)

Directory	File Name	Description
<*working_dir*>/<*system_name*>**/simulation/**	<*system_name*>**.v** or <*variation_name*>**.vhd**	System top-level RTL for simulation. **.v** file is IEEE Encrypted Verilog. **.vhd** file is generated VHDL.
<*working_dir*>/<*system_name*>**/synthesis/submodules/**	***.v**, ***.sv**, ***.tcl**, ***.sdc**, ***.ppf**	RTL and constraints files for synthesis.
<*working_dir*>/<*system_name*>**/simulation/submodules/**	***.v**, ***.sv**, ***.hex**, ***.mif**	RTL and ROM data for simulation.

Table 9–11 lists the prefixes or instance names of submodule files within the memory interface IP. These instances are concatenated to form unique synthesis and simulation filenames.

Table 9–11. Prefixes of Submodule Files

Prefixes	Description
_c0	Specifies the controller.
_d0	Specifies the driver or traffic generator.
_dll0	Specifies the DLL.
_e0	Specifies the example design.
_if0	Specifies the memory Interface.
_m0	Specifies the AFI mux.
_oct0	Specifies the OCT.
_p0	Specifies the PHY.
_pll0	Specifies the PLL.
_s0	Specifies the sequencer.
_t0	Specifies the traffic generator status checker.

Parameterizing Memory Controllers with ALTMEMPHY IP

This section describes the parameters you can set for the DDR, DDR2, and DDR3 SDRAM memory controllers with the ALTMEMPHY IP.

The **Parameter Settings** page in the ALTMEMPHY parameter editor allows you to parameterize the following settings:

- Memory Settings
- PHY Settings
- Board Settings

The text window at the bottom of the MegaWizard Plug-In Manager displays information about the memory interface, warnings, and errors if you are trying to create something that is not supported. The **Finish** button is disabled until you correct all the errors indicated in this window.

The following sections describe the four tabs of the **Parameter Settings** page in more detail.

Memory Settings

Use this tab to apply the memory parameters from your memory manufacturer's data sheet.

Table 9–12 describes the **General Settings** available on the **Memory Settings** page of the ALTMEMPHY parameter editor.

Table 9–12. General Settings (Part 1 of 2)

Parameter Name	Description
Device family	Targets device family (for example, Arria II GX). The device family selected here must match the device family selected on page 2a of the parameter editor. For more information about selecting a device family, refer to the "Device Family Selection" section in the *Selecting your FPGA Device* chapter of the *External Memory Interface Handbook*.
Speed grade	Selects a particular speed grade of the device (for example, 2, 3, or 4 for the Arria II GX device family).
PLL reference clock frequency	Determines the clock frequency of the external input clock to the PLL. Ensure that you use three decimal points if the frequency is not a round number (for example, 166.667 MHz or 100 MHz) to avoid a functional simulation or a PLL locking problem.
Memory clock frequency	Determines the memory interface clock frequency. If you are operating a memory device below its maximum achievable frequency, ensure that you enter the actual frequency of operation rather than the maximum frequency achievable by the memory device. Also, ensure that you use three decimal points if the frequency is not a round number (for example, 333.333 MHz or 400 MHz) to avoid a functional simulation or a PLL locking issue.
Controller data rate	Selects the data rate for the memory controller. Sets the frequency of the controller to equal to either the memory interface frequency (full-rate) or half of the memory interface frequency (half-rate). The full-rate option is not available for DDR3 SDRAM devices.
Enable half rate bridge	This option is only available for HPC II full-rate controller. Turn on to keep the controller in the memory full clock domain while allowing the local side to run at half the memory clock speed, so that latency can be reduced.

Table 9–12. General Settings (Part 2 of 2)

Parameter Name	Description
Local interface clock frequency	Value that depends on the memory clock frequency and controller data rate.
Local interface width	Value that depends on the memory clock frequency and controller data rate.

When targeting a HardCopy device migration with performance improvement, the ALTMEMPHY IP should target the mid speed grade to ensure that the PLL and the PHY sequencer settings match. The compilation of the design can be executed in the faster speed grade.

Show in 'Memory Preset' List

Table 9–13 describes the options available to filter the **Memory Presets** that are displayed. This set of options is where you indicate whether you are creating a datapath for DDR3 SDRAM.

Table 9–13. Show in 'Memory Presets' List

Parameter Name	Description
Memory type	You can filter the type of memory to display, for example, DDR3 SDRAM.
Memory vendor	You can filter the memory types by vendor. JEDEC is also one of the options, allowing you to choose the JEDEC specifications. If your chosen vendor is not listed, you can choose JEDEC for the DDR3 SDRAM interfaces. Then, pick a device that has similar specifications to your chosen device and check the values of each parameter. Make sure you change the each parameter value to match your device specifications.
Memory format	You can filter the type of memory by format, for example, discrete devices or DIMM packages.
Maximum frequency	You can filter the type of memory by the maximum operating frequency.

Memory Presets

Pick a device in the **Memory Presets** list that is closest or the same as the actual memory device that you are using. Then, click the **Modify Parameters** button to parameterize the following settings in the **Preset Editor** dialog box:

- Memory attributes—These are the settings that determine your system's number of DQ, DQ strobe (DQS), address, and memory clock pins.
- Memory initialization options—These settings are stored in the memory mode registers as part of the initialization process.
- Memory timing parameters—These are the parameters that create and time-constrain the PHY.

Even though the device you are using is listed in **Memory Presets**, ensure that the settings in the **Preset Editor** dialog box are accurate, as some parameters may have been updated in the memory device datasheets.

You can change the parameters with a white background to reflect your system. You can also change the parameters with a gray background so the device parameters match the device you are using. These parameters in gray background are characteristics of the chosen memory device and changing them creates a new custom memory preset. If you click **Save As** (at the bottom left of the page) and save the new settings in the <*quartus_install_dir*>**\quartus\common\ip\altera\altmemphy\lib** directory, you can use this new memory preset in other Quartus II projects created in the same version of the software.

When you click **Save**, the new memory preset appears at the bottom of the **Memory Presets** list in the **Memory Settings** tab.

If you save the new settings in a directory other than the default directory, click **Load Preset** in the **Memory Settings** tab to load the settings into the **Memory Presets** list.

The **Advanced** option shows the percentage of memory specification that is calibrated by the FPGA. The percentage values are estimated by Altera based on the process variation.

Preset Editor Settings for DDR and DDR2 SDRAM

Table 9–14 through Table 9–16 describe the DDR2 SDRAM parameters available for memory attributes, initialization options, and timing parameters. DDR SDRAM has the same parameters, but their value ranges are different than DDR2 SDRAM.

Table 9–14. DDR2 SDRAM Attributes Settings (Part 1 of 2)

Parameter Name	Range (1)	Units	Description
Output clock pairs from FPGA	1–6	pairs	Defines the number of differential clock pairs driven from the FPGA to the memory. More clock pairs reduce the loading of each output when interfacing with multiple devices. Memory clock pins use the signal splitter feature in Arria II GX, Stratix III, and Stratix IV devices for differential signaling.
Total Memory chip selects	1, 2, 4, or 8	bits	Sets the number of chip selects in your memory interface. The number of chip selects defines the depth of your memory. You are limited to the range shown as the local side binary encodes the chip select address. You can set this value to the next higher number if the range does not meet your specifications. However, the highest address space of the ALTMEMPHY megafunction is not mapped to any of the actual memory address. The ALTMEMPHY megafunction works with multiple chip selects and calibrates against all chip select, `mem_cs_n` signals.
Memory interface DQ width	4–288	bits	Defines the total number of DQ pins on the memory interface. If you are interfacing with multiple devices, multiply the number of devices with the number of DQ pins per device. Even though the GUI allows you to choose 288-bit DQ width, the interface data width is limited by the number of pins on the device. For best performance, have the whole interface on one side of the device.

Table 9–14. DDR2 SDRAM Attributes Settings (Part 2 of 2)

Parameter Name	Range (1)	Units	Description
Memory vendor	JEDEC, Micron, Qimonda, Samsung, Hynix, Elpida, Nanya, other	—	Lists the name of the memory vendor for all supported memory standards.
Memory format	Discrete Device, Unbuffered DIMM, Registered DIMM	—	Specifies whether you are interfacing with devices or modules. SODIMM is supported under unbuffered or registered DIMMs.
Maximum memory frequency	See the memory device datasheet	MHz	Sets the maximum frequency supported by the memory.
Column address width	9–11	bits	Defines the number of column address bits for your interface.
Row address width	13–16	bits	Defines the number of row address bits for your interface.
Bank address width	2 or 3	bits	Defines the number of bank address bits for your interface.
Chip selects per DIMM	1 or 2	bits	Defines the number of chip selects on each DIMM in your interface.
DQ bits per DQS bit	4 or 8	bits	Defines the number of data (DQ) bits for each data strobe (DQS) pin.
Precharge address bit	8 or 10	bits	Selects the bit of the address bus to use as the precharge address bit.
Drive DM pins from FPGA	Yes or No	—	Specifies whether you are using DM pins for write operation. Altera devices do not support DM pins in ×4 mode.
Maximum memory frequency for CAS latency 3.0	80–533	MHz	Specifies the frequency limits from the memory data sheet per given CAS latency. The ALTMEMPHY parameter editor generates a warning if the operating frequency with your chosen CAS latency exceeds this number.
Maximum memory frequency for CAS latency 4.0			
Maximum memory frequency for CAS latency 5.0			
Maximum memory frequency for CAS latency 6.0			

Note to Table 9–14:

(1) The range values depend on the actual memory device used.

Table 9–15. DDR2 SDRAM Initialization Options

Parameter Name	Range	Units	Description
Memory burst length	4 or 8	beats	Sets the number of words read or written per transaction. Memory burst length of four equates to local burst length of one in half-rate designs and to local burst length of two in full-rate designs.
Memory burst ordering	Sequential or Interleaved	—	Controls the order in which data is transferred between memory and the FPGA during a read transaction. For more information, refer to the memory device datasheet.
Enable the DLL in the memory devices	Yes or No	—	Enables the DLL in the memory device when set to **Yes**. You must always enable the DLL in the memory device as Altera does not guarantee any ALTMEMPHY operation when the DLL is turned off. All timings from the memory devices are invalid when the DLL is turned off.
Memory drive strength setting	Normal or Reduced	—	Controls the drive strength of the memory device's output buffers. Reduced drive strength is not supported on all memory devices. The default option is normal.
Memory ODT setting	Disabled, 50, 75, 150	W	Sets the memory ODT value. Not available in DDR SDRAM interfaces.
Memory CAS latency setting	3, 4, 5, 6	cycles	Sets the delay in clock cycles from the read command to the first output data from the memory.

Table 9–16. DDR2 SDRAM Timing Parameter Settings (1) (Part 1 of 3)

Parameter Name	Range	Units	Description
t_{INIT}	0.001–1000	µs	Minimum memory initialization time. After reset, the controller does not issue any commands to the memory during this period.
t_{MRD}	2–39	ns	Minimum load mode register command period. The controller waits for this period of time after issuing a load mode register command before issuing any other commands. t_{MRD} is specified in ns in the DDR2 SDRAM high-performance controller and in terms of t_{CK} cycles in Micron's device datasheet. Convert t_{MRD} to ns by multiplying the number of cycles specified in the datasheet times t_{CK}, where t_{CK} is the memory operation frequency and not the memory device's t_{CK}.
t_{RAS}	8–200	ns	Minimum active to precharge time. The controller waits for this period of time after issuing an active command before issuing a precharge command to the same bank.
t_{RCD}	4–65	ns	Minimum active to read-write time. The controller does not issue read or write commands to a bank during this period of time after issuing an active command.
t_{RP}	4–65	ns	Minimum precharge command period. The controller does not access the bank for this period of time after issuing a precharge command.
t_{REFI}	1–65534	µs	Maximum interval between refresh commands. The controller performs regular refresh at this interval unless user-controlled refresh is turned on.

Table 9–16. DDR2 SDRAM Timing Parameter Settings (1) (Part 2 of 3)

Parameter Name	Range	Units	Description
t_{RFC}	14–1651	ns	Minimum autorefresh command period. The length of time the controller waits before doing anything else after issuing an auto-refresh command.
t_{WR}	4–65	ns	Minimum write recovery time. The controller waits for this period of time after the end of a write transaction before issuing a precharge command.
t_{WTR}	1–3	t_{CK}	Minimum write-to-read command delay. The controller waits for this period of time after the end of a write command before issuing a subsequent read command to the same bank. This timing parameter is specified in clock cycles and the value is rounded off to the next integer.
t_{AC}	300–750	ps	DQ output access time from CK/CK# signals.
t_{DQSCK}	100–750	ps	DQS output access time from CK/CK# signals.
t_{DQSQ}	100–500	ps	The maximum DQS to DQ skew; DQS to last DQ valid, per group, per access.
t_{DQSS}	0–0.3	t_{CK}	Positive DQS latching edge to associated clock edge.
t_{DS}	10–600	ps	DQ and DM input setup time relative to DQS, which has a derated value depending on the slew rate of the DQS (for both DDR and DDR2 SDRAM interfaces) and whether DQS is single-ended or differential (for DDR2 SDRAM interfaces). Ensure that you are using the correct number and that the value entered is referenced to V_{REF}(dc), not V_{IH}(ac) min or V_{IL}(ac) max. Refer to "Derating Memory Setup and Hold Timing" on page 9–58 for more information about how to derate this specification.
t_{DH}	10–600	ps	DQ and DM input hold time relative to DQS, which has a derated value depending on the slew rate of the DQS (for both DDR and DDR2 SDRAM interfaces) and whether DQS is single-ended or differential (for DDR2 SDRAM interfaces). Ensure that you are using the correct number and that the value entered is referenced to V_{REF}(dc), not V_{IH}(dc) min or V_{IL}(dc) max. Refer to "Derating Memory Setup and Hold Timing" on page 9–58 for more information about how to derate this specification.
t_{DSH}	0.1–0.5	t_{CK}	DQS falling edge hold time from CK.
t_{DSS}	0.1–0.5	t_{CK}	DQS falling edge to CK setup.
t_{IH}	100–1000	ps	Address and control input hold time, which has a derated value depending on the slew rate of the CK and CK# clocks and the address and command signals. Ensure that you are using the correct number and that the value entered is referenced to V_{REF}(dc), not V_{IH}(dc) min or V_{IL}(dc) max. Refer to "Derating Memory Setup and Hold Timing" on page 9–58 for more information about how to derate this specification.

Table 9–16. DDR2 SDRAM Timing Parameter Settings (1) (Part 3 of 3)

Parameter Name	Range	Units	Description
t_{IS}	100–1000	ps	Address and control input setup time, which has a derated value depending on the slew rate of the CK and CK# clocks and the address and command signals. Ensure that you are using the correct number and that the value entered is referenced to V_{REF}(dc), not V_{IH}(ac) min or V_{IL}(ac) max. Refer to "Derating Memory Setup and Hold Timing" on page 9–58 for more information about how to derate this specification.
t_{QHS}	100–700	ps	The maximum data hold skew factor.
t_{RRD}	2.06–64	ns	The activate command to activate time, per device, RAS to RAS delay timing parameter.
t_{FAW}	7.69–256	ns	The four-activate window time, per device.
t_{RTP}	2.06–64	ns	Read to precharge time.

Note to Table 9–16:

(1) Refer to the memory device data sheet for the parameter range. Some of the parameters are listed in a clock cycle (t_{CK}) unit. If the MegaWizard Plug-In Manager requires you to enter the value in a time unit (ps or ns), convert the number by multiplying it with the clock period of your interface (and not the maximum clock period listed in the memory data sheet).

Preset Editor Settings for DDR3 SDRAM

Table 9–17 through Table 9–19 describe the DDR3 SDRAM parameters available for memory attributes, initialization options, and timing parameters.

Table 9–17. DDR3 SDRAM Attributes Settings (Part 1 of 3)

Parameter Name	Range (1)	Units	Description
Output clock pairs from FPGA	1–6	pairs	Defines the number of differential clock pairs driven from the FPGA to the memory. Memory clock pins use the signal splitter feature in Arria II GX devices for differential signaling. The ALTMEMPHY parameter editor displays an error on the bottom of the window if you choose more than one for DDR3 SDRAM interfaces.
Total Memory chip selects	1, 2, 4, or 8	bits	Sets the number of chip selects in your memory interface. The number of chip selects defines the depth of your memory. You are limited to the range shown as the local side binary encodes the chip select address.
Memory interface DQ width	4–288	bits	Defines the total number of DQ pins on the memory interface. If you are interfacing with multiple devices, multiply the number of devices with the number of DQ pins per device. Even though the GUI allows you to choose 288-bit DQ width, DDR3 SDRAM variations are only supported up to 80-bit width due to restrictions in the board layout which affects timing at higher data width. Furthermore, the interface data width is limited by the number of pins on the device. For best performance, have the whole interface on one side of the device.

Table 9–17. DDR3 SDRAM Attributes Settings (Part 2 of 3)

Parameter Name	Range (1)	Units	Description
Mirror addressing	—	—	On multiple rank DDR3 SDRAM DIMMs address signals are routed differently to each rank; referred to in the JEDEC specification as address mirroring. Enter ranks with mirrored addresses in this field. There is one bit per chip select. For example, for four chip selects, enter 1011 to mirror the address on chip select #3, #1, and #0.
Memory vendor	Elpida, JEDEC, Micron, Samsung, Hynix, Nanya, other	—	Lists the name of the memory vendor for all supported memory standards.
Memory format	Discrete Device	—	Arria II GX devices only support DDR3 SDRAM components without leveling, for example, **Discrete Device** memory format.
Maximum memory frequency	See the memory device datasheet	MHz	Sets the maximum frequency supported by the memory.
Column address width	10–12	bits	Defines the number of column address bits for your interface.
Row address width	12–16	bits	Defines the number of row address bits for your interface. If your DDR3 SDRAM device's row address bus is 12-bit wide, set the row address width to **13** and set the 13th bit to logic-level low (or leave the 13th bit unconnected to the memory device) in the top-level file.
Bank address width	3	bits	Defines the number of bank address bits for your interface.
Chip selects per device	1 or 2	bits	Defines the number of chip selects on each device in your interface. Currently, calibration is done with all ranks but you can only perform timing analysis with one.
DQ bits per DQS bit	4 or 8	bits	Defines the number of data (DQ) bits for each data strobe (DQS) pin.
Drive DM pins from FPGA	Yes or No	—	Specifies whether you are using DM pins for write operation. Altera devices do not support DM pins with ×4 mode.

Table 9–17. DDR3 SDRAM Attributes Settings (Part 3 of 3)

Parameter Name	Range (1)	Units	Description
Maximum memory frequency for CAS latency 5.0	80–700	MHz	Specifies the frequency limits from the memory data sheet per given CAS latency. The ALTMEMPHY MegaWizard Plug-In Manager generates a warning if the operating frequency with your chosen CAS latency exceeds this number. The lowest frequency supported by DDR3 SDRAM devices is 300 MHz.
Maximum memory frequency for CAS latency 6.0			
Maximum memory frequency for CAS latency 7.0			
Maximum memory frequency for CAS latency 8.0			
Maximum memory frequency for CAS latency 9.0			
Maximum memory frequency for CAS latency 10.0			

Note to Table 9–17:

(1) The range values depend on the actual memory device used.

Table 9–18. DDR3 SDRAM Initialization Options (Part 1 of 2)

Parameter Name	Range	Units	Description
Memory burst length	4, 8, on-the-fly	beats	Sets the number of words read or written per transaction.
Memory burst ordering	Sequential or Interleaved	—	Controls the order in which data is transferred between memory and the FPGA during a read transaction. For more information, refer to the memory device datasheet.
DLL precharge power down	Fast exit or Slow exit	—	Sets the mode register setting to disable (**Slow exit**) or enable (**Fast exit**) the memory DLL when CKE is disabled.
Enable the DLL in the memory devices	Yes or No	—	Enables the DLL in the memory device when set to **Yes**. You must always enable the DLL in the memory device as Altera does not guarantee any ALTMEMPHY operation when the DLL is turned off. All timings from the memory devices are invalid when the DLL is turned off.
ODT `Rtt` nominal value	ODT disable, RZQ/4, RZQ/2, RZQ/6	W	RZQ in DDR3 SDRAM interfaces are set to 240 Ω. Sets the on-die termination (ODT) value to either 60 Ω (**RZQ/4**), 120 Ω (**RZQ/2**), or 40 Ω (**RZQ/6**). Set this to **ODT disable** if you are not planning to use ODT. For a single-ranked DIMM, set this to **RZQ/4**.

Table 9–18. DDR3 SDRAM Initialization Options (Part 2 of 2)

Parameter Name	Range	Units	Description
Dynamic ODT (`Rtt_WR`) value	Dynamic ODT off, RZQ/4, RZQ/2	W	RZQ in DDR3 SDRAM interfaces are set to 240 Ω. Sets the memory ODT value during write operations to 60 Ω (**RZQ/4**) or 120 Ω (**RZQ/2**). As ALTMEMPHY only supports single rank DIMMs, you do not need this option (set to **Dynamic ODT off**).
Output driver impedance	RZQ/6 (Reserved) or RZQ/7	W	RZQ in DDR3 SDRAM interfaces are set to 240 Ω. Sets the output driver impedance from the memory device. Some devices may not have **RZQ/6** available as an option. Be sure to check the memory device datasheet before choosing this option.
Memory CAS latency setting	5.0, 6.0, 7.0, 8.0, 9.0, 10.0	cycles	Sets the delay in clock cycles from the read command to the first output data from the memory.
Memory additive CAS latency setting	Disable, CL – 1, CL – 2	cycles	Allows you to add extra latency in addition to the CAS latency setting.
Memory write CAS latency setting (CWL)	5.0, 6.0, 7.0, 8.0	cycles	Sets the delay in clock cycles from the write command to the first expected data to the memory.
Memory partial array self refresh	Full array, Half array {BA[2:0]=000,001, 010,011}, Quarter array {BA[2:0]=000,001} , Eighth array {BA[2:0]=000}, Three Quarters array {BA[2:0]=010,011, 100,101,110,111}, Half array {BA[2:0]=100,101, 110,111}, Quarter array {BA[2:0]=110, 111}, Eighth array {BA[2:0]=111}	—	Determine whether you want to self-refresh only certain arrays instead of the full array. According to the DDR3 SDRAM specification, data located in the array beyond the specified address range are lost if **self refresh** is entered when you use this. This option is not supported by the DDR3 SDRAM Controller with ALTMEMPHY IP, so set to **Full Array** if you are using the Altera controller.
Memory auto self refresh method	Manual SR reference (SRT) or ASR enable (Optional)	—	Sets the auto self-refresh method for the memory device. The DDR3 SDRAM Controller with ALTMEMPHY IP currently does not support the ASR option that you need for extended temperature memory self-refresh.
Memory self refresh range	Normal or Extended	—	Determines the temperature range for self refresh. You need to also use the optional auto self refresh option when using this option. The Altera controller currently does not support the extended temperature self-refresh operation.

Table 9–19. DDR3 SDRAM Timing Parameter Settings (Part 1 of 2) *(1)*

Parameter Name	Range	Units	Description
Time to hold memory reset before beginning calibration	0–1000000	µs	Minimum time to hold the reset after a power cycle before issuing the MRS commands during the DDR3 SDRAM device initialization process.
t_{INIT}	0.001–1000	µs	Minimum memory initialization time. After reset, the controller does not issue any commands to the memory during this period.
t_{MRD}	2–39	ns	Minimum load mode register command period. The controller waits for this period of time after issuing a load mode register command before issuing any other commands. t_{MRD} is specified in ns in the DDR3 SDRAM high-performance controller and in terms of t_{CK} cycles in Micron's device datasheet. Convert t_{MRD} to ns by multiplying the number of cycles specified in the datasheet times t_{CK}, where t_{CK} is the memory operation frequency and not the memory device's t_{CK}.
t_{RAS}	8–200	ns	Minimum active to precharge time. The controller waits for this period of time after issuing an active command before issuing a precharge command to the same bank.
t_{RCD}	4–65	ns	Minimum active to read-write time. The controller does not issue read or write commands to a bank during this period of time after issuing an active command.
t_{RP}	4–65	ns	Minimum precharge command period. The controller does not access the bank for this period of time after issuing a precharge command.
t_{REFI}	1–65534	µs	Maximum interval between refresh commands. The controller performs regular refresh at this interval unless user-controlled refresh is turned on.
t_{RFC}	14–1651	ns	Minimum autorefresh command period. The length of time the controller waits before doing anything else after issuing an auto-refresh command.
t_{WR}	4–65	ns	Minimum write recovery time. The controller waits for this period of time after the end of a write transaction before issuing a precharge command.
t_{WTR}	1–6	t_{CK}	Minimum write-to-read command delay. The controller waits for this period of time after the end of a write command before issuing a subsequent read command to the same bank. This timing parameter is specified in clock cycles and the value is rounded off to the next integer.
t_{AC}	0–750	ps	DQ output access time.
t_{DQSCK}	50–750	ps	DQS output access time from CK/CK# signals.
t_{DQSQ}	50–500	ps	The maximum DQS to DQ skew; DQS to last DQ valid, per group, per access.
t_{DQSS}	0–0.3	t_{CK}	Positive DQS latching edge to associated clock edge.

Table 9–19. DDR3 SDRAM Timing Parameter Settings (Part 2 of 2) *(1)*

Parameter Name	Range	Units	Description
t_{DH}	10–600	ps	DQ and DM input hold time relative to DQS, which has a derated value depending on the slew rate of the differential DQS and DQ/DM signals. Ensure that you are using the correct number and that the value entered is referenced to V_{REF}(dc), not V_{IH}(dc) min or V_{IL}(dc) max. Refer to "Derating Memory Setup and Hold Timing" on page 9–58 for more information about how to derate this specification.
t_{DS}	10–600	ps	DQ and DM input setup time relative to DQS, which has a derated value depending on the slew rate of the differential DQS signals and DQ/DM signals. Ensure that you are using the correct number and that the value entered is referenced to V_{REF}(dc), not V_{IH}(ac) min or V_{IL}(ac) max. Refer to "Derating Memory Setup and Hold Timing" on page 9–58 for more information about how to derate this specification.
t_{DSH}	0.1–0.5	t_{CK}	DQS falling edge hold time from CK.
t_{DSS}	0.1–0.5	t_{CK}	DQS falling edge to CK setup.
t_{IH}	50–1000	ps	Address and control input hold time, which has a derated value depending on the slew rate of the CK and CK# clocks and the address and command signals. Ensure that you are using the correct number and that the value entered is referenced to $_{VREF}$(dc), not V_{IH}(dc) min or V_{IL}(dc) max. Refer to "Derating Memory Setup and Hold Timing" on page 9–58 for more information about how to derate this specification.
t_{IS}	65–1000	ps	Address and control input setup time, which has a derated value depending on the slew rate of the CK and CK# clocks and the address and command signals. Ensure that you are using the correct number and that the value entered is referenced to V_{REF}(dc), not V_{IH}(ac) min or V_{IL}(ac) max. Refer to "Derating Memory Setup and Hold Timing" on page 9–58 for more information about how to derate this specification.
t_{QHS}	0–700	ps	The maximum data hold skew factor.
t_{QH}	0.1–0.6	t_{CK}	DQ output hold time.
t_{RRD}	2.06–64	ns	The activate to activate time, per device, RAS to RAS delay timing parameter.
t_{FAW}	7.69–256	ns	The four-activate window time, per device.
t_{RTP}	2.06–64	ns	Read to precharge time.

Note to Table 9–19:

(1) Refer to the memory device data sheet for the parameter range. Some of the parameters are listed in a clock cycle (t_{CK}) unit. If the MegaWizard Plug-In Manager requires you to enter the value in a time unit (ps or ns), convert the number by multiplying it with the clock period of your interface (and not the maximum clock period listed in the memory data sheet).

Derating Memory Setup and Hold Timing

Because the base setup and hold time specifications from the memory device datasheet assume input slew rates that may not be true for Altera devices, derate and update the following memory device specifications in the **Preset Editor** dialog box:

- t_{DS}
- t_{DH}
- t_{IH}
- t_{IS}

For Arria II GX and Stratix IV devices (excluding DDR SDRAM), you need not derate using the **Preset Editor**. You only need to enter the parameters referenced to V_{REF}, and the deration is done automatically when you enter the slew rate information on the **Board Settings** tab.

After derating the values, you then need to normalize the derated value because Altera input and output timing specifications are referenced to V_{REF}. However, JEDEC base setup time specifications are referenced to V_{IH}/V_{IL} AC levels; JEDEC base hold time specifications are referenced to V_{IH}/V_{IL} DC levels.

When the memory device setup and hold time numbers are derated and normalized to V_{REF}, update these values in the **Preset Editor** dialog box to ensure that your timing constraints are correct.

Example 9–1. Derating DDR2 SDRAM

For example, according to JEDEC, 400-MHz DDR2 SDRAM has the following specifications, assuming 1V/ns DQ slew rate rising signal and 2V/ns differential slew rate:

- Base t_{DS} = 50
- Base t_{DH} = 125
- $V_{IH}(ac) = V_{REF} + 0.2$ V
- $V_{IH}(dc) = V_{REF} + 0.125$V
- $V_{IL}(ac) = V_{REF} – 0.2$ V
- $V_{IL}(dc) = V_{REF} – 0.125$ V

JEDEC lists two different sets of base and derating numbers for t_{DS} and t_{DH} specifications, whether you are using single-ended or differential DQS signaling, for any DDR2 SDRAM components with a maximum frequency up to 267 MHz. In addition, the $V_{IL}(ac)$ and $V_{IH}(ac)$ values may also be different for those devices.

The V_{REF} referenced setup and hold signals for a rising edge are:

t_{DS} (V_{REF}) = Base t_{DS} + delta t_{DS} + ($V_{IH}(ac) – V_{REF}$)/slew_rate = 50 + 0 + 200 = 250 ps

t_{DH} (V_{REF}) = Base t_{DH} + delta t_{DH} + ($V_{IH}(dc) – V_{REF}$)/slew_rate = 125 + 0 + 67.5 = 192.5 ps

If the output slew rate of the write data is different from 1V/ns, you have to first derate the t_{DS} and t_{DH} values, then translate these AC/DC level specs to V_{REF} specification.

For a 2V/ns DQ slew rate rising signal and 2V/ns DQS-DQSn slew rate:

t_{DS} (V_{REF}) = Base t_{DS} + delta t_{DS} + (V_{IH}(ac) – V_{REF})/slew_rate = 25 + 100 + 100 = 225 ps

t_{DH} (V_{REF}) = Base t_{DH} + delta t_{DH} + (V_{IH}(dc) – V_{REF})/slew_rate = 100 + 45 + 62.5 = 207.5 ps

For a 0.5V/ns DQ slew rate rising signal and 1V/ns DQS-DQSn slew rate:

t_{DS} (V_{REF}) = Base t_{DS} + delta t_{DS} + (V_{IH}(ac) – V_{REF})/slew_rate = 25 + 0 + 400 = 425 ps

t_{DH} (V_{REF}) = Base t_{DH} + delta t_{DH} + (V_{IH}(dc) – V_{REF})/slew_rate = 100 – 65 + 250 = 285 ps

A similar approach can be taken to address/command slew rate derating. For t_{IS}/t_{IH} the slew rate used in the derating equations is the address/command slew rate; for t_{DS}/t_{DH} the DQ slew rate is used.

Example 9–2. Derating DDR3 SDRAM

For example, according to JEDEC, 533-MHz DDR3 SDRAM has the following specifications, assuming 1V/ns DQ slew rate rising signal and 2V/ns DQS-DQSn slew rate:

- Base t_{DS} = 25
- Base t_{DH} = 100
- V_{IH}(ac) = V_{REF} + 0.175 V
- V_{IH}(dc) = V_{REF} + 0.100 V
- V_{IL}(ac) = V_{REF} – 0.175 V
- V_{IL}(dc) = V_{REF} – 0.100 V

The V_{REF} referenced setup and hold signals for a rising edge are:

t_{DS} (V_{REF}) = Base t_{DS} + delta t_{DS} + (V_{IH}(ac) – V_{REF})/slew_rate = 25 + 0 + 175 = 200 ps

t_{DH} (V_{REF}) = Base t_{DH} + delta t_{DH} + (V_{IH}(dc) – V_{REF})/slew_rate = 100 + 0 + 100 = 200 ps

If the output slew rate of the write data is different from 1V/ns, you have to first derate the t_{DS} and t_{DH} values, then translate these AC/DC level specs to V_{REF} specification.

For a 2V/ns DQ slew rate rising signal and 2V/ns DQS-DQSn slew rate:

t_{DS} (V_{REF}) = Base t_{DS} + delta t_{DS} + (V_{IH}(ac) – V_{REF})/slew_rate = 25 + 88 + 87.5 = 200.5 ps

t_{DH} (V_{REF}) = Base t_{DH} + delta t_{DH} + (V_{IH}(dc) – V_{REF})/slew_rate = 100 + 50 + 50 = 200 ps

For a 0.5V/ns DQ slew rate rising signal and 1V/ns DQS-DQSn slew rate:

t_{DS} (V_{REF}) = Base t_{DS} + delta t_{DS} + (V_{IH}(ac) – V_{REF})/slew_rate = 25 + 5 + 350 = 380 ps

t_{DH} (V_{REF}) = Base t_{DH} + delta t_{DH} + (V_{IH}(dc) – V_{REF})/slew_rate = 100 + 10 + 200 = 310 ps

PHY Settings

Click **Next** or the **PHY Settings** tab to set the options described in Table 9–20. The options are available if they apply to the target Altera device.

Table 9–20. ALTMEMPHY PHY Settings (Part 1 of 3)

Parameter Name	Applicable Device Families		Description
	DDR/DDR2 SDRAM	DDR3 SDRAM	
Use dedicated PLL outputs to drive memory clocks	HardCopy II and Stratix II (prototyping for HardCopy II)	Not supported	Turn on to use dedicated PLL outputs to generate the external memory clocks, which is required for HardCopy II ASICs and their Stratix II FPGA prototypes. When turned off, the DDIO output registers generate the clock outputs. When you use the DDIO output registers for the memory clock, both the memory clock and the DQS signals are well aligned and easily meets the t_{DQSS} specification. However, when the dedicated clock outputs are for the memory clock, the memory clock and the DQS signals are not aligned properly and requires a positive phase offset from the PLL to align the signals together.
Dedicated memory clock phase	HardCopy II and Stratix II (prototyping for HardCopy II)	Not supported	The required phase shift to align the CK/CK# signals with DQS/DQS# signals when using dedicated PLL outputs to drive memory clocks.
Use differential DQS	Arria II GX, Stratix III, and Stratix IV	Not supported	Enable this feature for better signal integrity. Recommended for operation at 333 MHz or higher. An option for DDR2 SDRAM only, as DDR SDRAM does not support differential DQSS.
Enable external access to reconfigure PLL prior to calibration	HardCopy II, Stratix II, Stratix III, and Stratix IV (prototyping for HardCopy II)	HardCopy II	When enabling this option for HardCopy II, Stratix II, Stratix III, and Stratix IV devices, the inputs to the ALTPLL_RECONFIG megafunction are brought to the top level for debugging purposes. This option allows you to reconfigure the PLL before calibration to adjust, if necessary, the phase of the memory clock (`mem_clk_2x`) before the start of the calibration of the resynchronization clock on the read side. The calibration of the resynchronization clock on the read side depends on the phase of the memory clock on the write side.

Table 9–20. ALTMEMPHY PHY Settings (Part 2 of 3)

Parameter Name	Applicable Device Families		Description
	DDR/DDR2 SDRAM	DDR3 SDRAM	
Instantiate DLL externally	All supported device families, except for Cyclone® III devices	All supported device families	Use this option with Stratix III, Stratix IV, HardCopy III, or HardCopy IV devices, if you want to apply a non-standard phase shift to the DQS capture clock. The ALTMEMPHY DLL offsetting I/O can then be connected to the external DLL and the Offset Control Block. As Cyclone III devices do not have DLLs, this feature is not supported.
Enable dynamic parallel on-chip termination	Stratix III and Stratix IV	Not supported	This option provides I/O impedance matching and termination capabilities. The ALTMEMPHY megafunction enables parallel termination during reads and series termination during writes with this option checked. Only applicable for DDR and DDR2 SDRAM interfaces where DQ and DQS are bidirectional. Using the dynamic termination requires that you use the OCT calibration block, which may impose a restriction on your DQS/DQ pin placements depending on your R_{UP}/R_{DN} pin locations. Although DDR SDRAM does not support ODT, dynamic OCT is still supported in Altera FPGAs. For more information, refer to the *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* or the *External Memory Interfaces in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.
Clock phase	Arria II GX, Arria GX, Cyclone III, HardCopy II, Stratix II, and Stratix II GX	Arria II GX	Adjusting the address and command phase can improve the address and command setup and hold margins at the memory device to compensate for the propagation delays that vary with different loadings. You have a choice of 0°, 90°, 180°, and 270°, based on the rising and falling edge of the `phy_clk` and `write_clk` signals. In Stratix IV and Stratix III devices, the clock phase is set to **dedicated**.
Dedicated clock phase	Stratix III and Stratix IV	Not supported	When you use a dedicated PLL output for address and command, you can choose any legal PLL phase shift to improve setup and hold for the address and command signals. You can set this value to between 180° and 359°, the default is 240°. However, generally PHY timing requires a value of greater than 240° for half-rate designs and 270° for full-rate designs.

Table 9–20. ALTMEMPHY PHY Settings (Part 3 of 3)

Parameter Name	Applicable Device Families		Description
	DDR/DDR2 SDRAM	DDR3 SDRAM	
Board skew	All supported device families except Arria II GX and Stratix IV devices	Not supported	Maximum skew across any two memory interface signals for the whole interface from the FPGA to the memory (either a discrete memory device or a DIMM). This parameter includes all types of signals (data, strobe, clock, address, and command signals). You need to input the worst-case skew, whether it is within a DQS/DQ group, or across all groups, or across the address and command and clocks signals. This parameter generates the timing constraints in the **.sdc**.
Autocalibration simulation options	All supported device families		Choose between **Full Calibration** (long simulation time), **Quick Calibration**, or **Skip Calibration**. For more information, refer to the "Simulation Options" section in the *Simulating Memory IP* chapter.

Board Settings

Click **Next** or the **Board Settings** tab to set the options described in Table 9–21. The board settings parameters are set to model the board level effects in the timing analysis. The options are available if you choose Arria II GX or Stratix IV device for your interface. Otherwise, the options are disabled. The options are also disabled for all devices using DDR SDRAM.

Table 9–21. ALTMEMPHY Board Settings (Part 1 of 2)

Parameter Name	Units	Description
Number of slots/discrete devices	—	Sets the single-rank or multi-rank configuration.
CK/CK# slew rate (differential)	V/ns	Sets the differential slew rate for the CK and CK# signals.
Addr/command slew rate	V/ns	Sets the slew rate for the address and command signals.
DQ/DQS# slew rate (differential)	V/ns	Sets the differential slew rate for the DQ and DQS# signals.
DQ slew rate	V/ns	Sets the slew rate for the DQ signals.
Addr/command eye reduction (setup)	ns	Sets the reduction in the eye diagram on the setup side due to the ISI on the address and command signals.
Addr/command eye reduction (hold)	ns	Sets the reduction in the eye diagram on the hold side due to the ISI on the address and command signals.
DQ eye reduction	ns	Sets the total reduction in the eye diagram on the setup side due to the ISI on the DQ signals.
Delta DQS arrival time	ns	Sets the increase of variation on the range of arrival times of DQS due to ISI.
Max skew between DIMMs/devices	ns	Sets the largest skew or propagation delay on the DQ signals between ranks, especially true for DIMMs in different slots. This value affects the Resynchronization margin for the DDR2 interfaces in multi-rank configurations for both DIMMs and devices.
Max skew within DQS group	ns	Sets the largest skew between the DQ pins in a DQS group. This value affects the Read Capture and Write margins for the DDR2 interfaces in all configurations (single- or multi-rank, DIMM or device).

Table 9–21. ALTMEMPHY Board Settings (Part 2 of 2)

Parameter Name	Units	Description
Max skew between DQS groups	ns	Sets the largest skew between DQS signals in different DQS groups. This value affects the Resynchronization margin for the DDR2 interfaces in both single- or multi-rank configurations.
Addr/command to CK skew	ns	Sets the skew or propagation delay between the CK signal and the address and command signals. The positive values represent the address and command signals that are longer than the CK signals, and the negative values represent the address and command signals that are shorter than the CK signals. This skew is used by the Quartus II software to optimize the delay of the address/command signals to have appropriate setup and hold margins for the DDR2 interfaces.

Controller Settings

This section describes parameters for the High Performance Controller II (HPC II) with advanced features introduced in version 11.0 for designs generated in version 11.0. Designs created in earlier versions and regenerated in version 11.0 do not inherit the new advanced features; for information on parameters for HPC II without the version 11.0 advanced features, refer to the *External Memory Interface Handbook* for Quartus II version 10.1, available in the Literature: External Memory Interfaces page of the Altera website.

Table 9–22 lists the options provided in the **Controller Settings** tab.

Table 9–22. Controller Settings (Part 1 of 2)

Parameter	Description
Controller architecture	Specifies the controller architecture.
Enable self-refresh controls	Turn on to enable the controller to allow you to have control on when to place the external memory device in self-refresh mode, refer to the "User-Controlled Self-Refresh" section in the *Functional Description—HPC II Controller* chapter of the *External Memory Interface Handbook*.
Enable power down controls	Turn on to enable the controller to allow you to have control on when to place the external memory device in power-down mode.
Enable auto power down	Turn on to enable the controller to automatically place the external memory device in power-down mode after a specified number of idle controller clock cycles is observed in the controller. You can specify the number of idle cycles after which the controller powers down the memory in the **Auto Power Down Cycles** field, refer to the "Automatic Power-Down with Programmable Time-Out" section in the *Functional Description—HPC II Controller* chapter of the *External Memory Interface Handbook*.
Auto power down cycles	Determines the desired number of idle controller clock cycles before the controller places the external memory device in a power-down mode. The legal range is 1 to 65,535. The auto power-down mode is disabled if you set the value to 0 clock cycles.
Enable user auto-refresh controls	Turn on to enable the controller to allow you to issue a single refresh.
Enable auto-precharge control	Turn on to enable the auto-precharge control on the controller top level. Asserting the auto-precharge control signal while requesting a read or write burst allows you to specify whether or not the controller should close (auto-precharge) the current opened page at the end of the read or write burst.

Table 9–22. Controller Settings (Part 2 of 2)

Parameter	Description
Enable reordering	Turn on to allow the controller to perform command and data reordering to achieve the highest efficiency.
Starvation limit for each command	Specifies the number of commands that can be served before a waiting command is served. The legal range is from 1 to 63.
Local-to-memory address mapping	Allows you to control the mapping between the address bits on the Avalon interface and the chip, row, bank, and column bits on the memory interface. If your application issues bursts that are greater than the column size of the memory device, choose the Chip-Row-Bank-Column option. This option allows the controller to use its look-ahead bank management feature to hide the effect of changing the currently open row when the burst reaches the end of the column. On the other hand, if your application has several masters that each use separate areas of memory, choose the Chip-Bank-Row-Column option. This option allows you to use the top address bits to allocate a physical bank in the memory to each master. The physical bank allocation avoids different masters accessing the same bank which is likely to cause inefficiency, as the controller must then open and close rows in the same bank.
Command queue look-ahead depth	Specifies a command queue look-ahead depth value to control the number of read or write requests the look-ahead bank management logic examines.
Local maximum burst count	Specifies a burst count to configure the maximum Avalon burst count that the controller slave port accepts.
Reduce controller latency by	Specifies, in controller clock cycles, a value by which to reduce the controller latency. The default value is **0** but you have the option to choose **1** to enhance the latency performance of your design at the expense of timing closure.
Enable configuration and status register interface	Turn on to enable run-time configuration and status retrieval of the memory controller. Enabling this option adds an additional Avalon-MM slave port to the memory controller top level that allows run-time reconfiguration and status retrieving for memory timing parameters, memory address size and mode register settings, and controller features. If the **Error Detection and Correction Logic** option is enabled, the same slave port also allows you to control and retrieve the status of this logic. For more information, refer to the "Configuration and Status Register (CSR) Interface" section in the *Functional Description—HPC II Controller* chapter of the *External Memory Interface Handbook*
Enable error detection and correction logic	Turn on to enable error correction coding (ECC) for single-bit error correction and double-bit error detection.
Enable auto error correction	Turn on to allow the controller to perform auto correction when the ECC logic detects a single-bit error. Alternatively, you can turn off this option and schedule the error correction at a desired time for better system efficiency.
Multiple controller clock sharing	This option is only available in SOPC Builder Flow. Turn on to allow one controller to use the Avalon clock from another controller in the system that has a compatible PLL. This option allows you to create SOPC Builder systems that have two or more memory controllers that are synchronous to your master logic. ☞ This option is not for use with Cyclone III or Cyclone IV family devices.
Local interface protocol	Specifies the local side interface between the user logic and the memory controller. The Avalon-MM interface allows you to easily connect to other Avalon-MM peripherals. The HPC II architecture supports only the Avalon-MM interface.

Parameterizing Memory Controllers with UniPHY IP

This section describes the parameters you can set for the LPDDR2, DDR2, DDR3 SDRAM, QDR II, QDR II+ SRAM, RLDRAM II, and RLDRAM 3 memory controllers with the UniPHY IP.

The **Parameter Settings** page in the UniPHY parameter editor allows you to parameterize the following settings:

- PHY Settings
- Memory Parameters
- Memory Timing
- Board Settings
- Controller Settings
- Diagnostics

The text window at the bottom of the MegaWizard Plug-In Manager displays information about the memory interface, warnings, and errors if you are trying to create something that is not supported. The **Finish** button is disabled until you correct all the errors indicated in this window.

The following sections describe the tabs of the **Parameter Settings** page in more detail.

PHY Settings

Table 9–23 lists the PHY parameters.

Table 9–23. Clock Parameters (Part 1 of 4)

Parameter	Description
General Settings	
Speed Grade	Specifies the speed grade of the targeted FPGA device that affects the generated timing constraints and timing reporting.
Generate PHY only	Turn on this option to generate the UniPHY core without a memory controller. When you turn on this option, the AFI interface is exported so that you can easily connect your own memory controller. Not applicable to RLDRAM 3 UniPHY as no controller support for RLDRAM 3 UniPHY.
Clocks	
Memory clock frequency	The frequency of the clock that drives the memory device. Use up to 4 decimal places of precision. To obtain the maximum supported frequency for your target memory configuration, refer to the External Memory Interface Spec Estimator page on the Altera website.
Achieved memory clock frequency	The actual frequency the PLL generates to drive the external memory interface (memory clock).
PLL reference clock frequency	The frequency of the input clock that feeds the PLL. Use up to 4 decimal places of precision.

Table 9–23. Clock Parameters (Part 2 of 4)

Parameter	Description
Rate on Avalon-MM interface	The width of data bus on the Avalon-MM interface. **Full** results in a width of 2× the memory data width. **Half** results in a width of 4× the memory data width. **Quarter** results in a width of 8× the memory data width. Use **Quarter** for memory frequency 533 MHz and above. To determine the Avalon-MM interface rate selection for other memories, refer to the local interface clock rate for your target device in the External Memory Interface Spec Estimator page on the Altera website.
Achieved local clock frequency	The actual frequency the PLL generates to drive the local interface for the memory controller (AFI clock).
Enable AFI half rate clock	Export the afi_half_rate clock which is running half of the AFI clock rate to the top level.
Advanced PHY Settings	
Advanced clock phase control	Enables access to clock phases. Default value should suffice for most DIMMs and board layouts, but can be modified if necessary to compensate for larger address and command versus clock skews. This option is available for DDR, DDR2 and DDR3 SDRAM only.
Additional address and command clock phase	Allows you to increase or decrease the amount of phase shift on the address and command clock. The base phase shift center aligns the address and command clock at the memory device, which may not be the optimal setting under all circumstances. Increasing or decreasing the amount of phase shift can improve timing. The default value is 0 degrees. In DDR, DDR2, DDR3 SDRAM, and LPDDR2 SDRAM, you can set this value from -360 to 360 degrees. In QDRII/II+ SRAM and RLDRAM II, the available settings are -45, -22.5, 22.5, and 45. To achieve the optimum setting, adjust the value based on the address and command timing analysis results.
Additional phase for core-to-periphery transfer	Allows you to phase shift the latching clock of the core-to-periphery transfers. By delaying the latch clock, a positive phase shift value improves setup timing for transfers between registers in the core and the half-rate `DDIO_OUT` blocks in the periphery, respectively. Adjust this setting according to the core timing analysis. The default value is 0 degrees. You can set this value from -179 to 179 degrees.
Additional CK/CK# phase	Allows you to increase or decrease the amount of phase shift on the CK/CK# clock. The base phase shift center aligns the address and command clock at the memory device, which may not be the optimal setting under all circumstances. Increasing or decreasing the amount of phase shift can improve timing. Increasing or decreasing the phase shift on CK/CK# also impacts the read, write, and leveling transfers, which increasing or decreasing the phase shift on the address and command clocks does not. To achieve the optimum setting, adjust the value based on the address and command timing analysis results. Ensure that the read, write, and write leveling timings are met after adjusting the clock phase. Adjust this value when there is a core timing failure after adjusting **Additional address and command clock phase**. The default value is 0 degrees. You can set this value from -360 to 360 degrees. This option is available for LPDDR2, DDR, DDR2, and DDR3 SDRAM only.
Supply voltage	The supply voltage and sub-family type of memory. This option is available for DDR3 SDRAM only.
I/O standard	The I/O standard voltage. Set the I/O standard according to your design's memory standard.

Table 9–23. Clock Parameters (Part 3 of 4)

Parameter	Description
PLL sharing mode	When you select **No sharing**, the parameter editor instantiates a PLL block without exporting the PLL signals. When you select **Master**, the parameter editor instantiates a PLL block and exports the signals. When you select **Slave**, the parameter editor exposes a PLL interface and you must connect an external PLL master to drive the PLL slave interface signals. Select **No sharing** if you are not sharing PLLs, otherwise select **Master** or **Slave.** For more information about resource sharing, refer to "The DLL and PLL Sharing Interface" section in the *Functional Description—UniPHY* chapter of the *External Memory Interface Handbook*.
Number of PLL sharing interfaces	This option allows you to specify the number of PLL sharing interfaces to create, facilitating creation of many one-to-one connections in Qsys flow. In Megawizard, you can select one sharing interface and manually connect the master to all the slaves. This option is enabled when you set **PLL sharing mode** to **Master**.
DLL sharing mode	When you select **No sharing**, the parameter editor instantiates a DLL block without exporting the DLL signals. When you select **Master**, the parameter editor instantiates a DLL block and exports the signals. When you select **Slave**, the parameter editor exposes a DLL interface and you must connect an external DLL master to drive the DLL slave signals. Select **No sharing** if you are not sharing DLLs, otherwise select **Master** or **Slave.** For more information about resource sharing, refer to "The DLL and PLL Sharing Interface" section in the *Functional Description—UniPHY* chapter of the *External Memory Interface Handbook*.
Number of DLL sharing interfaces	This option allows you to specify the number of DLL sharing interfaces to create, facilitating creation of many one-to-one connections in Qsys flow. In Megawizard, you can select one sharing interface and manually connect the master to all the slaves. This option is enabled when you set **PLL sharing mode** to **Master**.
OCT sharing mode	When you select **No sharing**, the parameter editor instantiates an OCT block without exporting the OCT signals. When you select **Master**, the parameter editor instantiates an OCT block and exports the signals. When you select **Slave**, the parameter editor exposes an OCT interface and you must connect an external OCT control block to drive the OCT slave signals. Select **No sharing** if you are not sharing OCT blocks, otherwise select **Master** or **Slave.** For more information about resource sharing, refer to "The OCT Sharing Interface" section in the *Functional Description—UniPHY* chapter of the *External Memory Interface Handbook*.
Number of OCT sharing interfaces	This option allows you to specify the number of OCT sharing interfaces to create, facilitating creation of many one-to-one connections in Qsys flow. In Megawizard, you can select one sharing interface and manually connect the master to all the slaves. This option is enabled when you set **PLL sharing mode** to **Master**.
HardCopy compatibility	Enables all required HardCopy compatibility options for the generated IP core. For some parameterizations, a pipeline stage is added to the write datapath to help the more challenging timing closure for designs using HardCopy devices; the pipeline stage does not affect the overall read and write latency. Turn on this option if you are migrating your design to a HardCopy device. For more information, refer to the *HardCopy Design Migration Guidelines* chapter.

Table 9–23. Clock Parameters (Part 4 of 4)

Parameter	Description
Reconfigurable PLL location	When you set the PLL used in the UniPHY memory interface to be reconfigurable at run time, you must specify the location of the PLL. This assignment generates a PLL that can only be placed in the given sides. This option is enabled when you turn on **HardCopy compatibility**. In HardCopy designs, you must specify the PLL location according to the location of the interface.
Sequencer optimization	Select **Performance** to enable the Nios II-based sequencer, or **Area** to enable the RTL-based sequencer. Altera recommends that you enable the Nios-based sequencer for memory clock frequencies greater than 400 MHz and enable the RTL-based sequencer if you want to reduce resource utilization. This option is available for QDRII and QDR II+ SRAM, and RLDRAM II only.

Memory Parameters

Use this tab to apply the memory parameters from your memory manufacturer's data sheet.

LPDDR2, DDR2 and DDR3 SDRAM

Table 9–24 lists the memory parameters for LPDDR2, DDR2 and DDR3 SDRAM.

Table 9–24. Memory Parameters (Part 1 of 4)

Parameter	Description
Memory vendor	The vendor of the memory device. Select the memory vendor according to the memory vendor you use. For memory vendors that are not listed in the setting, select JEDEC with the nearest memory parameters and edit the parameter values according to the values of the memory vendor that you use. However, if you select a configuration from the list of memory presets, the default memory vendor for that preset setting is automatically selected.
Memory format	The format of the memory device. Select **Discrete** if you are using just the memory device. Select **Unbuffered** or **Registered** for DIMM format. Use the DIMM format to turn on levelling circuitry for LPDDR2 support device only. DDR2 supports DIMM also.
Memory device speed grade	The maximum frequency at which the memory device can run.
Total interface width	The total number of DQ pins of the memory device. Limited to 144 bits for DDR2 and DDR3 SDRAM (with or without leveling). The total interface is depending on the rate on Avalon-MM interface because the maximum Avalon data width is 1024. If you select 144 bit for total interface width with Quarter-rate, the avalon data width is 1152 exceeding maximum avalon data width.
DQ/DQS group size	The number of DQ bits per DQS group.
Number of DQS groups	The number of DQS groups is calculated automatically from the Total interface width and the DQ/DQS group size parameters.
Number of chip selects (DDR2 and DDR3 SDRAM device only)	The number of chip-selects the IP core uses for the current device configuration. Specify the total number of chip-selects according to the number of memory device.

Table 9–24. Memory Parameters (Part 2 of 4)

Parameter		Description
Depth expansion (LPDDR2 SDRAM)		Specifies number of devices are expanded in depth. Only single chip select is supported.
Number of clocks		The width of the clock bus on the memory interface.
Row address width		The width of the row address on the memory interface.
Column address width		The width of the column address on the memory interface.
Bank-address width		The width of the bank address bus on the memory interface.
Enable DM pins		Specifies whether the DM pins of the memory device are driven by the FPGA. You can turn off this option to avoid overusing FPGA device pins when using x4 mode memory devices. When you are using x4 mode memory devices, turn off this option for DDR3 SDRAM. You must turn on this option if you are using Avalon byte enable.
DQS# Enable (DDR2)		Turn on differential DQS signaling to improve signal integrity and system performance. This option is available for DDR2 SDRAM only.
Memory Initialization Options—DDR2		
Address and command parity		Enables address/command parity checking. This is required for Registered DIMM.
Mode Register 0	**Burst length**	Specifies the burst length.
	Read burst type	Specifies accesses within a given burst in sequential or interleaved order. Specify sequential ordering for use with the Altera memory controller. Specify interleaved ordering only for use with an interleaved-capable custom controller, when the **Generate PHY only** parameter is enabled on the PHY Settings tab.
	DLL precharge power down	Determines whether the DLL in the memory device is in slow exit mode or in fast exit mode during precharge power down. For more information, refer to memory vendor data sheet.
	Memory CAS latency setting	Determines the number of clock cycles between the READ command and the availability of the first bit of output data at the memory device. For more information, refer to memory vendor data sheet speed bin table. Set this parameter according to the target memory speed grade and memory clock frequency.
Mode Register 1	**Output drive strength setting**	Determines the output driver impedance setting at the memory device. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
	Memory additive CAS latency setting	Determines the posted CAS additive latency of the memory device. Enable this feature to improve command and bus efficiency, and increase system bandwidth. For more information, refer to the *Optimizing the Controller* chapter.
	Memory on-die termination (ODT) setting	Determines the on-die termination resistance at the memory device. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
Mode Register 2	**SRT Enable**	Determines the selfrefresh temperature (SRT). Select **1x refresh rate** for normal temperature (0-85C)or select **2x refresh rate** for high-temperature (>85C).

Table 9–24. Memory Parameters (Part 3 of 4)

Parameter		Description
Memory Initialization Options—DDR3		
Mirror Addressing: 1 per chip select		Specifies the mirror addressing for multiple rank DIMMs. Refer to memory vendor data sheet for more information. Enter ranks with mirrored addresses in this field. For example, for four chip selects, enter 1101 to mirror the address on chip select #3, #2, and #0.
Address and command parity		Enables address/command parity checking to detect errors in data transmission. This is required for registered DIMM (RDIMM).
Mode Register 0	**Read burst type**	Specifies accesses within a given burst in sequential or interleaved order. Specify sequential ordering for use with the Altera memory controller. Specify interleaved ordering only for use with an interleaved-capable custom controller, when the **Generate PHY only** parameter is enabled on the PHY Settings tab.
	DLL precharge power down	Specifies whether the DLL in the memory device is off or on during precharge power-down.
	Memory CAS latency setting	The number of clock cycles between the read command and the availability of the first bit of output data at the memory device and also interface frequency. Refer to memory vendor data sheet speed bin table. Set this parameter according to the target memory speed grade and memory clock frequency.
Mode Register 1	**Output drive strength setting**	The output driver impedance setting at the memory device. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
	Memory additive CAS latency setting	The posted CAS additive latency of the memory device. Enable this feature to improve command and bus efficiency, and increase system bandwidth. For more information, refer to the *Optimizing the Controller* chapter.
	ODT Rtt nominal value	The on-die termination resistance at the memory device. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
Mode Register 2	**Auto selfrefresh method**	Disable or enable auto selfrefresh.
	Selfrefresh temperature	Specifies the selfrefresh temperature as **Normal** or **Extended**.
	Memory write CAS latency setting	The number of clock cycles from the releasing of the internal write to the latching of the first data in, at the memory device and also interface frequency. Refer to memory vendor data sheet speed bin table and set according to the target memory speed grade and memory clock frequency.
	Dynamic ODT (Rtt_WR) value	The mode of the dynamic ODT feature of the memory device. This is used for multi-rank configurations. Refer to *DDR2 and DDR3 SDRAM Board Layout Guidelines.* To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
Memory Initialization Options— LPDDR2		
Mode Register 1	**Burst Length**	Specifies the burst length.
	Read Burst Type	Specifies accesses within a given burst in sequential or interleaved order. Specify sequential ordering for use with the Altera memory controller. Specify interleaved ordering only for use with an interleaved-capable custom controller, when the **Generate PHY only** parameter is enabled on the PHY Settings tab.

Table 9–24. Memory Parameters (Part 4 of 4)

Parameter		Description
Mode Register 2	**Memory CAS latency setting**	Determines the number of clock cycles between the READ command and the availability of the first bit of output data at the memory device. Set this parameter according to the target memory interface frequency. Refer to memory data sheet and also target memory speed grade.
Mode Register 3	**Output drive strength settings**	Determines the output driver impedance setting at the memory device. To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.

QDR II and QDR II+ SRAM

Table 9–25 describes the memory parameters for QDR II and QDR II+ SRAM.

Table 9–25. Memory Parameters

Parameter	Description
Address width	The width of the address bus on the memory device.
Data width	The width of the data bus on the memory device.
Data-mask width	The width of the data-mask on the memory device,
CQ width	The width of the CQ (read strobe) bus on the memory device.
K width	The width of the K (write strobe) bus on the memory device.
Burst length	The burst length supported by the memory device. For more information, refer to memory vendor data sheet.
Topology	
x36 emulated mode	Emulates a larger memory-width interface using smaller memory-width interfaces on the FPGA. Turn on this option when the target FPGA do not support x36 DQ/DQS group. This option allows two x18 DQ/DQS groups to emulate 1 x36 read data group.
Emulated write groups	Number of write groups to use to form the x36 memory interface on the FPGA. Select **2** to use 2 x18 DQ/DQS group to form x36 write data group. Select **4** to use 4 x9 DQ/DQS group to form x36 write data group.
Device width	Specifies the number of memory devices used for width expansion.

RLDRAM II

Table 9–25 describes the memory parameters for RLDRAM II.

Table 9–26. Memory Parameters (Part 1 of 2)

Parameter	Description
Address width	The width of the address bus on the memory device.
Data width	The width of the data bus on the memory device.
Bank-address width	The width of the bank-address bus on the memory device.

Table 9–26. Memory Parameters (Part 2 of 2)

Parameter	Description
Data-mask width	The width of the data-mask on the memory device,
QK width	The width of the QK (read strobe) bus on the memory device. Select **1** when data width is set to 9. Select **2** when data width is set to 18 or 36.
DK width	The width of the DK (write strobe) bus on the memory device. Select **1** when data width is set to 9 or 18. Select **2** when data width is set to 36.
Burst length	The burst length supported by the memory device. For more information, refer to memory vendor data sheet.
Memory mode register configuration	Configuration bits that set the memory mode. Select the option according to the interface frequency.
Device impedance	Select External (ZQ) to adjust the driver impedance using the external impedance resistor (RQ). The output impedance range is 25-60Ω You must connect the RQ resistor between ZQ pin and ground. The value of RQ must be 5 times the output impedance. For example, 60Ω output impedance requires 300Ω RQ. Set the value according to the board simulation.
On-Die Termination	Turn on this option to enable ODT in the memory to terminate the DQs and DM pins to Vtt. Dynamically switch off during read operation and switch on during write operation. Refer to memory vendor data sheet for more information.
Topology	
Device width	Specifies the number of memory devices used for width expansion.

Memory Timing

Use this tab to apply the memory timings from your memory manufacturer's data sheet. Table 9–27 lists the memory timing parameters.

Table 9–27. Memory Timing Parameters (Part 1 of 3)

Parameter	Description
DDR2/DDR3 SDRAM	
tIS (base)	Address and control setup to CK clock rise. Set according to the memory speed grade and refer to the memory vendor data sheet.
tIH (base)	Address and control hold after CK clock rise. Set according to the memory speed grade and refer to the memory vendor data sheet.
tDS (base)	Data setup to clock (DQS) rise. Set according to the memory speed grade and refer to the memory vendor data sheet.
tDH (base)	Data hold after clock (DQS) rise. Set according to the memory speed grade and refer to the memory vendor data sheet.
tDQSQ	DQS, DQS# to DQ skew, per access. Set according to the memory speed grade and refer to the memory vendor data sheet.
tQHS (DDR2) **tQH (DDR3)**	DQ output hold time from DQS, DQS# (absolute time value) DQ output hold time from DQS, DQS# (percentage of tCK). Set according to the memory speed grade and refer to the memory vendor data sheet.
tDQSCK	DQS output access time from CK/CK#. Set according to the memory speed grade and refer to the memory vendor data sheet.
tDQSS	First latching edge of DQS to associated clock edge (percentage of tCK). Set according to the memory speed grade and refer to the memory vendor data sheet.
tQSH (DDR3) **tDQSH (DDR2)**	DQS Differential High Pulse Width (percentage of tCK). Specifies the minimum high time of the DQS signal received by the memory. Set according to the memory speed grade and refer to the memory vendor data sheet.
tDSH	DQS falling edge hold time from CK (percentage of tCK). Set according to the memory speed grade and refer to the memory vendor data sheet.
tDSS	DQS falling edge to CK setup time (percentage of tCK). Set according to the memory speed grade and refer to the memory vendor data sheet.
tINIT	Memory initialization time at power-up. Set according to the memory speed grade and refer to the memory vendor data sheet.
tMRD	Load mode register command period. Set according to the memory speed grade and refer to the memory vendor data sheet.
tRAS	Active to precharge time. Set according to the memory speed grade and refer to the memory vendor data sheet.
tRCD	Active to read or write time. Set according to the memory speed grade and refer to the memory vendor data sheet.
tRP	Precharge command period. Set according to the memory speed grade and refer to the memory vendor data sheet.
tREFI	Refresh command interval. Set according to the memory speed grade and refer to the memory vendor data sheet.
tRFC	Auto-refresh command interval. Set according to the memory speed grade and refer to the memory vendor data sheet.

Table 9–27. Memory Timing Parameters (Part 2 of 3)

Parameter	Description
tWR	Write recovery time. Set according to the memory speed grade and refer to the memory vendor data sheet.
tWTR	Write to read period. Set according to the memory speed grade and memory clock frequency. Refer to the memory vendor data sheet.
tFAW	Four active window time. Set according to the memory speed grade and page size. Refer to the memory vendor data sheet.
tRRD	RAS to RAS delay time. Set according to the memory speed grade, page side and memory clock rate. Refer to the memory vendor data sheet.
tRTP	Read to precharge time. Set according to memory speed grade. Refer to the memory vendor data sheet. Calculate the value based on the memory interface frequency and memory clock frequency.
QDR II and QDR II+ SRAM	
tWL (cycles)	The write latency. Set write latency 0 for burst length of 2, and set write latency to 1 for burst length of 4.
tRL (cycles)	The read latency. Set according to memory protocol. Refer to memory data sheet.
tSA	The address and control setup to K clock rise. Set according to memory protocol. Refer to memory data sheet.
tHA	The address and control hold after K clock rise. Set according to memory protocol. Refer to memory data sheet.
tSD	The data setup to clock (K/K#) rise. Set according to memory protocol. Refer to memory data sheet.
tHD	The data hold after clock (K/K#) rise. Set according to memory protocol. Refer to memory data sheet.
tCQD	Echo clock high to data valid. Set according to memory protocol. Refer to memory data sheet.
tCQDOH	Echo clock high to data invalid. Set according to memory protocol. Refer to memory data sheet.
Internal jitter	The QDRII/II+ internal jitter. Refer to memory data sheet.
TCQHCQnH	The CQ clock rise to CQ*n* clock rise (rising edge to rising edge). Set according to memory speed grade. Refer to memory data sheet.
TKHKnH	The K clock rise to Kn clock rise (rising edge to rising edge). Set according to memory speed grade. Refer to memory data sheet.
RLDRAM II	
Maximum memory clock frequency	The maximum frequency at which the memory device can run. Set according to memory speed grade. Refer to memory data sheet.
Refresh interval	The refresh interval. Set according to memory speed grade. Refer to memory data sheet.
tCKH (%)	The input clock (CK/CK#) high expressed as a percentage of the full clock period. Set according to memory speed grade. Refer to memory data sheet.
tQKH (%)	The read clock (QK/QK#) high expressed as a percentage of tCKH. Set according to memory speed grade. Refer to memory data sheet.

Table 9–27. Memory Timing Parameters (Part 3 of 3)

Parameter	Description
tAS	Address and control setup to CK clock rise. Set according to memory speed grade. Refer to memory data sheet.
tAH	Address and control hold after CK clock rise. Set according to memory speed grade. Refer to memory data sheet.
tDS	Data setup to clock (CK/CK#) rise. Set according to memory speed grade. Refer to memory data sheet.
tDH	Data hold after clock (CK/CK#) rise. Set according to memory speed grade. Refer to memory data sheet.
tQKQ_max	QK clock edge to DQ data edge (in same group). Set according to memory speed grade. Refer to memory data sheet.
tQKQ_min	QK clock edge to DQ data edge (in same group). Set according to memory speed grade. Refer to memory data sheet.
tCKDK_max	Clock to input data clock (max). Set according to memory speed grade. Refer to memory data sheet.
tCKDK_min	Clock to input data clock (min). Set according to memory speed grade. Refer to memory data sheet.

Table 9–28. RLDRAM 3 Memory Parameters

Parameter	Description
Enable data-mask pins	Specifies whether the DM pins of the memory device are driven by the FPGA.
Data-mask width	The width of the data-mask on the memory device.
Data width	The width of the data bus on the memory device.
QK width	The width of the QK (read strobe) bus on the memory device. Select 2 when data width is set to 18. Select 4 when data width is set to 36.
DK width	The width of the DK (write strobe) bus on the memory device. For x36 device, DQ[8:0] and DQ[26:18] are referenced to DK0/DK0#, and DQ[17:9] and DQ[35:27] are referenced to DK1/DK1#.
Address width	The width of the address bus on the memory device.
Bank-address width	The width of the bank-address bus on the memory device.
Burst length	The burst length supported by the memory device. Refer to memory vendor data sheet.
tRC	Mode register bits that set the tRC. Set the tRC according to the memory speed grade and data latency. Refer to the tRC table in the memory vendor data sheet.
Data Letency	Mode register bits that set the latency. Set latency according to the interface frequency and memory speed grade. Refer to speed bin table in the memory data sheet.
Output Drive	Mode register bits that set the output drive impedance setting. Set the value according to the board simulation.
ODT	Mode register bits that set the ODT setting. Set the value according to the board simulation.
AREF Protocol	Mode register setting for refreshing memory content of a bank. Select **Multibank** to allow refresh 4 bank simultaneously. Select Bank Address Control to refresh a particular bank by setting the bank address.
Write Protocol	Mode register setting for write protocol. When multiple bank (dual bank or quad bank) is selected, identical data is written to multiple banks.
Topology	
Device width	Specifies the number of memory devices used for width expansion.

Table 9–29. RLDRAM 3 Memory Timing (Part 1 of 2)

Parameter	Description
Memory Device Timing	
Maximum memory clock frequency	The maximum frequency at which the memory device can run.
tDS (base)	Base specification for data setup to DK/DK#. Set according to memory speed grade. Refer to memory data sheet.

Table 9–29. RLDRAM 3 Memory Timing (Part 2 of 2)

Parameter	Description
tDH (base)	Base specification for data hold from DK/DK#. Set according to memory speed grade. Refer to memory data sheet.
tQKQ_max	QK/QK# clock edge to DQ data edge (in same group). Set according to memory speed grade. Refer to memory data sheet.
tQH (% of CK)	DQ output hold time from QK/QK#. Set according to memory speed grade. Refer to memory data sheet.
tCKDK_max(% of CK)	Clock to input data clock (max). Set according to memory speed grade. Refer to memory data sheet.
tCKDK_min (% of CK)	Clock to input data clock (min). Set according to memory speed grade. Refer to memory data sheet.
tCKQK_max	QK edge to clock edge skew (max). Set according to memory speed grade. Refer to memory data sheet.
tIS (base)	Base specification for address and control setup to CK.Set according to memory speed grade. Refer to memory data sheet.
tIH (base)	Base specification for address and control hold from CK. Set according to memory speed grade. Refer to memory data sheet.
Controller Timing	
Read-to-Write NOP commands (min)	Minimum number of no operation commands following a read command and before a write command. The value must be at least ((Burst Length/2) + RL - WL + 2). The value, along with other delay/skew parameters, are used by the "Bus Turnaround" timing analysis to determine if bus contention is an issue. Set according to the controller specification.
Write-to-Read NOP commands (min)	Minimum number of no operation commands following a write command and before a read command. The value must be at least ((Burst Length/2) + WL - RL + 1). The value, along with other delay/skew parameters, are used by the "Bus Turnaround" timing analysis to determine if bus contention is an issue. Set according to the controller specification.
RLDRAM 3 Board Derate	
CK/CK# slew rate (differential)	CK/CK# slew rate (differential).
Address/Command slew rate	Address and command slew rate.
DK/DK# slew rate (Differential)	DK/DK# slew rate (differential).
DQ slew rate	DQ slew rate.
tIS	Address/command setup time to CK.

Board Settings

Use the **Board Settings** tab to model the board-level effects in the timing analysis. The **Board Settings** tab allows you to set the following settings:

- Setup and hold derating (Available for LPDDR2/DDR2/DDR3 SDRAM, RLDRAM 3 and RLDRAM II only)
- Intersymbol interference
- Board skews

For accurate timing results, you must enter board settings parameters that are correct for your PCB.

The IP core supports single and multiple chip-select configurations. Altera has determined the effects on the output signalling of these configurations for certain Altera boards, and has stored the effects on the output slew rate and the intersymbol interference (ISI) within the wizard.

These stored values are representative of specific Altera boards. You must change the values to account for the board-level effects for your board. You can use HyperLynx or similar simulators to obtain values that are representative of your board.

For more information about how to include your board simulation results in the Quartus II software and how to assign pins using pin planners, refer to the design flow tutorials and design examples on the List of designs using Altera External Memory IP page of the Altera Wiki website

For information about timing deration methodology, refer to the "Timing Deration Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs" section in the *Analyzing Timing of Memory IP* chapter.

Setup and Hold Derating

The slew rate of the output signals affects the setup and hold times of the memory device. You can specify the slew rate of the output signals to see their effect on the setup and hold times of both the address and command signals and the DQ signals, or specify the setup and hold times directly.

You should enter information derived during your PCB development process of prelayout (line) and postlayout (board) simulation.

Table 9–30 lists the setup and hold derating parameters.

Table 9–30. Setup and Hold Derating Parameters (Part 1 of 2)

Parameter	Description
LPDDR2/DDR2/DDR3 SDRAM/RLDRAM 3	
Derating method	Derating method. The default settings are based on Altera internal board simulation data. To obtain accurate timing analysis according to the condition of your board, Altera recommends that you perform board simulation and enter the slew rate in the Quartus II software to calculate the derated setup and hold time automatically or enter the derated setup and hold time directly. For more information, refer to the "Timing Deration Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs" section in the *Analyzing Timing of Memory IP* chapter.
CK/CK# slew rate (differential)	CK/CK# slew rate (differential).
Address/Command slew rate	Address and command slew rate.
DQS/DQS# slew rate (Differential)	DQS and DQS# slew rate (differential).
DQ slew rate	DQ slew rate.
tIS	Address/command setup time to CK.
tIH	Address/command hold time from CK.
tDS	Data setup time to DQS.
tDH	Data hold time from DQS.
RLDRAM II	
tAS Vref to CK/CK# Crossing	For a given address/command and CK/CK# slew rate, the memory device data sheet provides a corresponding "tAS Vref to CK/CK# Crossing" value that can be used to determine the derated address/command setup time.
tAS VIH MIN to CK/CK# Crossing	For a given address/command and CK/CK# slew rate, the memory device data sheet provides a corresponding "tAS VIH MIN to CK/CK# Crossing" value that can be used to determine the derated address/command setup time.
tAH CK/CK# Crossing to Vref	For a given address/command and CK/CK# slew rate, the memory device data sheet provides a corresponding "tAH CK/CK# Crossing to Vref" value that can be used to determine the derated address/command hold time.
tAH CK/CK# Crossing to VIH MIN	For a given address/command and CK/CK# slew rate, the memory device data sheet provides a corresponding "tAH CK/CK# Crossing to VIH MIN" value that can be used to determine the derated address/command hold time.

Table 9–30. Setup and Hold Derating Parameters (Part 2 of 2)

Parameter	Description
tDS Vref to CK/CK# Crossing	For a given data and DK/DK# slew rate, the memory device data sheet provides a corresponding "tDS Vref to CK/CK# Crossing" value that can be used to determine the derated data setup time.
tDS VIH MIN to CK/CK# Crossing	For a given data and DK/DK# slew rate, the memory device data sheet provides a corresponding "tDS VIH MIN to CK/CK# Crossing" value that can be used to determine the derated data setup time.
tDH CK/CK# Crossing to Vref	For a given data and DK/DK# slew rate, the memory device data sheet provides a corresponding "tDH CK/CK# Crossing to Vref" value that can be used to determine the derated data hold time.
tDH CK/CK# Crossing to VIH MIN	For a given data and DK/DK# slew rate, the memory device data sheet provides a corresponding "tDH CK/CK# Crossing to VIH MIN" value that can be used to determine the derated data hold time.
Derated tAS	The derated address/command setup time is calculated automatically from the "tAS", the "tAS Vref to CK/CK# Crossing", and the "tAS VIH MIN to CK/CK# Crossing" parameters.
Derated tAH	The derated address/command hold time is calculated automatically from the "tAH", the "tAH CK/CK# Crossing to Vref", and the "tAH CK/CK# Crossing to VIH MIN" parameters.
Derated tDS	The derated data setup time is calculated automatically from the "tDS", the "tDS Vref to CK/CK# Crossing", and the "tDS VIH MIN to CK/CK# Crossing" parameters.
Derated tDH	The derated data hold time is calculated automatically from the "tDH", the "tDH CK/CK# Crossing to Vref", and the "tDH CK/CK# Crossing to VIH MIN" parameters.

Intersymbol Interference

Intersymbol interference is the distortion of a signal in which one symbol interferes with subsequent symbols. Typically, when going from a single chip-select configuration to a multiple chip-select configuration there is an increase in intersymbol interference because there are multiple stubs causing reflections. Table 9–31 lists the intersymbol interference parameters.

Table 9–31. ISI Parameters

Parameter	Description
Derating method	Choose between default Altera settings (with specific Altera boards) or manually enter board simulation numbers obtained for your specific board. This option is supported in LPDDR2/DDR2/DDR3 SDRAM only.
Address and command eye reduction (setup)	The reduction in the eye diagram on the setup side (or left side of the eye) due to ISI on the address and command signals compared to a case when there is no ISI. (For single rank designs, ISI can be zero; in multirank designs, ISI is necessary for accurate timing analysis.) For more information about how to measure the ISI value for the address and command signals, refer to the "Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time" section in the *Analyzing Timing of Memory IP* chapter.
Address and command eye reduction (hold)	The reduction in the eye diagram on the hold side (or right side of the eye) due to ISI on the address and command signals compared to a case when there is no ISI. For more information about how to measure the ISI value for the address and command signals, refer to "Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time" section in the *Analyzing Timing of Memory IP* chapter.
DQ/ D eye reduction	The total reduction in the eye diagram due to ISI on DQ signals compared to a case when there is no ISI. Altera assumes that the ISI reduces the eye width symmetrically on the left and right side of the eye. For more information about how to measure the ISI value for the address and command signals, refer to "Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time" section in the *Analyzing Timing of Memory IP* chapter.
Delta DQS/Delta K/ Delta DK arrival time	The increase in variation on the range of arrival times of DQS compared to a case when there is no ISI. Altera assumes that the ISI causes DQS to further vary symmetrically to the left and to the right. For more information about how to measure the ISI value for the address and command signals, refer to "Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time" section in the *Analyzing Timing of Memory IP* chapter.

Board Skews

PCB traces can have skews between them that can reduce timing margins. Furthermore, skews between different chip selects can further reduce the timing margin in multiple chip-select topologies. The **Board Skews** section of the parameter editor allows you to enter parameters to compensate for these variations.

You must ensure the timing margin reported in TimeQuest Report DDR is positive when the board skew parameters are correct for the PCB.

Table 9–32 lists the board skew parameters. For parameter equations containing delay values, delays should be measured as follows:

- Non-fly-by topology (Balanced Clock Tree)
  - For discrete devices–all the delay (CK, Addr/Cmd, DQ and DQS) from the FPGA are right to every memory device
  - For UDIMMs–all the delay (CK, Addr/Cmd, DQ and DQS) from the FPGA to UDIMM connector for every memory device on the UDIMM.
  - For RDIMMs–the Addr/Cmd and CK delay are from the FPGA to the register on the RDIMM. The DQ and DQS delay are from FPGA to RDIMM connector for every memory device on the RDIMM.
  - For LRDIMMS–the delay from the FPGA to the register on the LRDIMM.
- Fly-by topology
  - For discrete devices–the Addr/Cmd and CK delay are from the FPGA to the first memory device. The DQ and DQS delay are from FPGA to every memory device.
  - For UDIMMs–the Addr/Cmd and CK delay are from the FPGA to the UDIMM connector for the first memory device on the UDIMM. The DQ and DQS delay are from the FPGA to UDIMM connector for every memory device on the UDIMM.
  - For RDIMMs–the Addr/Cmd and CK delay are from the FPGA to the register on the RDIMM. The DQ and DQS delay are from FPGA to RDIMM connector for every memory device on the RDIMM.
  - For LRDIMMS–the delay from the FPGA to the register on the LRDIMM.

Equations apply to any given memory device, except when marked by the board or group qualifiers (_*b* or _*g*), where they apply to the particular device or group being iterated over.

Use board skew parameter calculator below to help you to calculate the board skews, refer to Board skew parameter tool.

Table 9–32. Board Skew Parameters (Part 1 of 8)

Parameter	Description
LPDDR2/DDR2/DDR3 SDRAM	
DQ/DQS Package Deskew	Enable this parameter if you will deskew the FPGA package with your board traces on the DQ and DQS pins. This option increases the read capture and write margins. Enable this option when memory clock frequency is larger than 533MHz. Enabling this option improves the read capture and write timing margin. You can also rely on the read capture and write timing margin in timing report to enable this option. When this option is enabled, package skews are output on the DQ and DQS pins in the Pin-Out File (**.pin**) and package skew is not included in timing analysis. All of the other board delay and skew parameters related to DQ or DQS must consider the package and the board together. For more information, refer to *DDR2 and DDR3 Board Layout Guidelines*.

Table 9–32. Board Skew Parameters (Part 2 of 8)

Parameter	Description
Address/Command Package Deskew	Enable this parameter if you will deskew the FPGA package with your board traces on the address and command pins. This option increases the address and command margins. Enable this option when memory clock frequency is larger than 533MHz.Enabling this option will improve the address and command timing margin. You can also rely on the address and command margin in timing report to enable this option. When this option is enabled, package skews are output on the address and command pins in the Pin-Out File (**.pin**) and package skew is not included in timing analysis. All of the other board delay and skew parameters related to Address and Command must consider the package and the board together. For more information, refer to *DDR2 and DDR3 Board Layout Guidelines*.
Maximum CK delay to DIMM/device	The delay of the longest CK trace from the FPGA to the memory device, whether on a DIMM or the same PCB as the FPGA is expressed by the following equation: $\max_r[\max_n(CK_{n_r}PathDelay)\ ]$ Where *n* is the number of memory clock and *r* is number rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clocks in each rank DIMM, the maximum CK delay is expressed by the following equation: $\mathrm{nax}(CK_1\mathrm{PathDelay\ rank\ 1},\ CK_2\mathrm{Path\ Delay\ rank\ 1},\ CK_1\mathrm{Path\ Delay\ rank\ 2},\ CK_2\mathrm{Path\ Delay\ rank\ 2})$
Maximum DQS delay to DIMM/device	The delay of the longest DQS trace from the FPGA to the memory device, whether on a DIMM or the same PCB as the FPGA is expressed by the following equation: $\max_r[\max_n(\mathrm{DQS}_{n_r}\mathrm{Path\ Delay})]$ Where *n* is the number of DQS and *r* is number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 DQS in each rank DIMM, the maximum DQS delay is expressed by the following equation: $\mathrm{nax}(\mathrm{DQS}_1\mathrm{PathDelay\ rank\ 1},\ \mathrm{DQS}_2\mathrm{Path\ Delay\ rank\ 1},\ \mathrm{DQS}_1\mathrm{Path\ Delay\ rank\ 2},\ \mathrm{DQS}_2\mathrm{Path\ Delay\ rank\ 2})$

Table 9–32. Board Skew Parameters (Part 3 of 8)

Parameter	Description
Minimum delay difference between CK and DQS	The minimum skew or smallest positive skew (or largest negative skew) between the CK signal and any DQS signal when arriving at the same DIMM/device over all DIMMs/devices is expressed by the following equation: $\min_r \left[\min_{n,\ m} \{(\mathrm{CK}_{n_r}\mathrm{Delay} - \mathrm{DQS}_{m_r}\mathrm{Delay})\} \right]$ Where *n* is the number of memory clock, *m* is the number of DQS, and *r* is the number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clock and 4 DQS signals (two for each clock) for each rank DIMM, the minimum delay difference between CK and DQS is expressed by the following equation: $\min \left\{ \begin{array}{l} (\mathrm{CK}_{1_1}\mathrm{Delay} - \mathrm{DQS}_{1_1}\mathrm{Delay}),\ (\mathrm{CK}_{1_1}\mathrm{Delay} - \mathrm{DQS}_{2_1}\mathrm{Delay}),\ (\mathrm{CK}_{2_1}\mathrm{Delay} - \mathrm{DQS}_{3_1}\mathrm{Delay}),\ (\mathrm{CK}_{2_1}\mathrm{Delay} - \mathrm{DQS}_{4_1}\mathrm{Delay}) \\ (\mathrm{CK}_{1_2}\mathrm{Delay} - \mathrm{DQS}_{1_2}\mathrm{Delay}),\ (\mathrm{CK}_{1_2}\mathrm{Delay} - \mathrm{DQS}_{2_2}\mathrm{Delay}),\ (\mathrm{CK}_{2_2}\mathrm{Delay} - \mathrm{DQS}_{3_2}\mathrm{Delay}),\ (\mathrm{CK}_{2_1}\mathrm{Delay} - \mathrm{DQS}_{4_2}\mathrm{Delay}) \end{array} \right\}$ This parameter value affects the write leveling margin for DDR3 interfaces with leveling in multi-rank configurations. This parameter value also applies to non-leveling configurations of any number of ranks with the requirement that DQS must have positive margins in Timequest Report DDR. For multiple boards, the minimum skew between the CK signal and any DQS signal when arriving at the same DIMM over all DIMMs is expressed by the following equation, if you want to use the same design for several different boards: $\underset{b}{\overset{boards}{Min}} [\underset{g}{\overset{groups}{Min}} [CK_{g_b} - DQS_{g_b}]]$
Maximum delay difference between CK and DQS	The maximum skew or smallest negative skew (or largest positive skew) between the CK signal and any DQS signal when arriving at the same DIMM/device over all DIMMs/devices is expressed by the following equation: $\max_r \left[\max_{n,m} \{(\mathrm{CK}_{n_r}\mathrm{Delay} - \mathrm{DQS}_{m_r}\mathrm{Delay})\} \right]$ Where *n* is the number of memory clock, *m* is the number of DQS, and *r* is the number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clock and 4 DQS signals (two for each clock) for each rank DIMM, the maximum delay difference between CK and DQS is expressed by the following equation: $\max \left\{ \begin{array}{l} (\mathrm{CK}_{1_1}\mathrm{Delay} - \mathrm{DQS}_{1_1}\mathrm{Delay}),\ (\mathrm{CK}_{1_1}\mathrm{Delay} - \mathrm{DQS}_{2_1}\mathrm{Delay}),\ (\mathrm{CK}_{2_1}\mathrm{Delay} - \mathrm{DQS}_{3_1}\mathrm{Delay}),\ (\mathrm{CK}_{2_1}\mathrm{Delay} - \mathrm{DQS}_{4_1}\mathrm{Delay}) \\ (\mathrm{CK}_{1_2}\mathrm{Delay} - \mathrm{DQS}_{1_2}\mathrm{Delay}),\ (\mathrm{CK}_{1_2}\mathrm{Delay} - \mathrm{DQS}_{2_2}\mathrm{Delay}),\ (\mathrm{CK}_{2_2}\mathrm{Delay} - \mathrm{DQS}_{3_2}\mathrm{Delay}),\ (\mathrm{CK}_{2_1}\mathrm{Delay} - \mathrm{DQS}_{4_2}\mathrm{Delay}) \end{array} \right\}$ This value affects the write Leveling margin for DDR3 interfaces with leveling in multi-rank configurations. This parameter value also applies to non-leveling configurations of any number of ranks with the requirement that DQS must have positive margins in Timequest Report DDR. For multiple boards, the maximum skew (or largest positive skew) between the CK signal and any DQS signal when arriving at the same DIMM over all DIMMs is expressed by the following equation, if you want to use the same design for several different boards: $\underset{b}{\overset{boards}{Max}} [\underset{g}{\overset{groups}{Max}} [CK_{g_b} - DQS_{g_b}]]$

Table 9–32. Board Skew Parameters (Part 4 of 8)

Parameter	Description
Maximum skew within DQS group	The largest skew among DQ and DM signals in a DQS group. This value affects the read capture and write margins for DDR2 and DDR3 SDRAM interfaces in all configurations (single or multiple chip-select, DIMM or component). For multiple boards, the largest skew between DQ and DM signals in a DQS group is expressed by the following equation: $\underset{b}{\overset{boards}{Max}}\,[\,\underset{g}{\overset{groups}{Max}}\,[maxDQ_{g_b} - minDQ_{g_b}]]$
Maximum skew between DQS groups	The largest skew between DQS signals in different DQS groups. This value affects the resynchronization margin in memory interfaces without leveling such as DDR2 SDRAM and discrete-device DDR3 SDRAM in both single- or multiple chip-select configurations. For multiple boards, the largest skew between DQS signals in different DQS groups is expressed by the following equation, if you want to use the same design for several different boards: $\underset{b}{\overset{boards}{Max}}\left[\underset{g}{\overset{groups}{Max}}\,[DQS_{g_b}] - \underset{b}{\overset{boards}{Min}}\left[\underset{g}{\overset{groups}{Min}}\,[DQS_{g_b}]\right]\right]$
Average delay difference between DQ and DQS	The average delay difference between each DQ signal and the DQS signal, calculated by averaging the longest and smallest DQ signal delay values minus the delay of DQS. The average delay difference between DQ and DQS is expressed by the following equation: $\frac{\sum_{n=1}^{n=n}\left[\left(\frac{\textit{Longest DQ Path Delay in DQS}_n \text{ group} + \textit{Shortest DQ Path Delay in DQS}_n \text{ group}}{2}\right) - \textit{DQS}_n\textit{PathDelay}\right]}{n}$ where n is the number of DQS groups. For multi-rank or multiple CS configuration, the equation is: $\frac{\Sigma_{r=1}^{r=r}\left[\text{Average delay differnt between DQ and DQS in rank r}\right]}{r}$
Maximum skew within address and command bus	The largest skew between the address and command signals for a single board is expressed by the following equation: $0.5[(MaxACdelay - MinCKdelay) - (MinACdelay - MaxCKdelay)]$ For multiple boards, the largest skew between the address and command signals is expressed by the following equation, if you want to use the same design for several different boards: $\frac{\underset{b}{\overset{boards}{Max}}\,[(MaxAC_b - MinCK_b) - \underset{b}{\overset{boards}{Min}}\,(MaxAC_b - MinCK_b)]}{2}$

Table 9–32. Board Skew Parameters (Part 5 of 8)

Parameter	Description
Average delay difference between address and command and CK	A value equal to the average of the longest and smallest address and command signal delay values, minus the delay of the CK signal. The value can be positive or negative. Positive values represent address and command signals that are longer than CK signals; negative values represent address and command signals that are shorter than CK signals. The average delay difference between address and command and CK is expressed by the following equation: $\frac{\sum_{n=1}^{n=n}\left[\left(\frac{\textit{Longest AC Path Delay} + \textit{Shortest AC Path Delay}}{2}\right) - CK_n PathDelay\right]}{n}$ where n is the number of memory clocks. For multi-rank or multiple CS configuration, the equation is: $\frac{\Sigma_{r=1}^{r=r}\left[\text{Average delay differnt between AC and CK in rank r}\right]}{r}$ The Quartus II software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins for DDR2 and DDR3 SDRAM interfaces. You should derive this value through board simulation. For multiple boards, the average delay difference between address and command and CK is expressed by the following equation, if you want to use the same design for several different boards: $\underset{b}{\overset{boards}{Avg}}\left[\left(\frac{MaxAC_b + MinAC_b}{2}\right) - \left(\frac{MaxCK_b + MinCK_b}{2}\right)\right]$
QDR II and QDR II+	
Maximum delay difference between devices	The maximum delay difference of data signals between devices is expressed by the following equation: $Abs\left[\left(\frac{\textit{Longest device 1 delay} - \textit{Shortest device 2 delay}}{2}\right) - \left(\frac{\textit{Longest device 2 delay} - \textit{Shortest device 1 delay}}{2}\right)\right]$ For example, in a two-device configuration there is greater propagation delay for data signals going to and returning from the furthest device relative to the nearest device. This parameter is applicable for depth expansion. Set the value to 0 for non-depth expansion design.
Maximum skew within write data group (ie, K group)	The maximum skew between D and BWS signals referenced by a common K signal.
Maximum skew within read data group (ie, CQ group)	The maximum skew between Q signals referenced by a common CQ signal.

Table 9–32. Board Skew Parameters (Part 6 of 8)

Parameter	Description
Maximum skew between CQ groups	The maximum skew between CQ signals of different read data groups.
Maximum skew within address/command bus	The maximum skew between the address/command signals. $0.5[(MaxACdelay - MinCKdelay) - (MinACdelay - MaxCKdelay)]$
Average delay difference between address/command and K	A value equal to the average of the longest and smallest address/command signal delay values, minus the delay of the K signal. The value can be positive or negative. The average delay difference between the address and command and K is expressed by the following equation: $\frac{\sum_{n=1}^{n=n}\left[\left(\frac{\text{Longest AC Path Delay} + \text{Shortest AC Path Delay}}{2}\right) - K_n PathDelay\right]}{n}$ where n is the number of K clocks.
Average delay difference between write data signals and K	A value equal to the average of the longest and smallest write data signal delay values, minus the delay of the K signal. Write data signals include the D and BWS signals. The value can be positive or negative. The average delay difference between D and K is expressed by the following equation: $\frac{\sum_{n=1}^{n=n}\left[\left(\frac{\text{Longest D Path Delay in } K_n \text{ group} + \text{Shortest D Path Delay in } K_n \text{ group}}{2}\right) - K_n PathDelay\right]}{n}$ where n is the number of DQS groups.
Average delay difference between read data signals and CQ	A value equal to the average of the longest and smallest read data signal delay values, minus the delay of the CQ signal. The value can be positive or negative. The average delay difference between Q and CQ is expressed by the following equation: $\frac{\sum_{n=1}^{n=n}\left[\left(\frac{\text{Longest Q Path Delay in } CQ_n \text{ group} + \text{Shortest Q Path Delay in } CQ_n \text{ group}}{2}\right) - CQ_n PathDelay\right]}{n}$ where n is the number of CQ groups.
RLDRAM II//RLDRAM 3	
Maximum CK delay to device	The delay of the longest CK trace from the FPGA to any device/DIMM is expressed by the following equation: $\max_{n}(CK_n PathDelay)$ where n is the number of memory clocks. For example, the maximum CK delay for two pairs of memory clocks is expressed by the following equation: $\max_{2}(CK_1 PathDelay, CK_2 PathDelay)$

Table 9–32. Board Skew Parameters (Part 7 of 8)

Parameter	Description
Maximum DK delay to device	The delay of the longest DK trace from the FPGA to any device/DIMM is expressed by the following equation: $\max_{n}(DK_n PathDelay)$ where n is the number of DK. For example, the maximum DK delay for two DK is expressed by the following equation: $\max_{2}(DK_1 PathDelay, DK_2 PathDelay)$
Minimum delay difference between CK and DK	The minimum delay difference between the CK signal and any DK signal when arriving at the memory device(s). The value is equal to the minimum delay of the CK signal minus the maximum delay of the DK signal. The value can be positive or negative. The minimum delay difference between CK and DK is expressed by the following equations: $\min_{n,m}(CK_n PathDelay - DK_m PathDelay)$ where n is the number of memory clocks and m is the number of DK. For example, the minimum delay difference between CK and DK for two pairs of memory clocks and four DK signals (two DK signals for each clock) is expressed by the following equation: $\min_{2,2}$ {(Ck_1 Delay–DK_1 Delay), (Ck_1 Delay–DK_2 Delay),(Ck_2 Delay–DK_3 Delay), (Ck_2 Delay–DK_4 Delay)}
Maximum delay difference between CK and DK	The maximum delay difference between the CK signal and any DK signal when arriving at the memory device(s). The value is equal to the maximum delay of the CK signal minus the minimum delay of the DK signal. The value can be positive or negative. The maximum delay difference between CK and DK is expressed by the following equations: $\max_{n,m}(CK_n PathDelay - DK_m PathDelay)$ where n is the number of memory clocks and m is the number of DK. For example, the maximum delay difference between CK and DK for two pairs of memory clocks and four DK signals (two DK signals for each clock) is expressed by the following equation: $\max_{2,2}$ {(Ck_1 Delay–DK_1 Delay), (Ck_1 Delay–DK_2 Delay),(Ck_2 Delay–DK_3 Delay), (Ck_2 Delay–DK_4 Delay)}
Maximum delay difference between devices	The maximum delay difference of data signals between devices is expressed by the following equation: $Abs\left[\left(\frac{\text{Longest device 1 delay} - \text{Shortest device 1 delay}}{2}\right) - \left(\frac{\text{Longest device 2 delay} - \text{Shortest device 2 delay}}{2}\right)\right]$ For example, in a two-device configuration there is greater propagation delay for data signals going to and returning from the furthest device relative to the nearest device. This parameter is applicable for depth expansion. Set the value to 0 for non-depth expansion design.
Maximum skew within QK group	The maximum skew between the DQ signals referenced by a common QK signal.
Maximum skew between QK groups	The maximum skew between QK signals of different data groups.

Table 9–32. Board Skew Parameters (Part 8 of 8)

Parameter	Description
Maximum skew within address/command bus	The maximum skew between the address/command signals. $0.5[(MaxACdelay - MinCKdelay) - (MinACdelay - MaxCKdelay)]$
Average delay difference between address/command and CK	A value equal to the average of the longest and smallest address/command signal delay values, minus the delay of the CK signal. The value can be positive or negative. The average delay difference between the address and command and CK is expressed by the following equation: $\frac{\sum_{(n=1)}^{(n=n)}\left[\left(\frac{\textit{Longest AC Path Delay} + \textit{Shortest AC Path Delay}}{2}\right) - CK_n PathDelay\right]}{n}$ where n is the number of memory clocks.
Average delay difference between write data signals and DK	A value equal to the average of the longest and smallest write data signal delay values, minus the delay of the DK signal. Write data signals include the DQ and DM signals. The value can be positive or negative. The average delay difference between DQ and DK is expressed by the following equation: $\frac{\sum_{(n=1)}^{(n=n)}\left[\left(\frac{\textit{Longest DQ Path Delay in } DK_n \text{ group} + \textit{Shortest DQ Path Delay in } DK_n \text{ group}}{2}\right) - DK_n PathDelay\right.}{n}$ where n is the number of DK groups.
Average delay difference between read data signals and QK	A value equal to the average of the longest and smallest read data signal delay values, minus the delay of the QK signal. The value can be positive or negative. The average delay difference between DQ and QK is expressed by the following equation: $\frac{\sum_{(n=1)}^{(n=n)}\left[\left(\frac{\textit{Longest DQ Path Delay in } QK_n \text{ group} + \textit{Shortest DQ Path Delay in } QK_n \text{ group}}{2}\right) - QK_n PathDelay\right.}{n}$ where n is the number of QK groups.

Controller Settings

Use this tab to apply the controller settings suitable for your design.

This section describes parameters for the High Performance Controller II (HPC II) with advanced features first introduced in version 11.0 for designs generated in version 11.0 or later. Designs created in earlier versions and regenerated in version 11.0 or later do not inherit the new advanced features; for information on parameters for HPC II without the advanced features, refer to the External Memory Interface Handbook for Quartus II version 10.1, available on the Literature: External Memory Interfaces page of the Altera website.

Table 9–33 lists the controller settings.

Table 9–33. Controller Settings (Part 1 of 4)

Parameter		Description
LPDDR2/DDR2/DDR3 SDRAM		
Avalon Interface	**Generate power-of-2 bus widths for SOPC Builder**	Rounds down the Avalon-MM side data bus to the nearest power of 2. You must enable this option for both Qsys and SOPC Builder systems. is option is enabled, the Avalon data buses are truncated to 256 bits wide. One Avalon read-write transaction of 256 bit width maps to four memory beat transactions, each of 72 bits (8 MSB bits are zero, while 64 LSB bits carry useful content). The four memory beats may comprise an entire burst length-of-4 transaction, or part of a burst-length-of-8 transaction.
	Generate SOPC Builder compatible resets	You must enable this option if the IP core is to be used in an SOPC Builder system. When turned on, the reset inputs become associated with the PLL reference clock and the paths must be cut. This option must be enabled for SOPC Builder, but is not required when using the MegaWizard Plug-in Manager or Qsys.
	Maximum Avalon-MM burst length	Specifies the maximum burst length on the Avalon-MM bus. Affects the `AVL_SIZE_WIDTH` parameter.
	Enable Avalon-MM byte-enable signal	When you turn on this option, the controller adds the byte enable signal (avl_be) for the Avalon-MM bus to control the data mask (mem_dm) pins going to the memory interface. You must also turn on **Enable DM pins** if you are turning on this option. When you turn off this option, the byte enable signal (avl_be) is not enabled for the Avalon-MM bus, and by default all bytes are enabled. However, if you turn on **Enable DM pins** with this option turned off, all write words are written.
	Avalon interface address width	The address width on the Avalon-MM interface.
	Avalon interface data width	The data width on the Avalon-MM interface.

Table 9–33. Controller Settings (Part 2 of 4)

Parameter		Description
Low Power Mode	**Enable self-refresh controls**	Enables the self-refresh signals on the controller top-level design. These controls allow you to control when the memory is placed into self-refresh mode.
	Enable auto-power down	Allows the controller to automatically place the memory into power-down mode after a specified number of idle cycles. Specifies the number of idle cycles after which the controller powers down the memory in the auto-power down cycles parameter.
	Auto power-down cycles	The number of idle controller clock cycles after which the controller automatically powers down the memory. The legal range is from 1 to 65,535 controller clock cycles.
Efficiency	**Enable user auto-refresh controls**	Enables the user auto-refresh control signals on the controller top level. These controller signals allow you to control when the controller issues memory autorefresh commands.
	Enable auto-precharge control	Enables the autoprecharge control on the controller top level. Asserting the autoprecharge control signal while requesting a read or write burst allows you to specify whether the controller should close (autoprecharge) the currently open page at the end of the read or write burst.
	Local-to-memory address mapping	Allows you to control the mapping between the address bits on the Avalon-MM interface and the chip, row, bank, and column bits on the memory. Select **Chip-Row-Bank-Col** to improve efficiency with sequential traffic. Select **Chip-Bank-Row-Col** to improve efficiency with random traffic. Select **Row-Chip-Bank-Col** to improve efficiency with multiple chip select and sequential traffic.
	Command queue look-ahead depth	Selects a look-ahead depth value to control how many read or writes requests the look-ahead bank management logic examines. Larger numbers are likely to increase the efficiency of the bank management, but at the cost of higher resource usage. Smaller values may be less efficient, but also use fewer resources. The valid range is from 1 to 16.
	Enable reordering	Allows the controller to perform command and data reordering that reduces bus turnaround time and row/bank switching time to improve controller efficiency.
	Starvation limit for each command	Specifies the number of commands that can be served before a waiting command is served. The valid range is from 1 to 63.

Table 9–33. Controller Settings (Part 3 of 4)

Parameter		Description
Configuration, Status, and Error Handling	Enable Configuration and Status Register Interface	Enables run-time configuration and status interface for the memory controller. This option adds an additional Avalon-MM slave port to the memory controller top level, which you can use to change or read out the memory timing parameters, memory address sizes, mode register settings and controller status. If Error Detection and Correction Logic is enabled, the same slave port also allows you to control and retrieve the status of this logic.
	CSR port host interface	Specifies the type of connection to the CSR port. The port can be exported, internally connected to a JTAG Avalon Master, or both. Select **Internal (JTAG)** to export the CSR port. Select **Avalon-MM Slave** to connect the CSR port to a JTAG Avalon Master. Select **Shared** to export and connect the CSR port to a JTAG Avalon Master.
	Enable error detection and correction logic	Enables ECC for single-bit error correction and double-bit error detection. Your memory interface must be a multiple of 40 or 72 bits wide to use ECC.
	Enable auto error correction	Allows the controller to perform auto correction when a single-bit error is detected by the ECC logic.
Advanced Controller Features	Enable half rate bridge	Turn on this option to enable half rate bridge block.
	Enable hard memory controller	Turn on this option to enable hard memory controller.
Multiple Port Front End	Export bonding port	Turn on this option to export bonding interface for wider avalon data width with two controllers. Bonding ports are exported to the top level.
	Number of ports	Specifies the number of Avalon-MM Slave ports to be exported. The number of ports depends on the width and the type of port you selected. There are four 64-bit read FIFOs and four 64-bit write FIFOs in the multi-port front-end (MPFE) component. For example, If you select 256 bits width and bidirectional slave port, all the FIFOs are fully utilized, therefore you can only select one port.
	Width	Specifies the local data width for each Avalon-MM Slave port. The width depends on the type of slave port and also the number of ports selected. This is due to the limitation of the FIFO counts in the MPFE. There are four 64-bit read FIFOs and four 64-bit write FIFOs in the MPFE. For example, if you select one bidirectional slave port, you can select up to 256 bits to utilize all the read and write FIFOs. As a general guideline to choosing an optimum port width for your half-rate or quarter-rate design, apply the following equation: port width = 2 x DQ width x Interface width multiplier where the interface width multiplier is 2 for half-rate interfaces and 4 for quarter-rate interfaces.
	Priority	Specifies the absolute priority for each Avalon-MM Slave port. Any transaction from the port with higher priority number will be served before transactions from the port with lower priority number.
	Weight	Specifies the relative priority for each Avalon-MM Slave port. When there are two or more ports having the same absolute priority, the transaction from the port with higher (bigger number) relative weight will be served first. You can set the weight from a range of 0 to 32.
	Type	Specifies the type of Avalon MM slave port to either a bidirectional port, read only port or write only port.

Table 9–33. Controller Settings (Part 4 of 4)

Parameter	Description
QDR II/QDR II+ SRAM and RLDRAM II	
Generate power-of-2 data bus widths for SOPC Builder	Rounds down the Avalon-MM side data bus to the nearest power of 2. You must enable this option for both Qsys and SOPC Builder systems.
Generate SOPC Builder compatible resets	This option must be enabled if this core is to be used in an SOPC Builder system.
Maximum Avalon-MM burst length	Specifies the maximum burst length on the Avalon-MM bus.
Enable Avalon-MM byte-enable signal	When you turn on this option, the controller adds a byte-enable signal (`avl_be_w`) for the Avalon-MM bus, in which controls the `bws_n` signal on the memory side to mask bytes during write operations. When you turn off this option, the `avl_be_w` signal is not available and the controller will always drive the memory `bws_n` signal so as to not mask any bytes during write operations.
Avalon interface address width	Specifies the address width on the Avalon-MM interface.
Avalon interface data width	Specifies the data width on the Avalon-MM interface.
Reduce controller latency by	Specifies the number of clock cycles by which to reduce controller latency. Lower controller latency results in lower resource usage and f_{MAX} while higher latency results in higher resource usage and f_{MAX},
Enable user refresh	Enables user-controlled refresh. Refresh signals will have priority over read/write requests. This option is available for RLDRAM II only.
Enable error detection parity	Enables per-byte parity protection. This option is available for RLDRAM II only

Diagnostics

The **Diagnostics** tab allows you to set parameters for certain diagnostic functions.

Table 9–34 describes parameters for simulation.

Table 9–34. Simulation Options

Parameter	Description
Simulation Options	
Auto-calibration mode	Specifies whether you want to improve simulation performance by reducing calibration. There is no change to the generated RTL. The following autocalibration modes are available: ■ **Skip calibration**—provides the fastest simulation. It loads the settings calculated from the memory configuration and enters user mode. ■ **Quick calibration**—calibrates (without centering) one bit per group before entering user mode. ■ **Full calibration**—calibrates the same as in hardware, and includes all phases, delay sweeps, and centering on every data bit. You can use timing annotated memory models. Be aware that full calibration can take hours or days to complete. To perform proper PHY simulation, select **Quick calibration** or **Full calibration**. For more information, refer to the "Simulation Options" section in the *Simulating Memory IP* chapter. For QDR II, QDR II+ SRAM, and RLDRAM II, the Nios II-based sequencer must be selected to enable the auto calibration modes selection.
Skip memory initialization delays	When you turn on this option, required delays between specific memory initialization commands are skipped to speed up simulation.
Enable verbose memory model output	Turn on this option to display more detailed information about each memory access during simulation.
Enable support for Nios II ModelSim® flow in Eclipse	Initializes the memory interface for use with the **Run as Nios II ModelSim** flow with Eclipse. This option is not available for QDR II and QDR II+ SRAM.
Debug Options	
Debug level	Specifies the debug level of the memory interface.
Efficiency Monitor and Protocol Checker Settings	
Enable the Efficiency Monitor and Protocol Checker on the Controller Avalon Interface	Enables efficiency monitor and protocol checker block on the controller Avalon interface. This option is not available for QDR II and QDR II+ SRAM.

Document Revision History

Table 9–35 lists the revision history for this document.

Table 9–35. Document Revision History

Date	Version	Changes
November 2012	6.0	■ Added RLDRAM 3 information ■ Added LPDDR2 information. ■ Changed chapter number from 8 to 9.
June 2012	5.0	■ Added number of sharing interfaces parameters to Clock Parameters table. ■ Added **DQ/DQS Package Deskew** and **Address/Command Package Deskew** descriptions to Board Skew Parameters table. ■ Added equations for multiple boards to several parameter descriptions in Board Skew Parameters table. ■ Added Feedback icon.
November 2011	4.0	■ Updated Installation and Licensing section. ■ Combined Qsys and SOPC Builder Interfaces sections. ■ Combined parameter settings for DDR, DDR2, DDR3 SDRAM, QDRII SRAM, and RLDRAM II for both ALTMEMPHY and UniPHY IP. ■ Added parameter usage details to Parameterizing Memory Controllers with UniPHY IP section. ■ Moved "Functional Description" section for DDR, DDR2, DDR3 SDRAM, QDRII SRAM, and RLDRAM II to volume 3 of the *External Memory Interface Handbook*.
June 2011	3.0	■ Removed references to High-Performance Controller. ■ Updated High-Performance Controller II information. ■ Removed HardCopy III, HardCopy IV E, HardCopy IV GX, Stratix III, and Stratix IV support. ■ Updated Generated Files lists. ■ Added Qsys and SOPC Builder Interfaces section.
December 2010	2.1	Updated the following items for 10.1: ■ Updated Design Flows and Generated Files information. ■ Updated Parameterizing Memory Controllers with UniPHY IP chapter.
July 2010	2.0	■ Added information for new GUI parameters: **Controller latency**, **Enable reduced bank tracking for area optimization**, and **Number of banks to track**. ■ Removed information about IP Advisor. This feature is removed from the DDR/DDR2 SDRAM IP support for version 10.0.
February 2010	1.3	Corrected typos.
February 2010	1.2	■ Full support for Stratix IV devices. ■ Added timing diagrams for initialization and calibration stages for HPC.
November 2009	1.1	Minor corrections.
November 2009	1.0	First published.

10. Simulating Memory IP

EMI_DG_009-5.1

This chapter describes the simulation basics so that you are aware of the supported simulators and options available to you when you perform functional simulation with Altera® external memory interface IP.

You need the following components to simulate your design:

- A simulator—The simulator must be any Altera-supported VHDL or Verilog HDL simulator
- A design using one of Altera's external memory IP
- An example driver (to initiate read and write transactions)
- A testbench and a suitable memory simulation model

Memory Simulation Models

There are two types of memory simulation models. You can use one of the following memory models:

- Altera-provided generic memory model.

  The Quartus® II software generates this model together with the example design and this model adheres to all the memory protocol specifications. You can parameterize the generic memory model.

- Vendor-specific memory model.

  Memory vendors such as Micron and Samsung provide simulation models for specific memory components that you can download from their websites. Although Denali models are also available, currently Altera does not provide support for Denali models. All memory vendor simulation models that you use to simulate Altera memory IP must be JEDEC compliant.

Simulation Options

With the example testbench, the following simulation options are available to improve simulation speed:

- Full calibration—Calibrates the same way as in hardware, and includes all phase, delay sweeps, and centering on every data bit.
- Quick calibration—Calibrates the read and write latency only, skipping per bit deskew.
- Skip calibration—Provides the fastest simulation. It loads the settings calculated from the memory configuration and enters user mode.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

By default, the UniPHY IP generates abstract PHY, which uses skip calibration regardless of the simulation options that you chose in the MegaWizard™-Plug In Manager.

Table 10–1 lists the typical simulation times implemented using UniPHY IP.

These simulation times are estimates based on average run times of a few example designs. The simulation times for your design may vary depending on the memory interface specifications, simulator, or the system you are using.

Table 10–1. Typical Simulation Times Using UniPHY IP

Calibration Mode/Run Time (1)	Simulation Time	
	Small Interface	Large Interface (×72 Quad Rank)
Full ■ Full calibration ■ Includes all phase/delay sweeps and centering	10 minutes	~ 1 day
Quick ■ Scaled down calibration ■ Calibrate one pin	3 minutes	4 hours
Skip ■ Skip all calibration, jump to user mode ■ Preload calculated settings	3 minutes	20 minutes

Note to Table 10–1:

(1) Uses one loop of driver test. One loop of driver is approximately 600 read or write requests, with burst length up to 64.

For more information about steps to follow before simulating, modifying the vendor memory model, and simulation flow for both ALTMEMPHY and UniPHY IPs, refer to the "Simulation Walkthrough with UniPHY IP" on page 10–3 and "Simulation Walkthrough with ALTMEMPHY IP" on page 10–16.

Simulation Walkthrough with UniPHY IP

Simulating the whole memory interface is a good way to determine the latency of your system. However, the latency found in simulation may be different than the latency found on the board because functional simulation does not take into account board trace delays and different process, voltage, and temperature scenarios. For a given design on a given board, the latency found may differ by one clock cycle (for full-rate designs) or two clock cycles (for half-rate designs) upon resetting the board. Different boards can also show different latencies even with the same design.

The UniPHY IP supports only functional simulation. Functional simulation is supported at the RTL level and after generating a post-fit functional simulation netlist. The post-fit netlist for designs that contain UniPHY IP is a hybrid of the gate level (for FPGA core) and RTL level (for the external memory interface IP).

Altera recommends that you validate the functional operation of your design via RTL simulation, and the timing of your design using TimeQuest Timing Analysis.

For high-performance memory controllers with UniPHY IP, you can simulate a functional simulation example design generated by the MegaWizard Plug-In Manager. The MegaWizard Plug-In Manager generates the relevant files to the \<*variation_name*>**_example_design** directory.

You can use the IP functional simulation model with any Altera-supported VHDL or Verilog HDL simulator.

The UniPHY IP does not support an SOPC Builder simulation flow.

After you have generated the memory IP, view the README. txt file located in the \<*variation_name*>**_example_design\simulation** directory for instructions on how to generate the simulation example design for Verilog HDL or VHDL. The **README.txt** file also explains how to run simulation using the ModelSim-Altera software. Altera provides simulation scripts for the Mentor, Cadence, Aldec, and Synopsis simulators; however, detailed instructions on how to perform simulation using these third party simulators are not provided.

Simulation Scripts

The Quartus II software generates three simulation scripts during project generation for four different third party simulation tools—Cadence, Synopsys, Aldec, and Mentor. These scripts reduces the number of files that you need to compile separately before simulating a design. These scripts are located in three separate folders under the <*project directory*>\<*varitation_name*>**_sim** directory, each named after the names of the simulation tools. The example designs also provide equivalent scripts after you run the .tcl script from the project located in the <*name*>**_example_design\simulation** directory.

Preparing the Vendor Memory Model

You can replace the Altera-supplied memory model with a vendor-specific memory model. In general, you may find vendor-specific models to be standardized, thorough, and well supported, but sometimes more complex to setup and use. Please note that Altera does not provide support for vendor-specific memory models. If you do want to replace the Altera-supplied memory model with a vendor-supplied memory model, observe the following guidelines:

- Ensure that you have the correct vendor-supplied memory model for your memory device.
- Disconnect all signals from the default memory model and reconnect them to the vendor-supplied memory model.
- If you intend to run simulation from the Quartus II software, ensure that the **.qip** file points to the vendor-supplied memory model.

When you are using a vendor memory model, instead of the MegaWizard-generated functional simulation model, you need to make some modifications to the vendor memory model and the testbench files by following these steps:

1. Obtain and copy the vendor memory model to the \<*variation_name*>**_example_design\simulation**\<*variation_name*>**_sim\submodules** directory. For example, obtain the **ddr2.v** and **ddr2_parameters.vh** simulation model files from the Micron website and save them in the directory.

   The auto-generated generic SDRAM model may be used as a placeholder for a specific vendor memory model.

   Some vendor DIMM memory models do not use data mask (DM) pin operation, which can cause calibration failures. In these cases, use the vendor's component simulation models directly.

2. Open the vendor memory model file in a text editor and specify the speed grade and device width at the top of the file. For example, you can add the following statements for a DDR2 SDRAM model file:

   ```
   'define sg25

   'define x8
   ```

   The first statement specifies the memory device speed grade as –25 (for 400 MHz operation). The second statement specifies the memory device width per DQS.

3. Check that the following statement is included in the vendor memory model file. If not, include it at the top of the file. This example is for a DDR2 SDRAM model file:

   ```
   `include "ddr2_parameters.vh"
   ```

4. Save the vendor memory model file.
5. Open the simulation example project file <*variation_name*>**_example_sim.qpf**, located in the <*variation_name*>**_example_design\simulation** directory.
6. On the Tools menu, select **TCL scripts** to run the **generate_sim_verilog_example_design.tcl** file, in which generates the simulation example design.

7. To enable vendor memory model simulation, you have to include and compile the vendor memory model by adding it into the simulation script. Open the **.tcl** script, **msim_setup.tcl**, located in the *<variation_name>***_example_design\simulation\verilog\mentor** directory in the text editor. Add in the following line in the '`# Compile the design files in correct order`' section:
```
vlog     +incdir+$QSYS_SIMDIR/submodules/
"$QSYS_SIMDIR/submodules/<vendor_memory>.v"
-work <variation_name>_example_sim_work
```
8. Open the simulation example design, *<variation_name>***_example_sim.v**, located in the *<variation_name>***_example_design\simulation\verilog** directory in a text editor and delete the following module:

   ```
   alt_mem_if_<memory_type>_mem_model_top_<memory_type>_mem_if_dm_pins_en_
   mem_if_dqsn_en
   ```

   The actual name of the pin may differ slightly depending on the memory controller you are using.

9. Instantiate the downloaded memory model and connect its signals to the rest of the design.
10. Ensure that the ports names and capitalization in the memory model match the port names and capitalization in the testbench.

    The vendor memory model may use different pin names and capitalization than the MegaWizard-generated functional model.

11. Save the testbench file.

The original instantiation may be similar to the following code:

```
alt_mem_if_ddr2_mem_model_top_mem_if_dm_pins_en_mem_if_dqsn_en #(
      .MEM_IF_ADDR_WIDTH            (13),
      .MEM_IF_ROW_ADDR_WIDTH        (12),
      .MEM_IF_COL_ADDR_WIDTH        (8),
      .MEM_IF_CS_PER_RANK           (1),
      .MEM_IF_CONTROL_WIDTH         (1),
      .MEM_IF_DQS_WIDTH             (1),
      .MEM_IF_CS_WIDTH              (1),
      .MEM_IF_BANKADDR_WIDTH        (3),
      .MEM_IF_DQ_WIDTH              (8),
      .MEM_IF_CK_WIDTH              (1),
      .MEM_IF_CLK_EN_WIDTH          (1),
      .DEVICE_WIDTH                 (1),
      .MEM_TRCD                     (6),
      .MEM_TRTP                     (3),
      .MEM_DQS_TO_CLK_CAPTURE_DELAY (100),
      .MEM_IF_ODT_WIDTH             (1),
```

```
        .MEM_MIRROR_ADDRESSING_DEC    (0),
        .MEM_REGDIMM_ENABLED          (0),
        .DEVICE_DEPTH                 (1),
        .MEM_INIT_EN                  (0),
        .MEM_INIT_FILE                (""),
        .DAT_DATA_WIDTH               (32)
    ) m0 (
        .mem_a      (e0_memory_mem_a),      // memory.mem_a
        .mem_ba     (e0_memory_mem_ba),     //        .mem_ba
        .mem_ck     (e0_memory_mem_ck),     //        .mem_ck
        .mem_ck_n   (e0_memory_mem_ck_n),   //        .mem_ck_n
        .mem_cke    (e0_memory_mem_cke),    //        .mem_cke
        .mem_cs_n   (e0_memory_mem_cs_n),   //        .mem_cs_n
        .mem_dm     (e0_memory_mem_dm),     //        .mem_dm
        .mem_ras_n  (e0_memory_mem_ras_n),  //        .mem_ras_n
        .mem_cas_n  (e0_memory_mem_cas_n),  //        .mem_cas_n
        .mem_we_n   (e0_memory_mem_we_n),   //        .mem_we_n
        .mem_dq     (e0_memory_mem_dq),     //        .mem_dq
        .mem_dqs    (e0_memory_mem_dqs),    //        .mem_dqs
        .mem_dqs_n  (e0_memory_mem_dqs_n),  //        .mem_dqs_n
        .mem_odt    (e0_memory_mem_odt)     //        .mem_odt
    );
```

Replace the original code with the following code:

```
ddr2 memory_0 (
.addr (e0_memory_mem_a), // memory.mem_a
.ba (e0_memory_mem_ba), // .mem_ba
.clk (e0_memory_mem_ck), // .mem_ck
.clk_n (e0_memory_mem_ck_n), // .mem_ck_n
.cke (e0_memory_mem_cke), // .mem_cke
.cs_n (e0_memory_mem_cs_n), // .mem_cs_n
.dm_rdqs (e0_memory_mem_dm), // .mem_dm
.ras_n (e0_memory_mem_ras_n), // .mem_ras_n
.cas_n (e0_memory_mem_cas_n), // .mem_cas_n
.we_n (e0_memory_mem_we_n), // .mem_we_n
.dq (e0_memory_mem_dq), // .mem_dq
.dqs (e0_memory_mem_dqs), // .mem_dqs
.rdqs_n (), // .mem_dqs_n
.dqs_n (e0_memory_mem_dqs_n), // .mem_dqs_n
.odt (e0_memory_mem_odt) // .mem_odt);
```

If you are interfacing with a DIMM or multiple memory components, you need to instantiate all the memory components in the simulation file.

Functional Simulations

This topic discusses VHDL and Verilog HDL simulations with UniPHY IP example design.

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Mentor Graphics ModelSim® and QuestaSim Support* chapter in volume 3 of the *Quartus II Software Handbook*.

Verilog HDL

Altera provides simulation scripts for you to run the example design. The simulation scripts are for Synopsys, Cadence, Aldec, and Mentor simulators. These simulation scripts are located in the following main folder locations:

Simulation scripts in the simulation folders are located as follows:

- *<variation_name>***_example_design\simulation\verilog\mentor\msim_setup.tcl**
- *<variation_name>***_example_design\simulation\verilog\synopsys\vcs\vcs_setup.sh**
- *<variation_name>***_example_design\simulation\verilog\synopsys\vcsmx\vcsmx_setup.sh**
- *<variation_name>***_example_design\simulation\verilog\aldec\rivierapro_setup.tcl**
- *<variation_name>***_example_design\simulation\verilog\cadence\ncsim_setup.sh**

Simulation scripts in the <>**_sim_folder** are located as follows:

- *<variation_name>***_sim\mentor\msim_setup.tcl**
- *<variation_name>***_sim\cadence\ncsim_setup.sh**
- *<variation_name>***_sim\synopsys\vcs\vcs_setup.sh**
- *<variation_name>***_sim\synopsys\vcsmx\vcsmx_setup.sh**
- *<variation_name>***_sim\aldec\rivierapro_setup.tcl**

VHDL

The UniPHY VHDL fileset is specifically for customers who use VHDL exclusively and who do not have a mixed-language (VHDL and Verilog) simulation license. All other customers should either select the Verilog language option during generation, or simulate using the synthesis fileset.

The UniPHY IP VHDL simulation fileset consists of the following types of files:

- IPFS-generated VHDL files
- IEEE Encrypted Verilog HDL files (for Mentor, and in addition the equivalent plain-text Verilog files for all simulators that support mixed-language simulations).
- Plain-text VHDL files.

Although the IEEE Encrypted files are written in Verilog, you can simulate these files in combination with VHDL without violating the single-language restrictions in ModelSim because they are encrypted.

Because the VHDL fileset consists of both VHDL and Verilog files, you must follow certain mixed-language simulation guidelines. The general guideline for mixed-language simulation is that you must always link the Verilog files (whether encrypted or not) against the Verilog version of the Altera libraries, and the VHDL files (whether simgen-generated or pure VHDL) against the VHDL libraries.

Altera provides simulation scripts for you to run the example design. The simulation scripts are for Synopsys, Cadence, Aldec, and Mentor simulators. These simulation scripts are located in the following main folder locations:

Simulation scripts in the simulation folders are located as follows:

- *<variation_name>*_**example_design\simulation\vhdl\mentor\msim_setup.tcl**
- *<variation_name>*_**example_design\simulation\vhdl\synopsys\vcsmx\vcsmx_setup.sh**
- *<variation_name>*_**example_design\simulation\vhdl\cadence\ncsim_setup.sh**
- *<variation_name>*_**example_design\simulation\vhdl\aldec\rivierapro_setup.tcl**

Simulation scripts in the <>_**sim_folder** are located as follows:

- *<variation_name>*_**sim\mentor\msim_setup.tcl**
- *<variation_name>*_**sim\cadence\ncsim_setup.sh**
- *<variation_name>*_**sim\synopsys\vcsmx\vcsmx_setup.sh**
- *<variation_name>*_**sim\aldec\rivierapro_setup.tcl**

Simulating the Example Design

The following section describes how to simulate the example design in Cadence, Synopsys, and Mentor simulators.

To simulate the example design in the Quartus II software using the Cadence simulator, follow these steps:

1. At the Linux shell command prompt, change directory to <*name*>**_example_design\simulation**<*verilog/vhdl*>**\cadence**
2. Run the simulation by typing the following commandat the command prompt:

   ```
   sh ncsim_setup.sh
   ```

To simulate the example design in the Quartus II software using the Synopsys simulator, follow these steps:

1. At the Linux shell command prompt, change directory to <*name*>**_example_design\simulation**<*verilog/vhdl*>**\synopsys\vcsmx**
2. Run the simulation by typing the following command at the command prompt:

   ```
   sh vcsmx_setup.sh
   ```

To simulate the example design in the Quartus II software using the Mentor simulator, follow these steps:

1. At the Linux or Windows shell command prompt, change directory to <*name*>**_example_design\simulation**<*verilog/vhdl*>**\mentor**
2. Execute the **msim_setup.tcl** script that automatically compiles and runs the simulation by typing the following command at the Linux or Windows command prompt:

   ```
   vsim -do run.do
   ```

   or

   Type the following command at the ModelSim command prompt:

   ```
   do run.do
   ```

   This simulation method is only applicable to UniPHY.

To simulate the example design in the Quartus II software using the Aldec simulator, follow these steps:

1. At the Linux or Windows shell command prompt, change directory to <*name*>**_example_design\simulation**<*verilog/vhdl*>**\aldec**
2. Execute the **rivierapro_setup.tcl** script that automatically compiles and runs the simulation by typing the following command at the Linux or Windows command prompt:

   ```
   vsim -do rivierapro.tcl
   ```

3. To compile and elaborate the design after the script loads, type `ld_debug`.
4. Type `run -all` to run the simulation.

For more information about simulation, refer to the *Simulating Altera Designs* chapter in volume 3 of the *Quartus II Handbook*.

If your Quartus II project appears to be configured correctly but the example testbench still fails, check the known issues on the Knowledge Database page of the Altera website before filing a service request.

Abstract PHY

In the Quartus II software version 11.1, UniPHY IP generates both synthesizable and abstract models for simulation, with the abstract model as default. The UniPHY abstract model replaces the PLL with simple fixed-delay model, and the detailed models of the hard blocks with simple cycle-accurate functional models.

The UniPHY abstract model always runs in skip calibration mode, regardless of the auto-calibration mode you select in the parameter editor because calibration does not support abstract models. For VHDL, the UniPHY abstract model is the only option because you cannot switch to regular simulation model. The PLL frequencies in simulation may differ from the real time simulation due to pico-second timing rounding.

However, you can switch to regular simulation models for Verilog HDL language. The full and quick calibration modes are available for regular simulation models.

Add an additional command line to the compilation script for the two relevant files to enable regular PHY simulation:

```
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
```

The two relevant files are:

- In <*variation_name*>**_example_design/simulation/verilog/submodules**:

  <*variation_name*>**_example_sim_e0_if0_p0.sv** and
  <*variation_name*>**_example_sim_e0_if0_pll0.sv**

or

- In <*variation_name*>**_sim/submodules:**

  <*variation_name*>**_p0.sv** and
  <*variation_name*>**_pll0.sv**

To switch to regular simulation models for the Verilog HDL language on the example simulation design, follow the appropriate steps for your simulator:

- For the Mentor simulator, edit the **msim_setup.tcl** file as follows:

vlog
- sv "**$QSYS_SIMDIR/submodules/**<*variation_name*>**_example_sim_e0_if0_p0.sv**"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
-work <*variation_name*>**_example_sim_work**

vlog - sv
"**$QSYS_SIMDIR/submodules/**<*variation_name*>**_example_sim_e0_if0_pll0.sv**"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
-work <*variation_name*>**_example_sim_work**

- For the Cadence simulator, edit the **ncsim_setup.sh** file as follows:

ncvlog
- sv "**$QSYS_SIMDIR/submodules/**<*variation_name*>**_example_sim_e0_if0_p0.sv**"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
-work <*variation_name*>**_example_sim_work**
- cdslib **./cds_libs/skip_example_sim_work.cds.lib**

ncvlog - sv
"**$QSYS_SIMDIR/submodules/**<*variation_name*>**_example_sim_e0_if0_pll0.sv**"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
-work <*variation_name*>**_example_sim_work**
- cdslib **./cds_libs/**<*variation_name*>**_example_sim_work.cds.lib**

- For the Synopsys simulator, edit the **vscmx_setup.sh** file as follows:

vlogan +v2k - sverilog
"**$QSYS_SIMDIR/submodules/**<*variation_name*>**_example_sim_e0_if0_p0.sv**"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
- work <*variation_name*>**_example_sim_work**

vlogan +v2k - sverilog
"**$QSYS_SIMDIR/submodules/**<*variation_name*>**_example_sim_e0_if0_pll0.sv**"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
- work <*variation_name*>**_example_sim_work**

If you use the UniPHY abstract model, the simulation is two times faster in magnitude if compared to the real simulation model. Instantiating a standalone UniPHY IP in your design further improves the simulation time if you use a half-rate controller with UniPHY or a larger memory DQ width.

PHY-Only Simulation

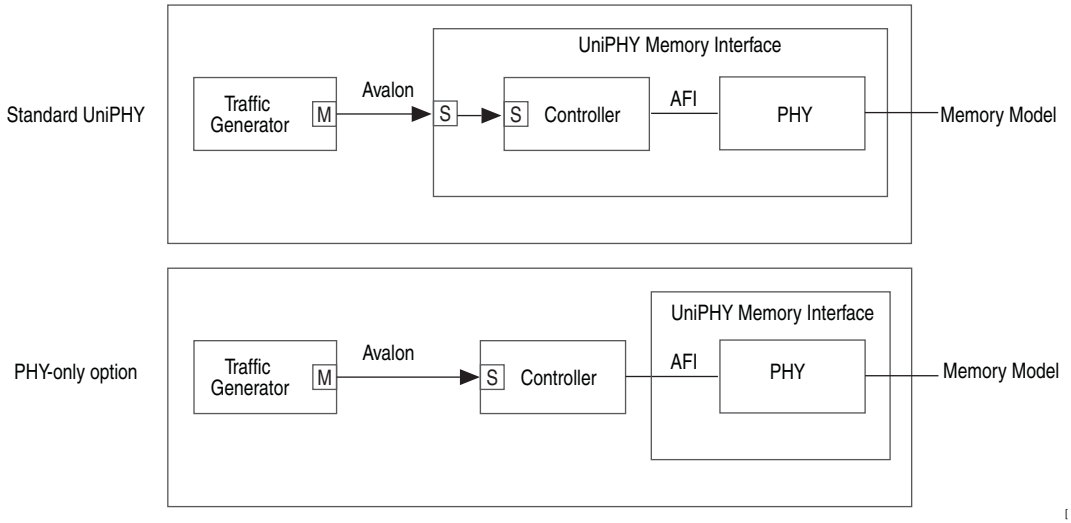
The PHY-only simulation option is a new feature in the Quartus II version 11.1 software. To enable this feature in the parameter editor, under PHY Settings tab, in the FPGA section, turn on **Generate PHY only**. This setting also applies to designs using Qsys. This option allows you to replace the Altera high-performance memory controllers with your own custom controller.

For more information about using a custom controller, refer to "Using a Custom Controller" section in the *Functional Description—ALTMEMPHY* chapter of the *External Memory Interface Handbook*.

When you are using a standard UniPHY memory interface, by default, the parameter editor generates an external memory interface with a controller and a PHY. The controller and PHY are connected internally with the Altera PHY interface (AFI). The memory interface has an Avalon slave port that connects to the controller to allow communication from the user logic. When you turn on the PHY-only option, the parameter editor generates the PHY without the controller. In this case, the PHY is accessed via the AFI port, which can be externally connected to a custom controller. In the example design, a controller is instantiated externally to the memory interface. This provides a fully functional example design and demonstrates how to connect the controller to manage the transactions from the traffic generator.

Figure 10–1 shows the difference in the UniPHY memory interface when the PHY-only option is enabled.

Figure 10–1. PHY-only Option



Post-fit Functional Simulation

The post-fit functional simulation does not work for the UniPHY IP core because of the following inherent problems:

- The UniPHY sample 'X's during calibration, in which causes an issue during timing simulation
- Some internal transfers that are 0-cycle require delays to properly function in a post-fit netlist

To enable functional simulation for a design that uses UniPHY IP core, a quasi-post fit scheme is implemented. This scheme allows gate-level simulation of the full design (excluding the UniPHY IP), while you use RTL simulation for the UniPHY IP. The quasi-post-fit scheme involves partitioning blocks in the EMIF and swaping them with simulation RTL. With this workaround the memory interface is partially post-fit RTL and partially premap RTL, therefore the simulation flow is not impeded.

Assuming that the Uniphy IP has been generated and inserted in some larger design, follow these steps to run post-fit simulation:

1. In the Quartus II software, set up a project that contains a UniPHY IP core.
2. On the Assignments menu, click **Assignment Editor**.
3. In the assignment editor, add the global assignment “VERILOG_MACRO” and set the value to “SYNTH_FOR_SIM=1”.
4. On the Assignments menu, click **Settings**.
5. In the **Category** list, under **EDA Tools Settings**, select **Simulation**.
6. On the Simulation page, select a tool name (for example, ModelSim-Altera).
7. In the **Format for output netlist** list, select a HDL language.

8. In the **Output directory** box, type or browse to the location where you want output files saved.
9. Click **More EDA Netlist Writer Settings** to choose from a list of other options.
10. Set the value for **Maintain hierarchy** to **PARTITION_ONLY**, and click **OK**.
11. Elaborate the project. On the Processing menu, select **Start** and click Sta**rt Hierarchy Elaboration**.
12. In the Project Navigator window, click the **Hierarchy** tab. In the **Entity** box, locate the instances for the following devices:
    a. For instances in Stratix III, Stratix IV, Arria II GX, Arria II GZ , click the + icon to expand the following top-level design entities, right-click on the lower-level entities, select **Design Partition**, and click **Set as Design Partition**:
       - *<hierarchy path to Uniphy top-level>**<name>*`_if0:if0\`*<name>*`_if0_p0:p0`
       - *<hierarchy path to Uniphy top-level>**<name>*`_if0:if0\`*<name>*`_if0_s0:s0`
    b. For instances in Arria V or Stratix V, click the + icon to expand the following top-level design entity, right-click on the lower-level entities, select **Design Partition**, and click **Set as Design Partition**:
       - *<hierarchy path to Uniphy top-level>**<name>*`_if0:if0\`*<name>*`_if0_s0:s0`

       For instances of hard memory interfaces in Arria V, no design partition is necessary.
13. In the **Design Partitions Window**, ensure that the netlist type value of the design partitions listed in Step12 a and 12b are set to **Post-synthesis**.
14. On the Processing menu, select **Start** and click **Start Analysis and Synthesis**.
15. Run the Pin assignments script. To run the pin assignment script, follow these steps:
    a. On the **Tools** menu,click **TCL Scripts**.
    b. In the **Libraries** list, locate the *<name>***_pin_assignment.tcl**.
    c. Click **Run**.
16. On the Processing menu, select **Start** and click **Partition Merge**.
17. On the Processing menu, select **Start** and click **Start Fitter**.
18. On the Processing menu, select **Start** and click **Start EDA netlist writer**.
19. The output post-fit netlist is located in the directory you chose in Step 8.
20. Assume that the netlist filename is **dut.vo** (or **dut.vho** for VHDL). Replace the instance of the partitioned modules (specified in step 12) in **dut.vo** and instantiate the original instance of the RTL. As a result, the RTL of those modules will simulate correctly instead of the the post-fit netlist. For example, you can delete the definition of the *<name>*`_if0_s0` (and *<name>*`_if0_p0`, if appropriate) modules in the post-fit netlist, and ensure that your simulator compiles the post-fit netlist and all the UniPHY RTL in order to properly link these modules for simulation.

    (This step does not apply to hard memory interfaces on Arria V devices.)

21. To match the post-fit netlist instantiation of `s0` (and `p0`, if appropriate) with the original RTL module definition (specified in step 12), you must also account for three device input ports that are added to the post-fit netlist. The easiest way to do this is to delete the following three connections from the `s0` (and `p0`, if appropriate) instances in the post-fit netlist:
    - `.devpor(devpor)`
    - `.devclrn(devclrn)`
    - `.devoe(devpoe)`

    (This step does not apply to hard memory interfaces on Arria V devices.)
22. For Stratix V the *<name>*`_if0_s0` instance in the post-fit netlist will also have a connection `.QIC_GND_PORT(` <wire name>) that you must delete because it does not match with the original RTL module.

    (This step does not apply to hard memory interfaces on Arria V devices.)
23. Set up and run your simulator.

Simulation Issues

When you simulate an example design in the ModelSim, you might see the following warnings, which are expected and not harmful:

```
# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (1 or 1) does not match connection size (7) for port
'local_size'.

#         Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (9 or 9) does not match connection size (1) for port
'ctl_cal_byte_lane_sel_n'.

#         Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (18 or 18) does not match connection size (1) for port
'afi_doing_read'.

#         Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (2 or 2) does not match connection size (1) for port
'afi_rdata_valid'.

#         Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (112 or 112) does not match connection size (1) for port
'bank_information'.

#         Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst
```

```
# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (8 or 8) does not match connection size (1) for port
'bank_open'.

#         Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3017)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_alt_ddrx_bank_timer_w
rapper.v(1191): [TFMPC] - Too few port connections. Expected 127, found 126.

#         Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst/
bank_timer_wrapper_inst/bank_timer_inst

# ** Warning: (vsim-3722)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_alt_ddrx_bank_timer_w
rapper.v(1191): [TFMPC] - Missing connection for port 'wr_to_rd_to_pch_all'.

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_alt_ddrx_bank_timer_w
rapper.v(1344): [PCDPC] - Port size (5 or 5) does not match connection size (1) for
port 'wr_to_rd_to_pch_all'.

#         Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst/
bank_timer_wrapper_inst/rank_monitor_inst

# ** Warning: (vsim-8598) Non-positive replication multiplier inside concat.
Replication will be ignored

Warning-[OSPA-N] Overriding same parameter again

/p/eda/acd/altera/quartusII/10.1/quartus/eda/sim_lib/synopsys/stratixv_atoms_ncryp
t.v, 8499

Warning-[ZONMCM] Zero or negative multiconcat multiplier
../quartus_stratix5/ddr3_ctlr_sim/ddr3_ctlr_sequencer.sv, 916

  Zero or negative multiconcat multiplier is found in design. It will be   replaced
by 1'b0.

  Source info: {INIT_COUNT_WIDTH {1'b0}}

Warning-[PCWM-W] Port connection width mismatch
../quartus_stratix5/ddr3_ctlr_sim/ddr3_ctlr_sequencer_cpu.v, 2830

"the_sequencer_cpu_nios2_oci_itrace"

  The following 38-bit expression is connected to 16-bit port "jdo" of module
"ddr3_ctlr_sequencer_cpu_nios2_oci_itrace", instance
"the_sequencer_cpu_nios2_oci_itrace".

  Expression: jdo

        use +lint=PCWM for more details
```

Simulation Walkthrough with ALTMEMPHY IP

For high-performance memory controllers with ALTMEMPHY IP, you can simulate the example top-level file with the MegaWizard-generated IP functional simulation models. The MegaWizard™ Plug-In Manager generates a VHDL or Verilog HDL testbench for the example top-level file, which is in the **\testbench** directory of your project directory.

You can use the IP functional simulation model with any Altera-supported VHDL or Verilog HDL simulator. You can perform a simulation in a third-party simulation tool from within the Quartus II software, using NativeLink.

The ALTMEMPHY megafunction cannot be simulated alone. To simulate the ALTMEMPHY megafunction, you must use all of the following blocks:

- Memory controller
- Example driver (to initiate read and write transactions)
- Testbench and a suitable vendor memory model

Simulating the whole memory interface is a good way to determine the latency of your system. However, the latency found in simulation may be different than the latency found on the board because functional simulation does not take into account board trace delays and different process, voltage, and temperature scenarios. For a given design on a given board, the latency found may differ by one clock cycle (for full-rate designs) or two clock cycles (for half-rate designs) upon resetting the board. Different boards can also show different latencies even with the same design.

☞ The ALTMEMPHY megafunction only supports functional simulation; it does not support gate-level simulation, for the following reasons:

- The ALTMEMPHY is a calibrated interface. Therefore, gate-level simulation time can be very slow and take up to several hours to complete.
- Gate-level timing annotations, and the phase sweeping that the calibration uses, determine setup and hold violations. Because of the effect of X (unknown value) propagation within the atom simulation models, this action causes gate-level simulation failures in some cases.
- Memory interface timing methodology does not match the timing annotation that gate-level simulations use. Therefore the gate-level simulation does not accurately match real device behavior.

☞ Altera recommends that you validate the functional operation of your design via RTL simulation, and the timing of your design using TimeQuest Timing Analysis.

Before Simulating

In general, you need the following files to simulate:

- Library files from the <*Quartus II install path*>**\quartus\eda\sim_lib** directory:
  - **220model**
  - **altera_primitives**
  - **altera_mf**
  - **sgate**
  - **arriaii_atoms, stratixiv_atoms, stratixiii_atoms, cycloneiii_atoms, stratixii_atoms, stratixiigx_atoms** (device dependent)

  If you are targeting Stratix IV devices, you need both the Stratix IV and Stratix III files (**stratixiv_atoms** and **stratixiii_atoms**) to simulate, unless you are using NativeLink.

- Sequencer wrapper file (in **.vo** or **.vhd** format)
- PLL file (for example <*variation_name*>**_alt_mem_phy_pll.v** or **.vhd**)
- ALTMEMPHY modules (in the <*variation_name*>**_alt_mem_phy.v**)
- Top-level file
- User logic, or a driver for the PHY
- Testbench
- Vendor memory model

Preparing the Vendor Memory Model

If you are using a vendor memory model, instead of the MegaWizard-generated functional simulation model, you need to make some modifications to the vendor memory model and the testbench files by following these steps:

1. Make sure the IP functional simulation model is generated by turning on **Generate Simulation Model** during the instantiate PHY and controller design flow step.
2. Obtain and copy the vendor memory model to the **\testbench** directory. For example, obtain the **ddr2.v** and **ddr2_parameters.vh** simulation model files from the Micron website and save them in the testbench directory.

   ☞ The auto-generated generic SDRAM model may be used as a placeholder for a specific vendor memory model.

   ☞ Some vendor DIMM memory models do not use data mask (DM) pin operation, which can cause calibration failures. In these cases, use the vendor's component simulation models directly.

3. Open the vendor memory model file in a text editor and specify the speed grade and device width at the top of the file. For example, you can add the following statements for a DDR2 SDRAM model file:

   ```
   'define sg25

   'define x8
   ```

   The first statement specifies the memory device speed grade as –25 (for 400 MHz operation). The second statement specifies the memory device width per DQS.
4. Check that the following statement is included in the vendor memory model file. If not, include it at the top of the file. This example is for a DDR2 SDRAM model file:

   ```
   `include "ddr2_parameters.vh"
   ```

5. Save the vendor memory model file.
6. Open the testbench in a text editor, delete the whole section between the `START MEGAWIZARD INSERT MEMORY_ARRAY` and `END MEGAWIZARD INSERT MEMORY_ARRAY` comments, instantiate the downloaded memory model, and connect its signals to the rest of the design.
7. Delete the `START MEGAWIZARD INSERT MEMORY_ARRAY` and `END MEGAWIZARD INSERT MEMORY_ARRAY` lines so that the wizard does not overwrite your changes if you use the wizard to regenerate the design.
8. Ensure that ports names and capitalization in the memory model match the port names and capitalization in the testbench.

   ☞ The vendor memory model may use different pin names and capitalization than the MegaWizard-generated functional model.

9. Save the testbench file.

☞ Step 6 to step 9 is only valid for ALTMEMPHY design using Verilog HDL.

The original instantiation (from step 3 to step 9) may be similar to the following code:

```
// << START MEGAWIZARD INSERT MEMORY_ARRAY

    // This will need updating to match the memory models you are using.

// Instantiate a generated DDR memory model to match the datawidth &
chipselect requirements

ddr2_mem_model mem (

        .mem_dq      (mem_dq),

        .mem_dqs     (mem_dqs),

        .mem_dqs_n   (mem_dqs_n),

        .mem_addr    (a_delayed),

        .mem_ba      (ba_delayed),

        .mem_clk     (clk_to_ram),

        .mem_clk_n   (clk_to_ram_n),

        .mem_cke     (cke_delayed),

        .mem_cs_n    (cs_n_delayed),

        .mem_ras_n   (ras_n_delayed),

        .mem_cas_n   (cas_n_delayed),

        .mem_we_n    (we_n_delayed),

        .mem_dm      (dm_delayed),

        .mem_odt     (odt_delayed)

    );

// << END MEGAWIZARD INSERT MEMORY_ARRAY
```

Replace the original code with the following code:

```
// << START MEGAWIZARD INSERT MEMORY_ARRAY

    // This will need updating to match the memory models you are using.

// Instantiate a generated DDR memory model to match the datawidth &
chipselect requirements

ddr2 memory_0 (

    .clk     (clk_to_ram),

    .clk_n   (clk_to_ram_n),

    .cke     (cke_delayed),

    .cs_n    (cs_n_delayed),

    .ras_n   (ras_n_delayed),

    .cas_n   (cas_n_delayed),

    .we_n    (we_n_delayed),

    .dm_rdqs (dm_delayed[0]),

    .ba      (ba_delayed),

    .addr    (a_delayed),

    .dq      (mem_dq[7:0]),

    .dqs     (mem_dqs[0]),

    .dqs_n   (mem_dqs_n[0]),
```

```
.rdqs_n (),
.odt (odt_delayed)
);
// << END MEGAWIZARD INSERT MEMORY_ARRAY
```

If you are interfacing with a DIMM or multiple memory components, you need to instantiate all the memory components in the testbench file.

Simulating Using NativeLink

To set up simulation in the Quartus® II software using NativeLink, follow these steps:

1. Create a custom variation with an IP functional simulation model.
2. Set the top-level entity to the example project.
   a. On the File menu, click **Open**.
   b. Browse to <*variation name*>**_example_top** and click **Open**.
   c. On the Project menu, click **Set as Top-Level Entity**.
3. Ensure that the Quartus II **EDA Tool Options** are configured correctly for your simulation environment.
   a. On the Tools menu, click **Options**.
   b. In the **Category** list, click **EDA Tool Options** and verify the locations of the executable files.
4. Set up the Quartus II NativeLink.
   a. On the Assignments menu, click **Settings**. In the **Category** list, expand **EDA Tool Settings** and click **Simulation**.
   b. From the **Tool name** list, click on your preferred simulator.
   c. In **NativeLink settings**, select **Compile test bench** and click **Test Benches**.
   d. Click **New** at the **Test Benches** page to create a testbench.
5. On the **New Test Bench Settings** dialog box:
   a. Type a name for the **Test bench name**, for example <*variation name*>`_example_top_tb`.
   b. In **Top level module in test bench**, type the name of the automatically generated testbench, <*variation name*>`_example_top_tb`.
   c. In **Design instance in test bench**, type the name of the top-level instance, `dut`.
   d. Under **Simulation period**, set **End simulation at** to `600 µs`.
   e. Add the testbench files and automatically-generated memory model files. In the **File name** field, browse to the location of the memory model and the testbench, click **Open** and then click **Add**. The testbench is <*variation name*>**_example_top_tb.v**; memory model is <*variation name*>**_mem_model.v**.
   f. Select the files and click **OK**.

6. On the Processing menu, point to **Start** and click **Start Analysis & Elaboration** to start analysis.
7. On the Tools menu, point to **Run Simulation Tool** and click **RTL Simulation**.

If your Quartus II project appears to be configured correctly but the example testbench still fails, check the known issues on the Knowledge Database page of the Altera website before filing a service request.

For a complete MegaWizard Plug-In Manager system design example containing the DDR and DDR2 SDRAM Controller with ALTMEMPHY IP, refer to the design tutorials and design examples on the List of designs using Altera External Memory IP page of the Altera Wiki website.

IP Functional Simulations

This topic discusses VHDL and Verilog HDL simulations with IP functional simulation models.

VHDL

For VHDL simulations with IP functional simulation models, perform the following steps:

1. Create a directory in the <*project directory*>**\testbench** directory.
2. Launch your simulation tool from this directory and create the following libraries:
   - **altera_mf**
   - **lpm**
   - **sgate**
   - <*device name*>
   - **altera**
   - **ALTGXB**
   - <*device name*>**_hssi**
   - **auk_ddr_hp_user_lib**
3. Compile the files into the appropriate library (Table 10–2). The files are in VHDL93 format.

Table 10–2. Files to Compile—VHDL IP Functional Simulation Models (Part 1 of 2)

Library	File Name
altera_mf	<*QUARTUS ROOTDIR*>**\eda\sim_lib\altera_mf_components.vhd** <*QUARTUS ROOTDIR*>**\eda\sim_lib\altera_mf.vhd**
lpm	**\eda\sim_lib\220pack.vhd** **\eda\sim_lib\220model.vhd**
sgate	**eda\sim_lib\sgate_pack.vhd** **eda\sim_lib\sgate.vhd**

Table 10–2. Files to Compile—VHDL IP Functional Simulation Models (Part 2 of 2)

Library	File Name
<device name>	**eda\sim_lib***<*device name*>**_atoms.vhd** **eda\sim_lib***<*device name*>**_ components.vhd** **eda\sim_lib***<*device name*>**_hssi_atoms.vhd** (1)
altera	**eda\sim_lib\altera_primitives_components.vhd** **eda\sim_lib\altera_syn_attributes.vhd** **eda\sim_lib\altera_primitives.vhd**
ALTGXB (1)	*<device name>***_mf.vhd** *<device name>***_mf_components.vhd**
*<device name>***_hssi** (1)	*<device name>***_hssi_components.vhd** *<device name>***_hssi_atoms.vhd**
auk_ddr_hp_user_lib	*<QUARTUS ROOTDIR>* **libraries\vhdl\altera\altera_europa_support_lib.vhd**
	*<project directory>\<variation name>***_phy_alt_mem_phy_seq_wrapper.vho**
	*<project directory>\<variation name>***_phy.vho**
	*<project directory>\<variation name>***.vhd**
	*<project directory>\<variation name>***_example_top.vhd**
	*<project directory>\<variation name>***_controller_phy.vhd**
	*<project directory>\<variation name>***_phy_alt_mem_phy_pll.vhd**
	*<project directory>\<variation name>***_phy_alt_mem_phy_seq.vhd**
	*<project directory>\<variation name>***_example_driver.vhd**
	*<project directory>\<variation name>***_ex_lfsr8.vhd**
	testbench*<*variation name*>**_example_top_tb.vhd**
	testbench*<*variation name*>**_mem_model.vhd**
	all files in your project directory with *alt*ddrx* as part of their filenames

Note for Table 10–2:

(1) Applicable only for Arria II GX and Stratix IV devices.

If you are targeting a Stratix IV device, you need both the Stratix IV and Stratix III files (**stratixiv_atoms** and **stratixiii_atoms**) to simulate in your simulator, unless you are using NativeLink.

4. Load the testbench in your simulator with the timestep set to **picoseconds**.
5. Compile the testbench file.

Verilog HDL

For Verilog HDL simulations with IP functional simulation models, follow these steps:

1. Create a directory in the <*project directory*>**\testbench** directory.
2. Launch your simulation tool from this directory and create the following libraries:
   - **altera_mf_ver**
   - **lpm_ver**
   - **sgate_ver**
   - <*device name*>**_ver**
   - **altera_ver**
   - **ALTGXB_ver**
   - <*device name*>**_hssi_ver**
   - **auk_ddr_hp_user_lib**
3. Compile the files into the appropriate library as shown in Table 10–3 on page 10–23.

Table 10–3. Files to Compile—Verilog HDL IP Functional Simulation Models (Part 1 of 2)

Library	File Name
altera_mf_ver	<*QUARTUS ROOTDIR*>**\eda\sim_lib\altera_mf.v**
lpm_ver	**\eda\sim_lib\220model.v**
sgate_ver	**eda\sim_lib\sgate.v**
<*device name*>**_ver**	**eda\sim_lib**<*device name*>**_atoms.v** **eda\sim_lib**<*device name*>**_hssi_atoms.v** (1)
altera_ver	**eda\sim_lib\altera_primitives.v**
ALTGXB_ver (1)	<*device name*>**_mf.v**
<*device name*>**_hssi_ver** (1)	<*device name*>**_hssi_atoms.v**

Table 10–3. Files to Compile—Verilog HDL IP Functional Simulation Models (Part 2 of 2)

Library	File Name
auk_ddr_hp_user_lib	*<QUARTUS ROOTDIR>***libraries\vhdl\altera\altera_europa_support_lib.v**
	alt_mem_phy_defines.v
	*<project directory>**<variation name>***_phy_alt_mem_phy_seq_wrapper.vo**
	*<project directory>**<variation name>***.v**
	*<project directory>**<variation name>***_example_top.v**
	*<project directory>**<variation name>***_phy.v**
	*<project directory>**<variation name>***_controller_phy.v**
	*<project directory>**<variation name>***_phy_alt_mem_phy_pll.v**
	*<project directory>**<variation name>***_phy_alt_mem_phy.v**
	*<project directory>**<variation name>***_example_driver.v**
	*<project directory>**<variation name>***_ex_lfsr8.v**
	testbench*<variation name>***_example_top_tb.v**
	testbench*<variation name>***_mem_model.v**
	all files in your project directory with *alt*ddrx* as part of their filenames

Note for Table 10–3:

(1) Applicable only for Arria II GX and Stratix IV devices.

If you are targeting a Stratix IV device, you need both the Stratix IV and Stratix III files (**stratixiv_atoms** and **stratixiii_atoms**) to simulate in your simulator, unless you are using NativeLink

4. Configure your simulator to use transport delays, a timestep of **picoseconds**, and to include all the libraries in Table 10–3.
5. Compile the testbench file.

Simulation Tips and Issues

This topic discusses simulation tips and issues.

Tips

The ALTMEMPHY datapath is in Verilog HDL; the sequencer is in VHDL. For ALTMEMPHY designs with the AFI, to allow the Verilog HDL simulator to simulate the design after modifying the VHDL sequencer, follow these steps:

1. On the View menu, point to **Utility Windows,** and click **TCL console**.
2. Enter the following command in the console:

```
quartus_map --read_settings_file=on --write_settings_file=off --
source=<variation_name>_phy_alt_mem_phy_seq.vhd --
source=<variation_name>_phy_alt_mem_phy_seq_wrapper.v --simgen --
simgen_parameter=CBX_HDL_LANGUAGE=verilog
<variation_name>_phy_alt_mem_phy_seq_wrapper -c
<variation_name>_phy_alt_mem_phy_seq_wrapper
```

The Quartus II software regenerates *<variation_name>*_**phy_alt_mem_phy_seq_wrapper.vo** and uses this file when the simulation runs.

DDR3 SDRAM (without Leveling) Warnings and Errors

You may see the following warning and error messages with skip calibration and quick calibration simulation:

- `WARNING: 200 us is required before RST_N goes inactive`
- `WARNING: 500 us is required after RST_N goes inactive before CKE goes active`

If these warning messages appear, change the values of the two parameters (`tinit_tck` and `tinit_rst`) in the following files to match the parameters in *<variation_name>*_**phy_alt_mem_phy_seq_wrapper.v**:

- *<variation_name>*_**phy_alt_mem_phy_seq_wrapper.vo** or
- *<variation_name>*_**phy_alt_mem_phy_seq_wrapper.vho** files

You may see the following warning and error messages with full calibration simulation during write leveling, which you can ignore:

- `Warning: tWLS violation on DQS bit 0 positive edge. Indeterminate CK capture is possible`
- `Warning: tWLH violation on DQS bit 0 positive edge. Indeterminate CK capture is possible.`
- `ERROR: tDQSH violation on DQS bit 0`

You may see the following warning messages at time 0 (before reset) of simulation, which you can ignore:

- `Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).`
- `Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0`

You may see the following warning and error messages during reset, which you can ignore:

- `Error : clock switches from 0/1 to X (Unknown value) on DLL instance`
- `Warning : Duty Cycle violation DLL instance Warning: Input clock duty cycle violation.`

Document Revision History

Table 10–4 lists the revision history for this document.

Table 10–4. Document Revision History

Date	Version	Changes
November 2012	5.1	■ Changed chapter number from 9 to 10.
June 2012	5.0	■ Added path to simulation scripts for Riviera-PRO to Functional Simulations section. ■ Added simulation procedure for Riviera-PRO to Simulating the Example Design section. ■ Updated the Abstract PHY section. ■ Updated the Post-fit Functional Simulation procedure. ■ Added Feedback icon.
November 2011	4.0	■ Added the PHY-Only Simulation section. ■ Added the Post-fit Functional Simulation section. ■ Updated the Simulation Walkthrough with UniPHY IP section.
June 2011	3.0	■ Added an overview about memory simulation. ■ Added the Simulation Walkthrough with UniPHY IP section.
December 2010	2.1	Updated for 10.1 release.
July 2010	2.0	Updated for 10.0 release.
January 2010	1.1	Corrected minor typos.
November 2009	1.0	First published.

11. Analyzing Timing of Memory IP

Ensuring that your external memory interface meets the various timing requirements of today's high-speed memory devices can be a challenge. Altera addresses this challenge by offering external memory physical layer (PHY) interface IPs—ALTMEMPHY and UniPHY, which employ a combination of source-synchronous and self-calibrating circuits to maximize system timing margins. This PHY interface is a plug-and-play solution that the Quartus® II TimeQuest Timing Analyzer timing constrains and analyzes. The ALTMEMPHY and UniPHY IP, and the numerous device features offered by Arria® II, Arria V, Cyclone® III, Cyclone IV, Cyclone V, Stratix® III, Stratix IV, and Stratix V FPGAs, greatly simplifies the implementation of an external memory interface. All the information presented in this document for Stratix III and Stratix IV devices is applicable to HardCopy® III and HardCopy IV devices, respectively.

This chapter details the various timing paths that determine overall external memory interface performance, and describes the timing constraints and assumptions that the PHY IP uses to analyze these paths.

This chapter focuses on timing constraints for external memory interfaces based on the ALTMEMPHY and UniPHY IP. For information about timing constraints and analysis of external memory interfaces and other source-synchronous interfaces based on the ALTDQ_DQS and ALTDQ_DQS2 megafunctions, refer to *AN 433: Constraining and Analyzing Source-Synchronous Interfaces* and the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

External memory interface timing analysis is supported only by the TimeQuest Timing Analyzer, for the following reasons:

- The wizard-generated timing constraint scripts only support the TimeQuest analyzer.
- The Classic Timing Analyzer does not offer analysis of source-synchronous outputs. For example, write data, address, and command outputs.
- The Classic Timing Analyzer does not support detailed rise and fall delay analysis.

The performance of an FPGA interface to an external memory device is dependent on the following items:

- Read datapath timing
- Write datapath timing
- Address and command path timing
- Clock to strobe timing (t_{DQSS} in DDR and DDR2 SDRAM, and $t_{KHK\#H}$ in QDR II and QDRII+ SRAM)

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

- Read resynchronization path timing (applicable for DDR, DDR2, and DDR3 SDRAM in Arria II, Arria V, Stratix III, Stratix IV, and Stratix V devices)
- Read postamble path timing (applicable for DDR and DDR2 SDRAM in Stratix II devices)
- Write leveling path timing (applicable for DDR3 SDRAM with ALTMEMPHY and DDR2 and DDR3 SDRAM with UniPHY)
- PHY timing paths between I/O element and core registers
- PHY and controller internal timing paths (core f_{MAX} and reset recovery/removal)
- I/O toggle rate
- Output clock specifications
- Bus turnaround timing (applicable for RLDRAM II and DDR2 and DDR3 SDRAM with UniPHY)

☞ External memory interface performance depends on various timing components, and overall system level performance is limited by performance of the slowest link (that is, the path with the smallest timing margins).

Memory Interface Timing Components

There are several categories of memory interface timing components, including source-synchronous timing paths, calibrated timing paths, internal FPGA timing paths, and other FPGA timing parameters.

Understanding the nature of timing paths enables you to use an appropriate timing analysis methodology and constraints. The following section examines these aspects of memory interface timing paths.

Source-Synchronous Paths

These are timing paths where clock and data signals pass from the transmitting device to the receiving device.

An example of such a path is the FPGA-to-memory write datapath. The FPGA device transmits DQ output data signals to the memory along with a center-aligned DQS output strobe signal. The memory device uses the DQS signal to clock the data on the DQ pins into its internal registers.

☞ For brevity, the remainder of this chapter refers to data signals and strobe and clock signals as DQ signals and DQS signals, respectively. While the terminology is formally correct only for DDR-type interfaces and does not match QDR II, QDR II+ and RLDRAM II pin names, the behavior is similar enough that most timing properties and concepts apply to both. The clock that captures address and command signals is always referred to as CK/CK# too.

Calibrated Paths

These are timing paths where the clock used to capture data is dynamically positioned within the data valid window (DVW) to maximize timing margin.

For Arria II FPGAs interfacing with a DDR2 and DDR3 SDRAM controller with ALTMEMPHY IP, the resynchronization of read data from the DQS-based capture registers to the FPGA system clock domain is implemented using a self-calibrating circuit. On initialization, the sequencer block analyzes all path delays between the read capture and resynchronization registers to set up the resynchronization clock phase for optimal timing margin.

In Cyclone III and Cyclone IV FPGAs, the ALTMEMPHY IP performs the initial data capture from the memory device using a self-calibrating circuit. The ALTMEMPHY IP does not use the DQS strobes from the memory for capture; instead, it uses a dynamic PLL clock signal to capture DQ data signals into core LE registers.

For UniPHY-based controllers, the sequencer block analyzes all path delays between the read capture registers and the read FIFO buffer to set up the FIFO write clock phase for optimal timing margin. The read postamble calibration process is implemented in a similar manner to the read resynchonization calibration. In addition, the sequencer block calibrates a read data valid signal to the delay between a controller issuing a read command and read data returning to controller.

In DDR2 and DDR3 SDRAM and RLDRAM II with UniPHY, the UniPHY IP calibrates the write-leveling chains and programmable output delay chain to align the DQS edge with the CK edge at memory to meet the t_{DQSS}, t_{DSS}, and t_{DSH} specifications.

UniPHY IP enables the dynamic deskew calibration with NIOS sequencer for read and write paths. Dynamic deskew process uses the programmable delay chains that exist within the read and write data paths to adjust the delay of each DQ and DQS pin to remove the skew between different DQ signals and to centre-align the DQS strobe in the DVW of the DQ signals. This process occurs at power up for the read and the write paths.

Internal FPGA Timing Paths

Other timing paths that have an impact on memory interface timing include FPGA internal f_{MAX} paths for PHY and controller logic. This timing analysis is common to all FPGA designs. With appropriate timing constraints on the design (such as clock settings), the TimeQuest Timing Analyzer reports the corresponding timing margins.

For more information about the TimeQuest Timing Analyzer, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Other FPGA Timing Parameters

Some FPGA data sheet parameters, such as I/O toggle rate and output clock specifications, can limit memory interface performance.

I/O toggle rates vary based on speed grade, loading, and I/O bank location—top/bottom versus left/right. This toggle rate is also a function of the termination used (OCT or external termination) and other settings such as drive strength and slew rate.

Ensure you check the I/O performance in the overall system performance calculation. Altera recommends that you perform signal integrity analysis for the specified drive strength and output pin load combination.

For information about signal integrity, refer to the board design guidelines chapters and *AN 476: Impact of I/O Settings on Signal Integrity in Stratix III Devices*.

Output clock specifications include clock period jitter, half-period jitter, cycle-to-cycle jitter, and skew between FPGA clock outputs. You can obtain these specifications from the FPGA data sheet and must meet memory device requirements. You can use these specifications to determine the overall data valid window for signals transmitted between the memory and FPGA device.

FPGA Timing Paths

This topic describes the FPGA timing paths, the timing constraints examples, and the timing assumptions that the constraint scripts use.

In Arria II, Arria V, Stratix III, Stratix IV, and Stratix V devices, the interface margin is reported based on a combination of the TimeQuest Timing Analyzer and further steps to account for calibration that occurs at runtime. First the TimeQuest analyzer returns the base setup and hold slacks, and then further processing adjusts the slacks to account for effects which cannot be modeled in TimeQuest.

Arria II Device PHY Timing Paths

Table 11–1 lists all Arria II devices external memory interface timing paths.

Table 11–1. Arria II Devices External Memory Interface Timing Paths (1) (Part 1 of 2)

Timing Path	Circuit Category	Source	Destination
Read Data (2), (7)	Source-Synchronous	Memory DQ, DQS Pins	DQ Capture Registers in IOE
Write Data (2), (7)	Source-Synchronous	FPGA DQ, DQS Pins	Memory DQ, DM, and DQS Pins
Address and command (2)	Source-Synchronous	FPGA CK/CK# and Addr/Cmd Pins	Memory Input Pins
Clock-to-Strobe (2)	Source-Synchronous	FPGA CK/CK# and DQS Output Pins	Memory Input Pins
Read Resynchronization (2), (3)	Calibrated	IOE Capture Registers	IOE Resynchronization Registers
Read Resynchronization (2), (6)	Calibrated	IOE Capture Registers	Read FIFO in FPGA Core
PHY IOE-Core Paths (2), (3)	Source-Synchronous	IOE Resynchronization Registers	FIFO in FPGA Core
PHY and Controller Internal Paths (2)	Internal Clock f_{MAX}	Core Registers	Core Registers
I/O Toggle Rate (4)	I/O	FPGA Output Pin	Memory Input Pins

Table 11–1. Arria II Devices External Memory Interface Timing Paths (1) (Part 2 of 2)

Timing Path	Circuit Category	Source	Destination
Output Clock Specifications (Jitter, DCD) (5)	I/O	FPGA Output Pin	Memory Input Pins

Notes to Table 11–1:

(1) Timing paths applicable for an interface between Arria II devices and SDRAM component.
(2) Timing margins for this path are reported by the TimeQuest Timing Analyzer Report DDR function.
(3) Only for ALTMEMPHY megafunctions.
(4) Altera recommends that you perform signal integrity simulations to verify I/O toggle rate.
(5) For output clock specifications, refer to the *Arria II Device Data Sheet* chapter of the *Arria II Handbook*.
(6) Only for UniPHY IP.
(7) Arria II GX devices use source-synchronous and calibrated path.

Figure 11–1 shows the Arria II GX devices input datapath registers and circuit types.

UniPHY IP interfaces bypass the synchronization registers.

Figure 11–1. Arria II GX Devices Input Data Path Registers and Circuit Types in SDRAM Interface

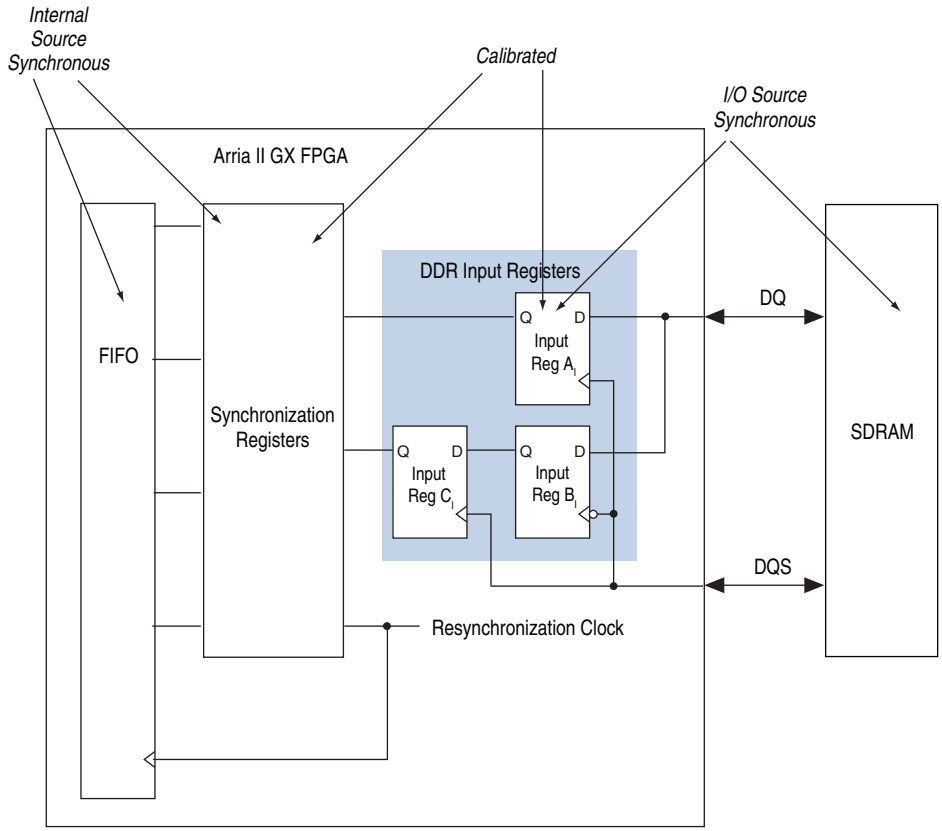
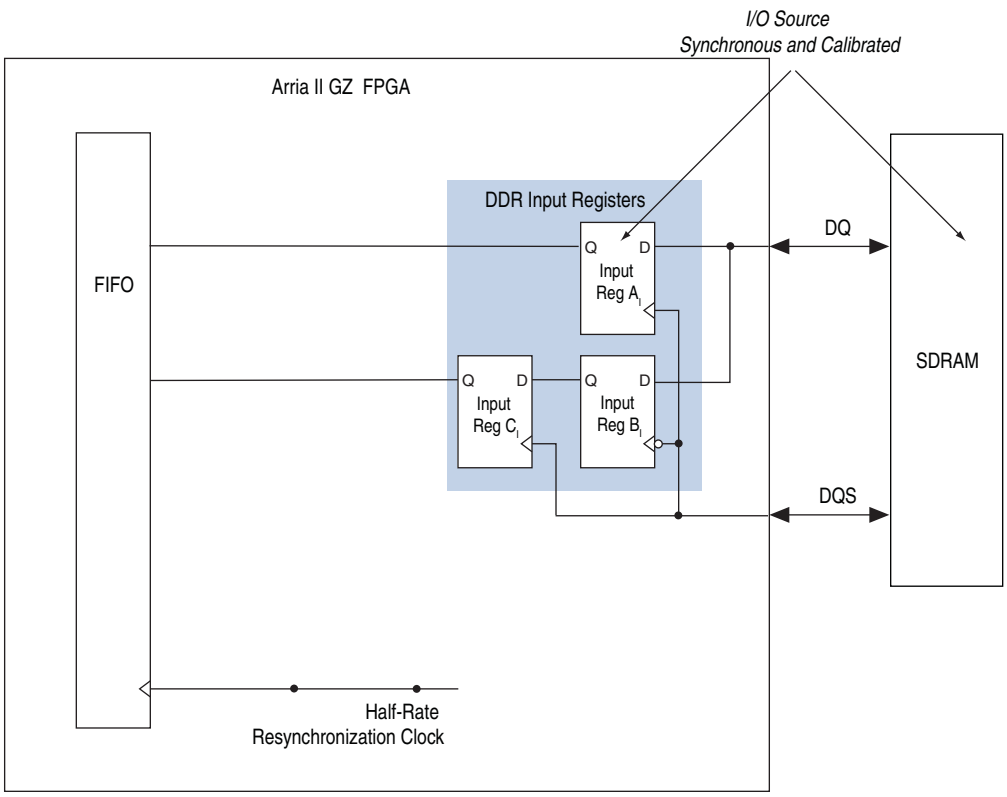


Figure 11–2 shows the Arria II GZ devices input datapath registers and circuit types.

Figure 11–2. Arria II GZ Devices Input Data Path Registers and Circuit Types in SDRAM Interface



Stratix III and Stratix IV PHY Timing Paths

A closer look at all the register transfers occurring in the Stratix III and Stratix IV input datapath reveals many source-synchronous and calibrated circuits.

The information in Figure 11–3 and Table 11–2 are based on Stratix IV devices, but they are applicable to Stratix III devices.

Figure 11–3 shows a block diagram of this input path with some of these paths identified for Stratix IV devices. The output datapath contains a similar set of circuits.

UniPHY IP interfaces bypass the alignment and synchronization registers.

Figure 11–3. Stratix IV Input Path Registers and Circuit Types in SDRAM Interface

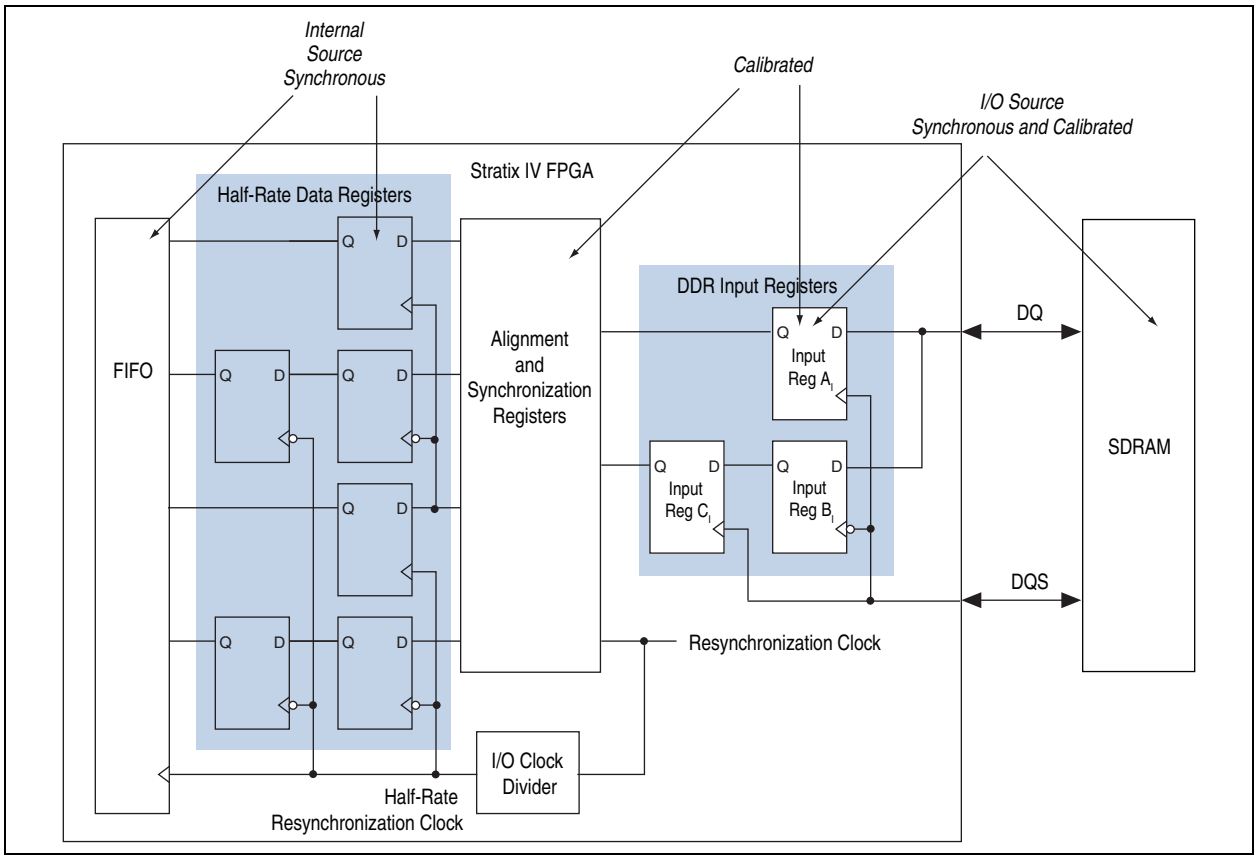


Table 11–2 lists the timing paths applicable for an interface between Stratix IV devices and half-rate SDRAM components.

The timing paths are also applicable to Stratix III devices, but Stratix III devices use only source-synchronous path for read and write data paths.

Table 11–2. Stratix IV External Memory Interface Timing Paths (Part 1 of 2)

Timing Path	Circuit Category	Source	Destination
Read Data (1)	Source-Synchronous and Calibrated	Memory DQ, DQS Pins	DQ Capture Registers in IOE
Write Data (1)	Source-Synchronous and Calibrated	FPGA DQ, DQS Pins	Memory DQ, DM, and DQS Pins
Address and command (1)	Source-Synchronous	FPGA CK/CK# and Addr/Cmd Pins	Memory Input Pins
Clock-to-Strobe (1)	Source-Synchronous	FPGA CK/CK# and DQS Output Pins	Memory Input Pins

Table 11–2. Stratix IV External Memory Interface Timing Paths (Part 2 of 2)

Timing Path	Circuit Category	Source	Destination
Read Resynchronization (1), (2)	Calibrated	IOE Capture Registers	IOE Alignment and Resynchronization Registers
Read Resynchronization (1), (5)	Calibrated	IOE Capture Registers	Read FIFO in FPGA Core
PHY IOE-Core Paths (1), (2)	Source-Synchronous	IOE Half Data Rate Registers and Half-Rate Resynchronization Clock	FIFO in FPGA Core
PHY & Controller Internal Paths (1)	Internal Clock f_{MAX}	Core registers	Core registers
I/O Toggle Rate (3)	I/O – Data sheet	FPGA Output Pin	Memory Input Pins
Output Clock Specifications (Jitter, DCD) (4)	I/O – Data sheet	FPGA Output Pin	Memory Input Pins

Notes to Table 11–2:

(1) Timing margins for this path are reported by the TimeQuest Timing Analyzer Report DDR function.
(2) Only for ALTMEMPHY megafunctions.
(3) Altera recommends that you perform signal integrity simulations to verify I/O toggle rate.
(4) For output clock specifications, refer to the *DC and Switching Characteristics* chapter of the *Stratix IV Device Handbook.*
(5) Only for UniPHY IP.

Arria V, Cyclone V, and Stratix V Timing paths

Figure 11–4 shows a block diagram of the Stratix V input data path.

Figure 11–4. Arria V, Cyclone V, and Stratix V Input Data Path

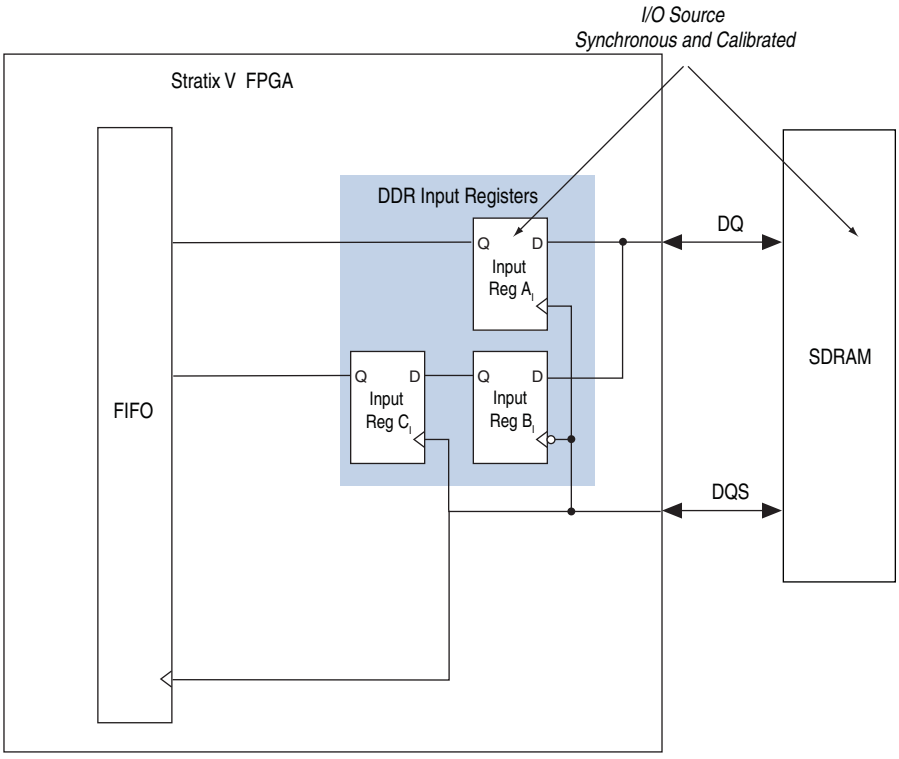


Table 11–3 lists all Stratix V devices external memory interface timing paths.

Table 11–3. Stratix V External Memory Interface Timing Paths (1)

Timing Path	Circuit Category	Source	Destination
Read Data (2)	Source-Synchronous and Calibrated	Memory DQ, DQS Pins	DQ Capture Registers in IOE
Write Data (2)	Source-Synchronous and Calibrated	FPGA DQ, DM, DQS Pins	Memory DQ, DM, and DQS Pins
Address and command (2)	Source-Synchronous	FPGA CK/CK# and Addr/Cmd Pins	Memory Input Pins
Clock-to-Strobe (2)	Source-Synchronous	FPGA CK/CK# and DQS Output Pins	Memory Input Pins
Read Resynchronization (2)	Source-Synchronous	IOE Capture Registers	Read FIFO in IOE
PHY & Controller Internal Paths (2)	Internal Clock f_{MAX}	Core Registers	Core Registers
i/O Toggle Rate (3)	I/O – Data sheet	FPGA Output Pin	Memory Input Pins
Output Clock Specifications (Jitter, DCD) (4)	I/O – Data sheet	FPGA Output Pin	Memory Input Pins

Notes to Table 11–3:

(1) This table lists the timing paths applicable for an interface between Arria V, Cyclone V, and Stratix V devices and half-rate SDRAM components.
(2) Timing margins for this path are reported by the TimeQuest Timing Analyzer Report DDR function.
(3) Altera recommends that you perform signal integrity simulations to verify I/O toggle rate.
(4) For output clock specifications, refer to the *DC and Switching Characteristics* chapter of the Stratix V Device Handbook.

Cyclone III and Cyclone IV PHY Timing Paths

Table 11–4 lists the various timing paths in a Cyclone III and Cyclone IV memory interface. Cyclone III and Cyclone IV devices use a calibrated PLL output clock for data capture and ignore the DQS strobe from the memory. Therefore, read resynchronization and postamble timing paths do not apply to Cyclone III and Cyclone IV designs. The read capture is implemented in LE registers specially placed next to the data pin with fixed routing, and data is transferred from the capture clock domain to the system clock domain using a FIFO block. Figure 11–5 shows the Cyclone III and Cyclone IV input datapath registers and circuit types.

Table 11–4. Cyclone III and Cyclone IV SDRAM External Memory Interface Timing Paths (1) **(Part 1 of 2)**

Timing Path	Circuit Category	Source	Destination
Read Data (2)	Calibrated	Memory DQ, DQS Pins	FPGA DQ Capture Registers in LEs
Write Data (2)	Source-Synchronous	FPGA DQ, DQS Pins	Memory DQ, DM, and DQS Pins
Address and command (2)	Source-Synchronous	FPGA CK/CK# and Addr/Cmd Pins	Memory Input Pins
Clock-to-Strobe (2)	Source-Synchronous	FPGA CK/CK# and DQS Output Pins	Memory Input Pins
PHY Internal Timing (2)	Internal Clock f_{MAX}	LE Half Data Rate Registers	FIFO in FPGA Core
I/O Toggle Rate (3)	I/O – Data sheet *I/O Timing* section	FPGA Output Pin	Memory Input Pins

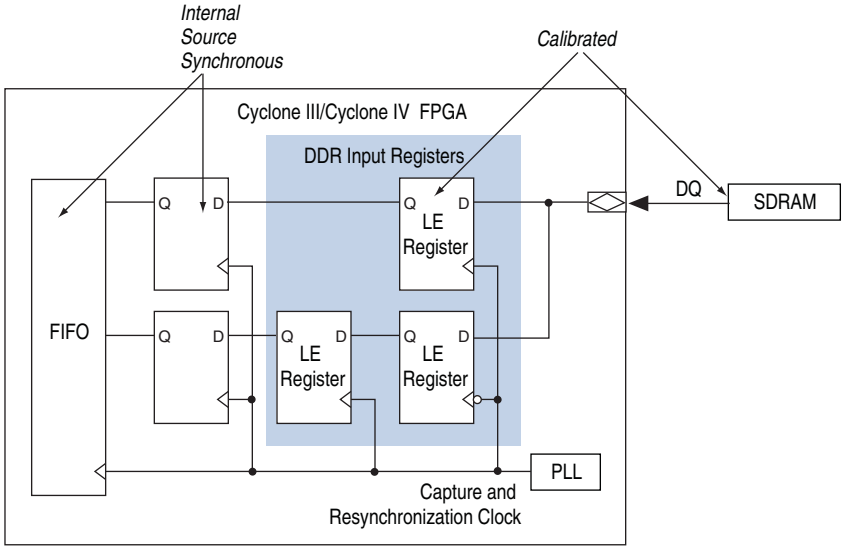
Table 11–4. Cyclone III and Cyclone IV SDRAM External Memory Interface Timing Paths (1) (Part 2 of 2)

Timing Path	Circuit Category	Source	Destination
Output Clock Specifications (Jitter, DCD) (4)	I/O – Data sheet *Switching Characteristics* section	FPGA Output Pin	Memory Input Pins

Notes to Table 11–4:

(1) Table 11–4 lists the timing paths applicable for an interface between Cyclone III and Cyclone IV devices and SDRAM.
(2) Timing margins for this path are reported by the TimeQuest Timing Analyzer Report DDR function.
(3) Altera recommends that you perform signal integrity simulations to verify I/O toggle rate.
(4) For output clock specifications, refer to the *DC and Switching Characteristics* chapter of the *Cyclone III Device Handbook* and of the *Cyclone IV Device Handbook*

Figure 11–5. Cyclone III or Cyclone IV Input Data Path Registers and Circuit Types in SDRAM Interface



Timing Constraint and Report Files

The timing contraints differ for the ALTMEMPHY megafunction and the UniPHY IP.

ALTMEMPHY Megafunction

To ensure a successful external memory interface operation, the ALTMEMPHY MegaWizard™ Plug-In Manager generates the following files for timing constraints and reporting scripts:

- <*variation_name*>**phy_ddr_timing.sdc**
- <*variation_name*>**phy_ddr_timing.tcl** (except Cyclone III devices)
- <*variation_name*>**phy_report_timing.tcl**
- <*variation_name*>**phy_report_timing_core.tcl** (except Cyclone III devices)
- <*variation_name*>**phy_ddr_pins.tcl**

*<variation_name>*_ddr_timing.sdc

The Synopsys Design Constraints File (**.sdc**) has the name <*controller_variation_name*>**_phy_ddr_timing.sdc** when you instantiate the ALTMEMPHY megafunction in the Altera® memory controller, and has the name <*phy_variation_name*>**_ddr_timing.sdc** when you instantiate the ALTMEMPHY megafunction as a stand-alone design.

To analyze the timing margins for all ALTMEMPHY megafunction timing paths, execute the Report DDR function in the TimeQuest Timing Analyzer; refer to the "Timing Analysis Description" on page 11–13. No timing constraints are necessary (or specified in the **.sdc**) for Arria II GX devices read capture and write datapaths, because all DQ and DQS pins are predefined. The capture and output registers are built into the IOE, and the signals are using dedicated routing connections. Timing constraints have no impact on the read and write timing margins. However, the timing margins for these paths are analyzed using FPGA data sheet specifications and the user-specified memory data sheet parameters.

The ALTMEMPHY megafunction uses the following **.sdc** constraints for internal FPGA timing paths, address and command paths, and clock-to-strobe timing paths:

- Creating clocks on PLL inputs
- Creating generated clocks using `derive_pll_clocks`, which includes all full-rate and half-rate PLL outputs, PLL reconfiguration clock, and I/O scan clocks
- Calling `derive_clock_uncertainty`
- Cutting timing paths for DDR I/O, calibrated paths, and most reset paths
- Setting output delays on address and command outputs (versus CK/CK# outputs)
- Setting `2T` or two clock-period multicycle setup for all half-rate address and command outputs, except nCS and on-die termination (ODT) (versus CK/CK# outputs)
- Setting output delays on DQS strobe outputs (versus CK/CK# outputs for DDR2 and DDR SDRAM)

The high-performance controller MegaWizard Plug-In Manager generates an extra <*variation_name*>**_example_top.sdc** for the example driver design. This file contains the timing constraints for the non-DDR specific parts of the project.

*<variation_name>*_ddr_timing.tcl

This script includes the memory interface and FPGA device timing parameters for your variation. It is included within <*variation_name*>**_report_timing.tcl** and <*variation_name*>**_ddr_timing.sdc** and runs automatically during compilation. This script is run for every instance of the same variation. Cyclone III devices do not have this **.tcl** file. All the parameters are in the **.sdc**.

*<variation_name>*_report_timing.tcl

This script reports the timing slacks for your variation. It runs automatically during compilation. You can also run this script with the Report DDR task in the TimeQuest Timing Analyzer window. This script is run for every instance of the same variation.

*<variation_name>*_report_timing_core.tcl

This script contains high-level procedures that <*variation_name*>**_report_timing.tcl** uses to compute the timing slacks for your variation. It runs automatically during compilation. Cyclone III devices do not have this **.tcl** file.

*<variation_name>*_ddr_pins.tcl

This script includes all the functions and procedures required by the <*variation_name*>**_report_timing.tcl** and <*variation_name*>**_ddr_timing.sdc** scripts. It is a library of useful functions to include at the top of an **.sdc**. It finds all the variation instances in the design and the associated clock, register, and pin names of each instances. The results are saved in the same directory as the **.sdc** and <*variation_name*>**_report_timing.tcl** as <*variation_name*>**_autodetectedpins.tcl**.

Because this **.tcl** file traverses the design for the project pin names, you do not need to keep the same port names on the top level of the design.

UniPHY IP

To ensure a successful external memory interface operation, the UniPHY IP generates two sets of files for timing constraints but in different folders and with slightly different filenames. One set of files are used for synthesis project, which is available under the <variation_name> folder located in the main project folder while the other set of files are the example designs, located in the <variation_name>example design\example_project folder.

The project folders contain the following files for timing constraints and reporting scripts:

- <*variation_name*>**.sdc**
- <*variation_name*>**_timing.tcl**
- <*variation_name*>**_report_timing.tcl**
- <*variation_name*>**_report_timing_core.tcl**
- <*variation_name*>**_pin_map.tcl**
- <*variation_name*>**_parameters.tcl**

<variation_name>.sdc

The <*variation_name*>**.sdc** is listed in the wizard-generated Quartus II IP File (**.qip)**. Including this file in the project allows the Quartus II Synthesis and Fitter to use the timing driven compilation to optimize the timing margins.

To analyze the timing margins for all UniPHY timing paths, execute the Report DDR function in the TimeQuest Timing Analyzer.

The UniPHY IP uses the **.sdc** to constrain internal FPGA timing paths, address and command paths, and clock-to-strobe timing paths, and more specifically:

- Creating clocks on PLL inputs
- Creating generated clocks
- Calling `derive_clock_uncertainty`

- Cutting timing paths for specific reset paths
- Setting input and output delays on DQ inputs and outputs
- Setting output delays on address and command outputs (versus CK/CK# outputs)

*<variation_name>*_timing.tcl

This script includes the memory, FPGA, and board timing parameters for your variation. It is included within <*variation_name*>**_report_timing.tcl** and <*variation_name*>**.sdc**. In multiple interface designs with PLL and DLL sharing, you must change the master core name and instance name in this file for the slave controller.

*<variation_name>*_report_timing.tcl

This script reports the timing slack for your variation. It runs automatically during compilation (during static timing analysis). You can also run this script with the Report DDR task in the TimeQuest Timing Analyzer. This script is run for every instance of the same variation.

*<variation_name>*_report_timing_core.tcl

This script contains high-level procedures that the <*variation_name*>**_report_timing.tcl** script uses to compute the timing slack for your variation. This script runs automatically during compilation.

*<variation_name>*_pin_map.tcl

This script is a library of functions and procedures that the <*variation_name*>**_report_timing.tcl** and <*variation_name*>**.sdc** scripts use. The <*variation_name*>**_pin_assignments.tcl** script, which is not relevant to timing constraints, also uses this library.

*<variation_name>*_parameters.tcl

This script defines some of the parameters that describe the geometry of the core and the PLL configuration. Do not change this file, except when you modify the PLL through the MegaWizard Plug-In Manager. In this case, the changes to the PLL parameters do not automatically propagate to this file and you must manually apply those changes in this file.

Timing Analysis Description

The following sections describe the timing analysis using the respective FPGA data sheet specifications and the user-specified memory data sheet parameters.

For detailed timing analysis description, refer to the scripts listed in "Timing Constraint and Report Files" on page 11–10.

To account for the effects of calibration, the ALTMEMPHY and UniPHY IP include additional scripts that are part of the <*phy_variation_name*>**_report_timing.tcl** and <*phy_variation_name*>**_ report_timing_core.tcl** files that determine the timing margin after calibration. These scripts use the setup and hold slacks of individual pins to emulate what is occurring during calibration to obtain timing margins that are

representative of calibrated PHYs. The effects considered as part of the calibrated timing analysis include improvements in margin because of calibration, and quantization error and calibration uncertainty because of voltage and temperature changes after calibration. The calibration effects do not apply to Stratix III and Cyclone III devices.

Address and Command

Address and command signals are single data rate signals latched by the memory device using the FPGA output clock. Some of the address and command signals are half-rate data signals, while others, such as the chip select, are full-rate signals. The TimeQuest Timing Analyzer analyzes the address and command timing paths using the `set_output_delay (max and min)` constraints.

PHY or Core

Timing analysis of the PHY or core path includes the path of soft registers in the device and the register in the I/O element. However, the analysis does not include the paths through the pin or the calibrated path. The PHY or core analyzes this path by calling the `report_timing` command in <*variation_name*>**_report_timing.tcl** and <*variation_name*>**_report_timing_core.tcl**.

PHY or Core Reset

The PHY or core reset is the internal timing of the asynchronous reset signals to the ALTMEMPHY or UniPHY IPs. The PHY or core analyzes this path by calling the `report_timing` command in <*variation_name*>**_report_timing.tcl** and <*variation_name*>**_report_timing_core.tcl**.

Read Capture and Write

Cyclone III and Stratix III memory interface designs perform read capture and write timing analysis using the TCCS and SW timing specification. Read capture and write timing analysis for Arria II, Cyclone IV, Stratix IV, and Stratix V memory interface designs are based on the timing slacks obtained from the TimeQuest Timing Analyzer and all the effects included with the Quartus II timing model such as die-to-die and within-die variations, aging, systematic skew, and operating condition variations. Because the PHY IP adjusts the timing slacks to account for the calibration effects, there are two sets of read capture and write timing analysis numbers—**Before Calibration** and **After Calibration**.

Cyclone III and Stratix III

This section details the timing margins, such as the read data and write data timing paths, which the TimeQuest Timing Analyzer callates for Cyclone III and Stratix III designs. Timing paths internal to the FPGA are either guaranteed by design and tested on silicon, or analyzed by the TimeQuest Timing Analyzer using corresponding timing constraints.

For design guidelines about implementing and analyzing your external memory interface using the PHY in Cyclone III, Stratix III, and Stratix IV devices, refer to the design tutorials on the List of designs using Altera External Memory IP page of the Altera Wiki website.

Timing margins for chip-to-chip data transfers can be defined as:

Margin = bit period – transmitter uncertainties – receiver requirements

where:

- Sum of all transmitter uncertainties = transmitter channel-to-channel skew (TCCS).

  The timing difference between the fastest and slowest output edges on data signals, including t_{CO} variation, clock skew, and jitter. The clock is included in the TCCS measurement and serves as the time reference.

- Sum of all receiver requirements = receiver sampling window (SW) requirement.

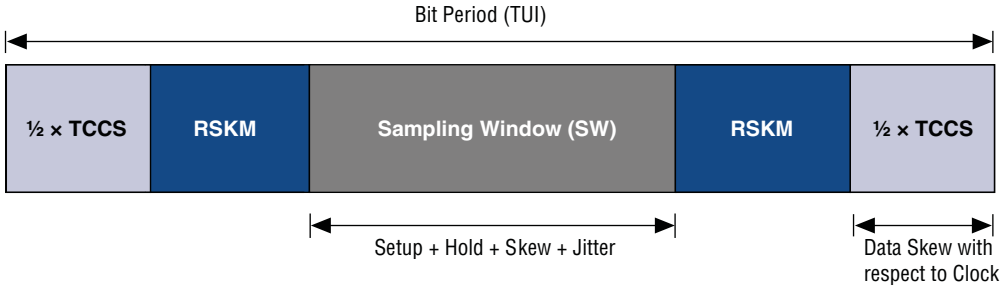
  The period of time during which the data must be valid to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window.

- Receiver skew margin (RSKM) = margin or slack at the receiver capture register.

For TCCS and SW specifications, refer to the *DC and Switching Characteristics* chapter of the *Cyclone III Device Handbook* or *Stratix III Device Handbook*.

Figure 11–6 relates this terminology to a timing budget diagram.

Figure 11–6. Sample Timing Budget Diagram



The timing budget regions marked "½ × TCCS" represent the latest data valid time and earliest data invalid times for the data transmitter. The region marked sampling window is the time required by the receiver during which data must stay stable. This sampling window comprises the following:

- Internal register setup and hold requirements
- Skew on the data and clock nets within the receiver device
- Jitter and uncertainty on the internal capture clock

The sampling window is not the capture margin or slack, but instead the requirement from the receiver. The margin available is denoted as RSKM.

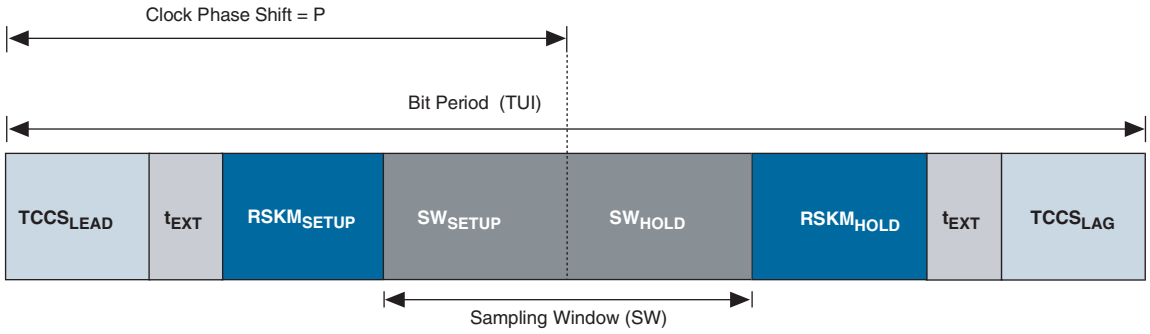
The simple example illustrated in Figure 11–6 does not consider any board level uncertainties, assumes a center-aligned capture clock at the middle of the receiver sampling window region, and assumes an evenly distributed TCCS with respect to the transmitter clock pin. In this example, the left end of the bit period corresponds to time t = 0, and the right end of the bit period corresponds to time t = TUI (where TUI stands for time unit interval). Therefore, the center-aligned capture clock at the receiver is best placed at time t = TUI/2.

Therefore:

$$\text{the total margin} = 2 \times \text{RSKM} = \text{TUI} - \text{TCCS} - \text{SW}.$$

Consider the case where the clock is not center-aligned within the bit period (clock phase shift = P), and the transmitter uncertainties are unbalanced ($TCCS_{LEAD} \neq$ and $TCCS_{LAG}$). $TCCS_{LEAD}$ is defined as the skew between the clock signal and latest data valid signal. $TCCS_{LAG}$ is defined as the skew between the clock signal and earliest data invalid signal. Also, the board level skew across data and clock traces are specified as t_{EXT}. For this condition, you should compute independent setup and hold margins at the receiver ($RSKM_{SETUP}$ and $RSKM_{HOLD}$). In this example, the sampling window requirement is split into a setup side requirement (SW_{SETUP}) and hold side (SW_{HOLD}) requirement. Figure 11–7 illustrates the timing budget for this condition. A timing budget similar to that shown in Figure 11–7 is used for Cyclone III and Stratix III FPGA read and write data timing paths.

Figure 11–7. Sample Timing Budget with Unbalanced (TCCS and SW) Timing Parameters



Therefore:

$$\text{Setup margin} = \text{RSKM}_{SETUP} = P - TCCS_{LEAD} - SW_{SETUP} - t_{EXT}$$

$$\text{Hold margin} = \text{RSKM}_{HOLD} = (TUI - P) - TCCS_{LAG} - SW_{HOLD} - t_{EXT}$$

The timing budget illustrated in Figure 11–6 with balanced timing parameters applies for calibrated paths where the clock is dynamically center-aligned within the data valid window. The timing budget illustrated in Figure 11–7 with unbalanced timing parameters applies for circuits that employ a static phase shift using a DLL or PLL to place the clock within the data valid window.

Read Capture

Memory devices provide edge-aligned DQ and DQS outputs to the FPGA during read operations. Stratix III FPGAs center-aligns the DQS strobe using static DLL-based delays, and the Cyclone III FPGAs use a calibrated PLL clock output to capture the read data in LE registers without using DQS. While Stratix III devices use a source synchronous circuit for data capture and Cyclone III devices use a calibrated circuit, the timing analysis methodology is quite similar, as shown in the following section.

When applying this methodology to read data timing, the memory device is the transmitter and the FPGA device is the receiver.

The transmitter channel-to-channel skew on outputs from the memory device is available from the corresponding device data sheet. Let us examine the TCCS parameters for a DDR2 SDRAM component.

For DQS-based capture:

- The time between DQS strobe and latest data valid is defined as t_{DQSQ}
- The time between earliest data invalid and next strobe is defined as t_{QHS}
- Based on earlier definitions, $TCCS_{LEAD} = t_{DQSQ}$ and $T_{CCSLAG} = t_{QHS}$

The sampling window at the receiver, the FPGA, includes several timing parameters:

- Capture register micro setup and micro hold time requirements
- DQS clock uncertainties because of DLL phase shift error and phase jitter
- Clock skew across the DQS bus feeding DQ capture registers
- Data skew on DQ paths from pin to input register including package skew

For TCCS and SW specifications, refer to the *DC and Switching Characteristics* chapter of the *Cyclone III Device Handbook* or the *Stratix III Device Handbook*.

Figure 11–8 shows the timing budget for a read data timing path.

Figure 11–8. Timing Budget for Read Data Timing Path

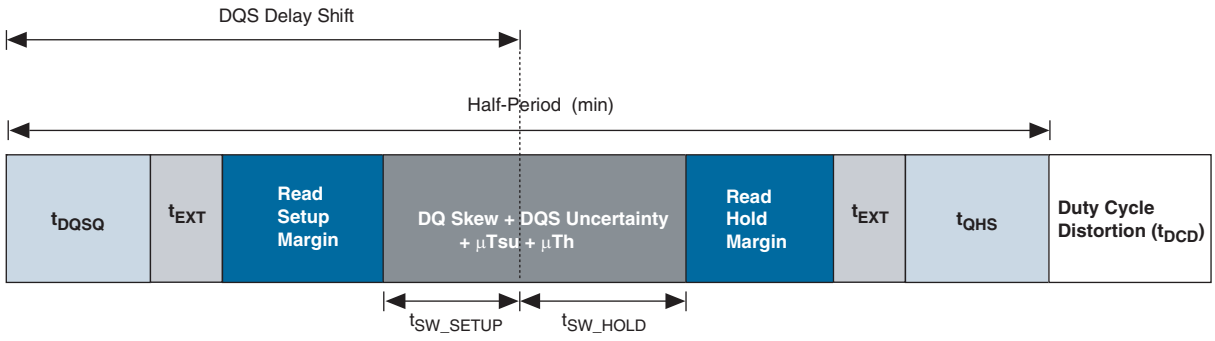


Table 11–5 lists a read data timing analysis for a Stratix III –2 speed-grade device interfacing with a 400-MHz DDR2 SDRAM component.

Table 11–5. Read Data Timing Analysis for Stratix III Device with a 400-MHz DDR2 SDRAM (1)

Parameter	Specifications	Value (ps)	Description
Memory Specifications (1)	t_{HP}	1250	Average half period as specified by the memory data sheet, t_{HP} = 1/2 * t_{CK}
	t_{DCD}	50	Duty cycle distortion = 2% × t_{CK} = 0.02 × 2500 ps
	t_{DQSQ}	200	Skew between DQS and DQ from memory
	t_{QHS}	300	Data hold skew factor as specified by memory
FPGA Specifications	t_{SW_SETUP}	181	FPGA sampling window specifications for a given configuration (DLL mode, width, location, and so on.)
	t_{SW_HOLD}	306	
Board Specifications	t_{EXT}	20	Maximum board trace variation allowed between any two signal traces (user specified parameter)
Timing Calculations	t_{DVW}	710	$t_{HP} – t_{DCD} – t_{DQSQ} – t_{QHS} – 2 × t_{EXT}$
	$t_{DQS_PHASE_DELAY}$	500	Ideal phase shift delay on DQS capture strobe = (DLL phase resolution × number of delay stages × t_{CK}) / 360° = (36° × 2 stages × 2500 ps)/360° = 500 ps
Results	Setup margin	99	$RSKM_{SETUP} = t_{DQSQ_PHASE_DELAY} – t_{DQSQ} – t_{SW_SETUP} – t_{EXT}$
	Hold margin	74	$RSKM_{HOLD} = t_{HP} – t_{DCD} – t_{DQS_PHASE_DELAY} – t_{QHS} – t_{SW_HOLD} – t_{EXT}$

Notes to Table 11–5:

(1) This sample calculation uses memory timing parameters from a 72-bit wide 256-MB micron MT9HTF3272AY-80E 400-MHz DDR2 SDRAM DIMM.

Table 11–6 lists a read data timing analysis for a DDR2 SDRAM component at 200 MHz using the SSTL-18 Class I I/O standard and termination. A 267-MHz DDR2 SDRAM component is required to ensure positive timing margins for the 200-MHz memory interface clock frequency for the 200 MHz operation.

Table 11–6. Read Data Timing Analysis for a 200-MHz DDR2 SDRAM on a Cyclone III Device (1)

Parameter	Specifications	Value (ps)	Description
Memory Specifications (1)	t_{HP}	2500	Average half period as specified by the memory data sheet
	t_{DCD_TOTAL}	250	Duty cycle distortion = 2% × tCK = 0.02 × 5000 ps
	t_{AC}	± 500	Data (DQ) output access time for a 267-MHz DDR2 SDRAM component
FPGA Specifications	t_{SW_SETUP}	580	FPGA sampling window specification for a given configuration (interface width, location, and so on).
	t_{SW_HOLD}	550	
Board Specifications (1)	t_{EXT}	20	Maximum board trace variation allowed between any two signal traces (user specified parameter)
Timing Calculations	t_{DVW}	1230	t_{HP} - t_{DCD} - 2 × t_{AC} – 2 × t_{EXT}
Results	Total margin	100	t_{DVW} - t_{SW_SETUP} - t_{SW_HOLD}

Notes to Table 11–6:

(1) For this sample calculation, total duty cycle distortion and board skew are split over both setup and hold margin. For more information on Cyclone III –6 speed-grade device read capture and timing analysis, refer to "Cyclone III and Cyclone IV PHY Timing Paths" on page 11–9.

Write Capture

During write operations, the FPGA generates a DQS strobe and a center-aligned DQ data bus using multiple PLL-driven clock outputs. The memory device receives these signals and captures them internally. The Stratix III family contains dedicated DDIO (double data rate I/O) blocks inside the IOEs.

For write operations, the FPGA device is the transmitter and the memory device is the receiver. The memory device's data sheet specifies data setup and data hold time requirements based on the input slew rate on the DQ/DQS pins. These requirements make up the memory sampling window, and include all timing uncertainties internal to the memory.

Output skew across the DQ and DQS output pins on the FPGA make up the TCCS specification. TCCS includes contributions from numerous internal FPGA circuits, including:

- Location of the DQ and DQS output pins
- Width of the DQ group
- PLL clock uncertainties, including phase jitter between different output taps used to center-align DQS with respect to DQ
- Clock skew across the DQ output pins, and between DQ and DQS output pins
- Package skew on DQ and DQS output pins

Refer to the *DC and Switching Characteristics* chapter of the *Cyclone III Device Handbook* or the *Stratix III Device Handbook* for TCCS and SW specifications.

Figure 11–9 illustrates the timing budget for a write data timing path.

Figure 11–9. Timing Budget for Write Data Timing Path

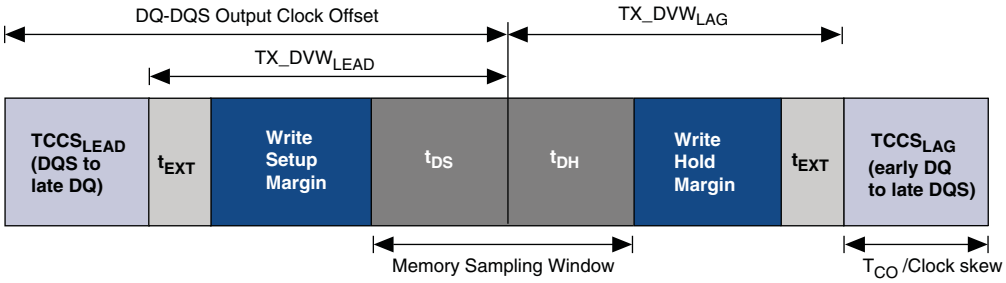


Table 11–7 lists a write data timing analysis for a Stratix III –2 speed-grade device interfacing with a DDR2 SDRAM component at 400 MHz. This timing analysis assumes the use of a differential DQS strobe with 2.0-V/ns edge rates on DQS, and 1.0-V/ns edge rate on DQ output pins. Consult your memory device's data sheet for derated setup and hold requirements based on the DQ/DQS output edge rates from your FPGA.

Table 11–7. Write Data Timing Analysis for 400-MHz DDR2 SDRAM Stratix III Device (1)

Parameter	Specifications	Value (ps)	Description
Memory Specifications (1)	t_{HP}	1250	Average half period as specified by the memory data sheet
	t_{DSA}	250	Memory setup requirement (derated for DQ/DQS edge rates and V_{REF} reference voltage)
	t_{DHA}	250	Memory hold requirement (derated for DQ/DQS edge rates and V_{REF} reference voltage)
FPGA Specifications	$TCCS_{LEAD}$	229	FPGA transmitter channel-to-channel skew for a given configuration (PLL setting, location, and width).
	$TCCS_{LAG}$	246	
Board Specifications	t_{EXT}	20	Maximum board trace variation allowed between any two signal traces (user specified parameter)
Timing Calculations	$t_{OUTPUT_CLOCK_OFFSET}$	625	Output clock phase offset between DQ & DQS output clocks = 90°. $t_{OUTPUT_CLOCK_OFFSET}$ = (output clock phase DQ and DQS offset x t_{CK})/360° = (90° x 2500)/360° = 625
	TX_DVW_{LEAD}	396	Transmitter data valid window = $t_{OUTPUT_CLOCK_OFFSET} – TCCS_{LEAD}$
	TX_DVW_{LAG}	379	Transmitter data valid window = tHP - $t_{OUTPUT_CLOCK_OFFSET} – TCCS_{LAG}$
Results	Setup margin	126	$TX_DVW_{LEAD} – t_{EXT} – t_{DSA}$
	Hold margin	109	$TX_DVW_{LAG} – t_{EXT} – t_{DHA}$

Notes to Table 11–7:

(1) This sample calculation uses memory timing parameters from a 72-bit wide 256-MB micron MT9HTF3272AY-80E 400-MHz DDR2 SDRAM DIMM

Table 11–8 lists a write timing analysis for a Cyclone III –6 speed-grade device interfacing with a DDR2 SDRAM component at 200 MHz. A 267-MHz DDR2 SDRAM component is used for this analysis.

Table 11–8. Write Data Timing Analysis for a 200-MHz DDR2 SDRAM Interface on a Cyclone III Device (1) (Part 1 of 2)

Parameter	Specifications	Value (ps)	Description
Memory Specifications	t_{HP}	2500	Average half period as specified by the memory data sheet
	t_{DCD_TOTAL}	250	Total duty cycle distortion = 5% × t_{CK} = 0.05 x 5000
	$t_{DS\ (derated)}$	395	Memory setup requirement from a 267-MHz DDR2 SDRAM component (derated for single-ended DQS and 1 V/ns slew rate)
	$t_{DH\ (derated)}$	335	Memory hold from DDR2 267-MHz component (derated for single-ended DQS and 1 V/ns slew rate)
FPGA Specifications	$TCCS_{LEAD}$	790	FPGA TCCS for a given configuration (PLL setting, location, width)
	$TCCS_{LAG}$	380	
Board Specifications	t_{EXT}	20	Maximum board trace variation allowed between any two signal traces (user specified parameter)

Table 11–8. Write Data Timing Analysis for a 200-MHz DDR2 SDRAM Interface on a Cyclone III Device (1) (Part 2 of 2)

Parameter	Specifications	Value (ps)	Description
Timing Calculations	TX_DVW_{LEAD}	460	Transmitter data valid window = $t_{OUTPUT_CLOCK_OFFSET} - TCCS_{LEAD}$
	TX_DVW_{LAG}	870	Transmitter data valid window = t_{HP} - $t_{OUTPUT_CLOCK_OFFSET} - TCCS_{LAG}$
	$t_{OUTPUT_CLOCK_OFFSET}$	1250	Output clock phase offset between DQ/DQS output clocks = 90° $t_{OUTPUT_CLOCK_OFFSET}$ = (output clock phase DQ & DQS offset x t_{CK})/360° = (90° x 5000)/360° = 1250
Results	Setup margin	45	$TX_DVW_{LEAD} - t_{EXT} - t_{DS}$
	Hold margin	265	$TX_DVW_{LAG} - t_{EXT} - t_{DH} - t_{DCD_TOTAL}$

Note to Table 11–8:

(1) For more information on Cyclone III –6 speed-grade device read capture and timing analysis, refer to "Read Capture" on page 11–17.

Arria II, Arria V, Cyclone IV, Cyclone V, Stratix IV and Stratix V

Read Capture

Read capture timing analysis indicates the amount of slack on the DDR DQ signals that are latched by the FPGA using the DQS strobe output of the memory device. The read capture timing paths are analyzed by a combination of the TimeQuest Timing Analyzer using the `set_input_delay (max and min)`, `set_max_delay`, and `set_min_delay` constraints, and further steps to account for calibration that occurs at runtime. The ALTMEMPHY and UniPHY IP include timing constraints in the <*phy_variation_name*>**_ddr_timing.sdc** (ALTMEMPHY) or <*phy_variation_name*>**.sdc** (UniPHY), and further slack analysis in <*phy_variation_name*>**_report_timing.tcl** and <*phy_variation_name*>**_report_timing_core.tcl** files.

The PHY IP captures the Cyclone IV devices read data using a PLL phase that is calibrated and tracked with the sequencer. The equations in <*phy_variation_name*>**_report_timing_core.tcl** ensures optimum read capture timing margin.

In Arria II, Cyclone IV, and Stratix IV devices, the margin is reported based on a combination of the TimeQuest Timing Analyzer calculation results and further processing steps that account for the calibration that occurs at runtime. First, the TimeQuest analyzer returns the base setup and hold slacks, and further processing steps adjust the slacks to account for effects which the TimeQuest analyzer cannot model.

Write

Write timing analysis indicates the amount of slack on the DDR DQ signals that are latched by the memory device using the DQS strobe output from the FPGA device. The write timing paths are analyzed by a combination of the TimeQuest Timing Analyzer using the `set_output_delay (max and min)` and further steps to account for calibration that occurs at runtime. The ALTMEMPHY and UniPHY IP include timing constraints in the <*phy_variation_name*>**_ddr_timing.sdc** (ALTMEMPHY) or <*phy_variation_name*>**.sdc** (UniPHY), and further slack analysis in <*phy_variation_name*>**_report_timing.tcl** and <*phy_variation_name*>**_report_timing_core.tcl** files.

Read Resynchronization

In the DDR3, DDR2, and DDR SDRAM interfaces with Arria II GX FPGAs, the resynchronization timing analysis concerns transferring read data that is captured with a DQS strobe to a clock domain under the control of the ALTMEMPHY. After calibration by a sequencer, a dedicated PLL phase tracks any movements in the data valid window of the captured data. The exact length of the DQS and CK traces does not affect the timing analysis. The calibration process centers the resynchronization clock phase in the middle of the captured data valid window to maximize the resynchronization setup and hold the margin, and removes any static offset from other timing paths. With the static offset removed, any remaining uncertainties are voltage and temperature variation, jitter and skew.

In a UniPHY interface, a FIFO buffer synchronizes the data transfer from the data capture to the core. The calibration process sets the depth of the FIFO buffer and no dedicated synchronization clock is required. Refer to *<phy_variation_name>*_**report_timing_core.tcl** for more information about the resynchronization timing margin equation.

Mimic Path

The mimic path mimics the FPGA portion of the elements of the round-trip delay, which enables the calibration sequencer to track delay variations because of voltage and temperature changes during the memory read and write transactions without interrupting the operation of the ALTMEMPHY megafunction.

As the timing path register is integrated in the IOE, there is no timing constraint required for the Arria II GX device families.

For Cyclone III and Cyclone IV devices, the mimic register is a register in the core and it is placed closer to the IOE by the fitter.

The UniPHY IP does not use any mimic path.

DQS versus CK—Arria II GX, Cyclone III, and Cyclone IV Devices

The DQS versus CK timing path indicates the skew requirement for the arrival time of the DQS strobe at the memory with respect to the arrival time of CK/CK# at the memory. Arria II GX, Cyclone III, and Cyclone IV devices require the DQS strobes and CK clocks to arrive edge aligned.

There are two timing constraints for DQS versus CK timing path to account for duty cycle distortion. The DQS/DQS# rising edge to CK/CK# rising edge (t_{DQSS}) requires the rising edge of DQS to align with the rising edge of CK to within 25% of a clock cycle, while the DQS/DQS# falling edge setup/hold time from CK/CK# rising edge (t_{DSS}/t_{DSH}) requires the falling edge of DQS to be more than 20% of a clock cycle away from the rising edge of CK.

The TimeQuest Timing Analyzer analyzes the DQS vs CK timing paths using the `set_output_delay (max and min)` constraints. For more information, refer to *<phy_variation_name>*_**phy_ddr_timing.sdc**.

Write Leveling t_{DQSS}

In DDR2 SDRAM (with UniPHY) and DDR3 SDRAM (with ALTMEMPHY and UniPHY) interfaces, write leveling t_{DQSS} timing is a calibrated path that details skew margin for the arrival time of the DQS strobe with respect to the arrival time of CK/CK# at the memory side. For proper write leveling configuration, DLL delay chain must be equal to 8. The PHY IP reports the margin through an equation. For more information, refer to <*phy_variation_name*>**_report_timing_core.sdc**.

Write Leveling t_{DSH}/t_{DSS}

In DDR2 SDRAM (with UniPHY) and DDR3 SDRAM (with ALTMEMPHY and UniPHY) interfaces, write leveling t_{DSH}/t_{DSS} timing details the setup and hold margin for the DQS falling edge with respect to the CK clock at the memory. The PHY IP reports the margin through an equation. For more information, refer to <*phy_variation_name*>**_report_timing_core.sdc**.

DK versus CK (RLDRAM II with UniPHY)

In RLDRAM II with UniPHY designs using the Nios-based sequencer, DK versus CK timing is a calibrated path that details skew margin for the arrival time of the DK clock versus the arrival time of CK/CK# on the memory side. The PHY IP reports the margin through an equation. For more information, refer to <*phy_variation_name*>**_report_timing_core.sdc**.

Bus Turnaround Time

In DDR2 and DDR3 SDRAM, and RLDRAM II (CIO) with UniPHY designs that use bidirectional data bus, you may have potential encounter with data bus contention failure when a write command follows a read command. The bus-turnaround time analysis determines how much margin there is on the switchover time and prevents bus contention. If the timing is violated, you can either increase the controller's bus turnaround time, which may reduce efficiency or board traces delay. Refer to <*variation*>**_report_timing_core.tcl** for the equation. You can find this analysis in the timing report. This analysis is only available for DDR2/3 SDRAM and RLDRAM II UniPHY IPs in Arria II GZ, Arria V, Cyclone V, Stratix IV, and Stratix V devices.

The RTL simulation for ALTMEMPHY IP is unable to detect timing violations because ALTMEMPHY IP is not enhanced with the bus turnaround analysis feature. Therefore, Altera recommends that you verify the design on board by manually changing the default values of MEM_IF_WR_TO_RD_TURNAROUND_OCT and MEM_IF_RD_TO_WR_TURNAROUND_OCT parameters in the controller wrapper file.

To determine whether the bus turnaround time issue is the cause of your design failure and to overcome this timing violation, follow these steps:

1. When the design fails, change the default values of MEM_IF_WR_TO_RD_TURNAROUND_OCT and MEM_IF_RD_TO_WR_TURNAROUND_OCT parameters in the controller wrapper file to a maximum value of 5. If the design passes after the change, it is a bus turnaround issue.

2. To solve the bus turnaround time issue, reduce the values of the MEM_IF_WR_TO_RD_TURNAROUND_OCT and MEM_IF_RD_TO_WR_TURNAROUND_OCT parameters gradually until you reach the minimum value needed for the design to pass on board.

Timing Report DDR

The **Report DDR** task in the TimeQuest Timing Analyzer generates custom timing margin reports for all ALTMEMPHY and UniPHY instances in your design. The TimeQuest Timing Analyzer generates this custom report by sourcing the wizard-generated <*variation*>**_report_timing.tcl** script.

This <*variation*>**_report_timing.tcl** script reports the following timing slacks on specific paths of the DDR SDRAM:

- Read capture
- Read resynchronization
- Mimic, address and command
- Core
- Core reset and removal
- Half-rate address and command
- DQS versus CK
- Write
- Write leveling (t_{DQSS})
- Write leveling (t_{DSS}/t_{DSH})

In Stratix III and Cyclone III designs, the <*variation_name*>**_report_timing.tcl** script checks the design rules and assumptions as listed in "Timing Model Assumptions and Design Rules" on page 11–29. If you do not adhere to these assumptions and rules, you receive critical warnings when the TimeQuest Timing Analyzer runs during compilation or when you run the **Report DDR** task.

To generate a timing margin report, follow these steps:

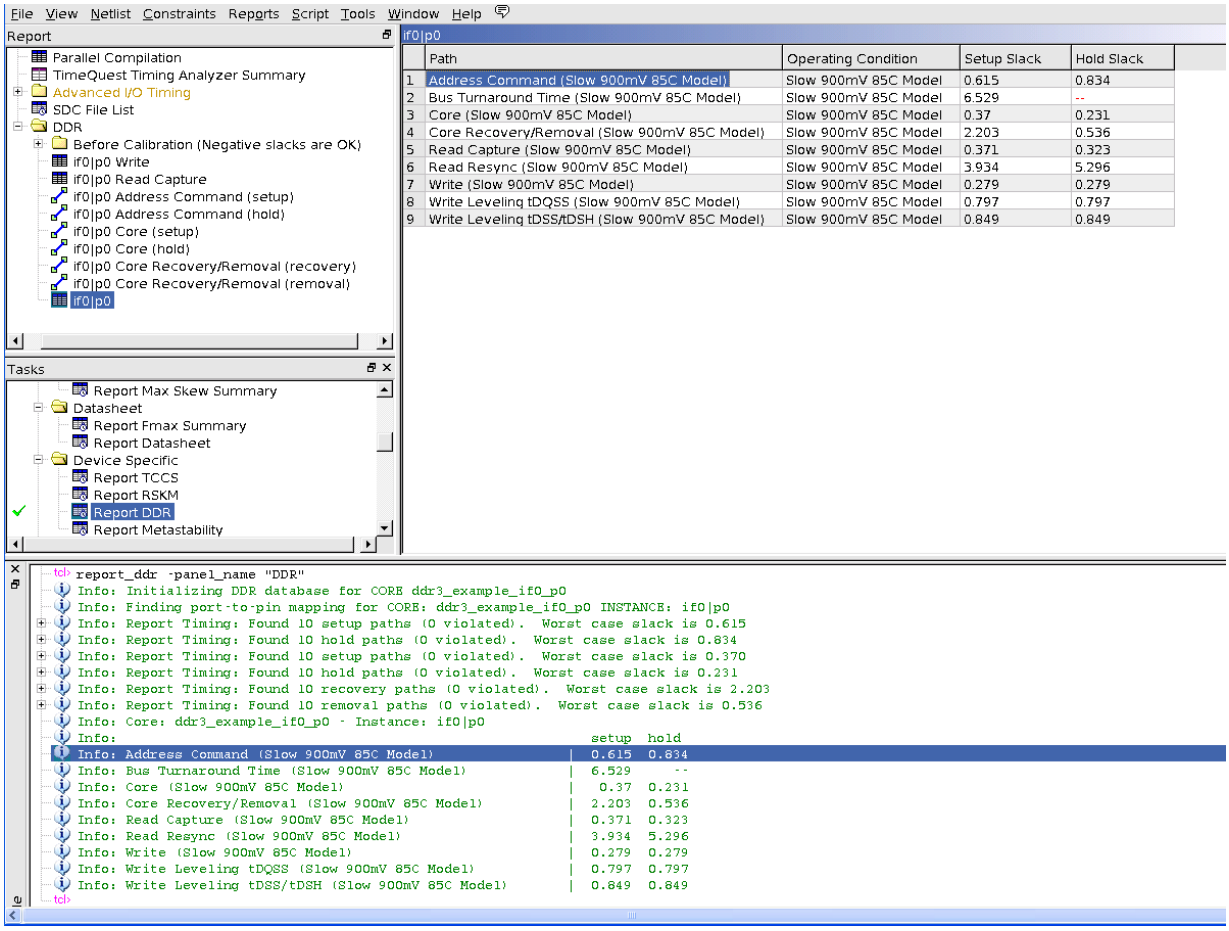
1. Compile your design in the Quartus II software.
2. Launch the TimeQuest Timing Analyzer.
3. Double-click **Report DDR** from the **Tasks** pane. This action automatically executes the **Create Timing Netlist**, **Read SDC File**, and **Update Timing Netlist** tasks for your project.

The **.sdc** may not be applied correctly if the variation top-level file is the top-level file of the project. You must have the top-level file of the project instantiate the variation top-level file.

The **Report DDR** feature creates a new DDR folder in the TimeQuest Timing Analyzer **Report** pane.

Expanding the DDR folder reveals the detailed timing information for each PHY timing path, in addition to an overall timing margin summary for the ALTMEMPHY or UniPHY instance, as shown in Figure 11–10.

Figure 11–10. Timing Margin Summary Window Generated by Report DDR Task



Bus turnaround time shown in Figure 11–10 is available in all UniPHY IPs and devices except in QDR II and QDR II+ SRAM memory protocols and Stratix III devices.

Figure 11–11 shows the timing analysis results calculated using FPGA timing model before adjustment in the **Before Calibration** panel.

Figure 11–11. Read and Write Before Calibration

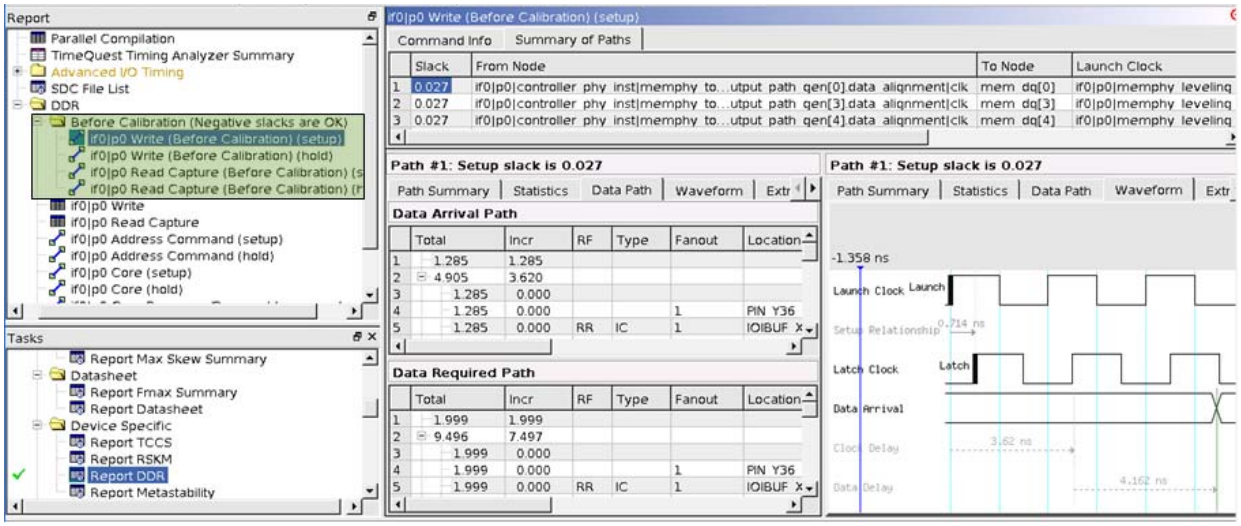


Figure 11–12 and Figure 11–13 show the read capture and write margin summary window generated by the Report DDR Task for a DDR3 core. It first shows the timing results calculated using the FPGA timing model. The *<variation_name>*_**report_timing_core.tcl** then adjusts these numbers to account for effects that are not modeled by either the timing model or by TimeQuest Timing Analyzer. The read and write timing margin analysis for Stratix III and Cyclone III devices do not need any adjustments.

Figure 11–12. Read Capture Margin Summary Window

if0|p0 Read Capture

	Operation	Setup Slack	Hold Slack
1	After Calibration Read Capture	0.371	0.323
2	Before Calibration Read Capture	0.280	0.273
3	Memory Calibration	0.078	0.150
4	Deskew Read	0.157	0.044
5	Quantization error	-0.050	-0.050
6	Calibration uncertainty	-0.093	-0.093

Figure 11–13. Write Capture Margin Summary Window

if0|p0 Write

	Operation	Setup Slack	Hold Slack
1	After Calibration Write	0.279	0.279
2	Before Calibration Write	0.027	0.026
3	Memory Calibration	0.135	0.113
4	Deskew Write and/or more clock pessimism removal	0.216	0.240
5	Quantization error	-0.050	-0.050
6	Calibration uncertainty	-0.050	-0.050

Report SDC

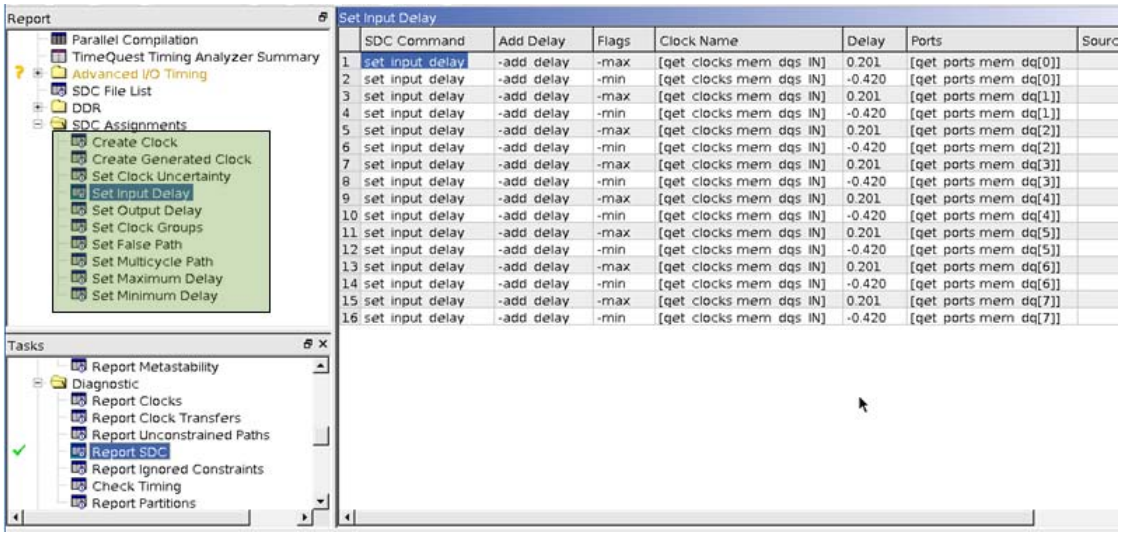
The **Report SDC** task in the TimeQuest Timing Analyzer generates the SDC assignment reports for your design. The TimeQuest Timing Analyzer generates this constraint report by sourcing the **.sdc**. The SDC assignment reports show the constraint applied in the design.

For example, the reports may include the following constraints:

- Create Clock
- Create Generated Clock
- Set Clock Uncertainty
- Set Input Delay
- Set Output Delay
- Set False Path
- Set Multicycle Path
- Set Maximum Delay
- Set Minimum Delay

Figure 11–14 shows the SDC assignments generated by the **Report SDC** task for a DDR3 SDRAM core design. The timing analyzer uses these constraint numbers in analysis to calculate the timing margin. Refer to the **.sdc** files of each constraints number.

Figure 11–14. SDC Assignments Report Window



Calibration Effect in Timing Analysis

Timing analysis for Arria II, Cyclone IV, Stratix IV, and Stratix V devices take into account the calibration effects to improve the timing margin. This section discusses ways to include the calibration effects in timing analysis.

Calibration Emulation for Calibrated Path

In conventional static timing analysis, calibration paths do not include calibration effects. To account for the calibration effects, the timing analyzer emulates the calibration process and integrates it into the timing analysis. Normally the calibration process involves adding or subtracting delays to a path. The analyzer uses the delay obtained through static timing analysis in the emulation algorithm to estimate the extra delay added during calibration. With these estimated delays, the timing analysis emulates hardware calibration and obtains a better estimate timing margin.

Refer to <*phy_variation_name*>**_report_timing.tcl** and <*phy_variation_name*>**_report_timing_core.tcl** for the files that determine the timing margin after calibration.

Calibration Error or Quantization Error

Hardware devices use calibration algorithms when delay information is unknown or incomplete. If the delay information is unknown, the timing analysis of the calibrated paths has to work with incomplete data. This unknown information may cause the timing analysis calibration operations to pick topologies that are different than what would actually occur in hardware. The differences between what can occur in hardware and what occurs in the timing analysis are quantified and included in the timing analysis of the calibrated paths as quantization error or calibration error.

Calibration Uncertainties

Calibration results may change or reduce due to one or more of the following uncertainties:

- Jitter and DCD effects
- Voltage and temperature variations
- Board trace delays changing due to noise on terminated supply voltages

These calibration uncertainties are accounted for in the timing analysis.

Memory Calibration

All the timing paths reported include one or more memory parameters, such as t_{DQSS} and t_{DQSQ}. These specifications indicate the amount of variation that occurs in various timing paths in the memory and abstracts them into singular values so that they can be used by others when interfacing with the memory device.

JEDEC defines these parameters in their specification for memory standards, and every memory vendor must meet this specification or improve it. However, there is no proportion of each specification due to different types of variations. Variations that are of interest are typically grouped into three different types: process variations (P), voltage variations (V), and temperature variations (T). These together compose PVT

variations that typically define the JEDEC specification. You can determine the maximum P variation by comparing different dies, and you can determine the maximum V and T variations by operating a design at the endpoints of the range of voltage and temperature. P variations do not change once the chip has been fabricated, while V and T variations change over time.

The timing analysis for Stratix V FPGAs at 667 MHz of various paths (if the analysis is comprehensive and includes all the sources of noise) indicate that there is no timing margin available. However, the designs do actually work in practice with a reasonable amount of margin. The reason for this behavior is that the memory devices typically have specifications that easily beat the JEDEC specification and that our calibration algorithms calibrate out the process portion of the JEDEC specification, leaving only the V and T portions of the variations.

The memory calibration figure determination includes noting what percentage of the JEDEC specification of various memory parameters is caused by process variations for which Altera IPs' (ALTMEMPHY and UniPHY) calibration algorithms can calibrate out, and to apply that to the full JEDEC specification. The remaining portion of the variation is caused by voltage and temperature variations which cannot be calibrated out.

You can find the percentage of the JEDEC specification that is due to process variation is set in *<variation>*_**report_timing.tcl**.

Timing Model Assumptions and Design Rules

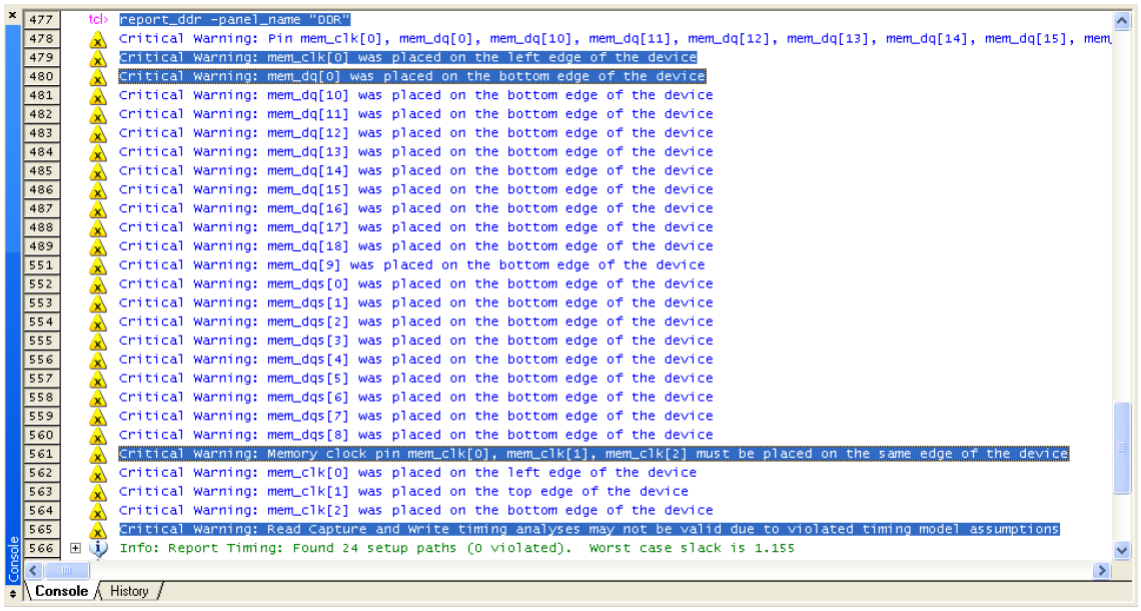
External memory interfaces using Altera IP are optimized for highest performance, and use a high-performance timing model to analyze calibrated and source-synchronous, double-data rate I/O timing paths. This timing model applies to designs that adhere to a set of predefined assumptions. These timing model assumptions include memory interface pin-placement requirements, PLL and clock network usage, I/O assignments (including I/O standard, termination, and slew rate), and many others.

For example, the read and write datapath timing analysis is based on the FPGA pin-level t_{TCCS} and t_{SW} specifications, respectively. While calculating the read and write timing margins, the Quartus II software analyzes the design to ensure that all read and write timing model assumptions are valid for your variation instance.

Timing model assumptions only apply to Stratix III and Cyclone III devices.

When the Report DDR task or **report_timing.tcl** script is executed, the timing analysis assumptions checker is invoked with specific variation configuration information. If a particular design rule is not met, the Quartus II software reports the failing assumption as a Critical Warning message. Figure 11–15 shows a sample set of messages generated when the memory interface DQ, DQS, and CK/CK# pins are not placed in the same edge of the device.

Figure 11–15. Read and Write Timing Analysis Assumption Verification



Memory Clock Output Assumptions

To verify the quality of the FPGA clock output to the memory device (CK/CK# or K/K#), which affects FPGA performance and quality of the read clock/strobe used to read data from the memory device, the following assumptions are necessary:

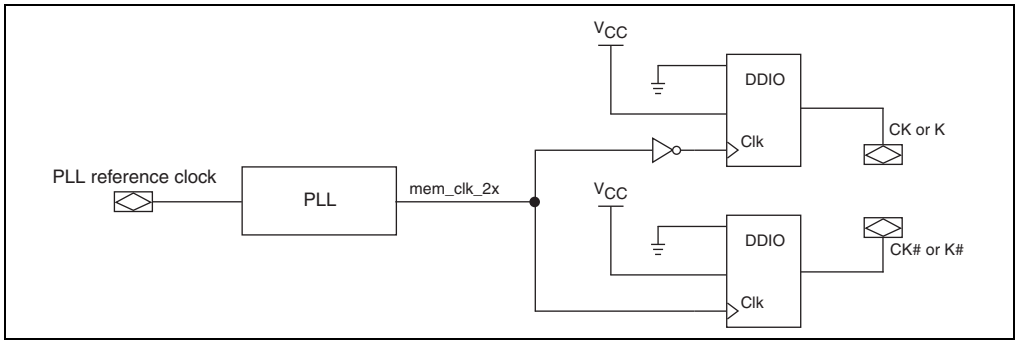
- The slew rate setting must be **Fast** or an on-chip termination (OCT) setting must be used.
- The output delay chains must all be **0** (the default value applied by the Quartus II software). These delay chains include the Cyclone III output register to pin delay chain and the Stratix III D5 and D6 output delay chains.
- The output open-drain parameter on the memory clock pin `IO_OBUF` atom must be **Off**. The **Output Open Drain** logic option must not be enabled.
- The weak pull-up on the CK and CK# pads must be **Off**. The **Weak Pull-Up Resistor** logic option must not be enabled.
- The bus hold on the CK and CK# pads must be **Off**. The **Enable Bus-Hold Circuitry** logic option must not be enabled.
- All CK and CK# pins must be declared as output-only pins or bidirectional pins with the output enable set to V_{CC}.

Cyclone III Devices

For Cyclone III devices the following additional memory clock assumptions are necessary:

- The memory clock output pins must be fed by DDIO output registers and placed on DIFFIO `p-` and `n-` pin pairs.
- The memory output clock signals must be generated using the DDIO configuration shown in Figure 11–16. In this configuration, the high register connects to V_{CC} and the low register connects to GND.

Figure 11–16. DDIO Configuration



- CK and CK# pins must be fed by a DDIO_OUT WYSIWYG with `datainhi` connected to GND and `datainlo` connected to V_{CC}.
- CK or K pins must be fed by a DDIO_OUT with its clock input from the PLL inverted.
- CK# or K# pins must be fed by a DDIO_OUT with its clock input from the PLL uninverted.
- The I/O standard and current strength settings on the memory clock output pins must be as follows:
  - SSTL-2 Class I and 12 mA, or SSTL-2 Class II and 16 mA for DDR SDRAM interfaces
  - SSTL-18 Class I and 12 mA, or SSTL-18 Class II and 16 mA for DDR2 SDRAM interfaces

For more information about placing memory clock output pins, refer to "Additional Placement Rules for Cyclone III and Cyclone IV Devices" in the *Planning Pin and Resource* chapter in volume 2 of the *External Memory Interface Handbook*.

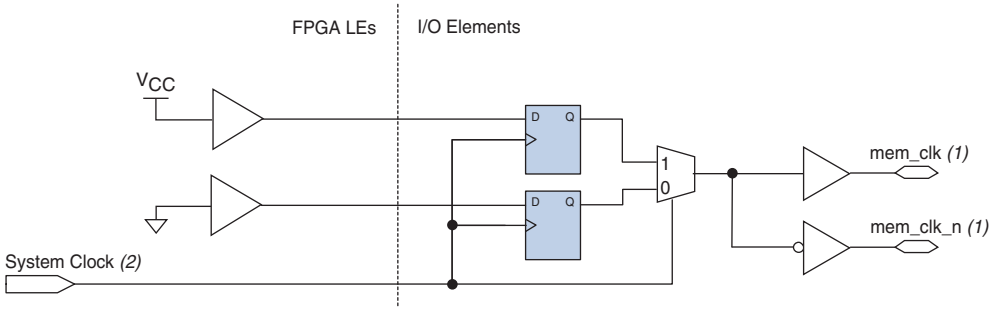
Stratix III Devices

For Stratix III devices the following additional memory clock assumptions are necessary:

- All memory clock output pins must be placed on `DIFFOUT` pin pairs on the same edge of the device.

- For DDR3 SDRAM interfaces:
  - The CK pins must be placed on FPGA output pins marked DQ, DQS, or DQSn.
  - The CK pin must be fed by an `OUTPUT_PHASE_ALIGNMENT` WYSIWYG with a 0° phase shift.
  - The PLL clock driving CK pins must be the same as the clock driving the DQS pins.
  - The T4 (`DDIO_MUX`) delay chains setting for the memory clock pins must be the same as the settings for the DQS pins.
- For non-DDR3 interfaces, the T4 (`DDIO_MUX`) delay chains setting for the memory clock pins must be greater than 0.
- The programmable rise and fall delay chain settings for all memory clock pins must be set to 0.
- The memory output clock signals must be generated with the DDIO configuration shown in Figure 11–17, with a signal splitter to generate the `n-` pin pair and a regional clock network-to-clock to output DDIO block.

Figure 11–17. DIDO Configuration with Signal Splitter



Notes to Figure 11–17:

(1) The `mem_clk[0]` and `mem_clk_n[0]` pins for DDR3, DDR2, and DDR SDRAM interfaces use the I/O input buffer for feedback, therefore bidirectional I/O buffers are used for these pins. For memory interfaces using a differential DQS input, the input feedback buffer is configured as differential input; for memory interfaces using a single-ended DQS input, the input buffer is configured as a single-ended input. Using a single-ended input feedback buffer requires that the I/O standard's VREF voltage is provided to that I/O bank's VREF pins.

(2) Regional QCLK (quadrant) networks are required for memory output clock generation to minimize jitter.

Write Data Assumptions

To verify the memory interface using the FPGA TCCS output timing specifications, the following assumptions are necessary:

- For QDRII, QDRII+, and RLDRAM II SIO memory interfaces, the write clock output pins (such as K/K# or DK/DK#) must be placed in `DQS/DQSn` pin pairs.
- The PLL clock used to generate the write-clock signals and the PLL clock used to generate the write-data signals must come from the same PLL.
- The slew rate for all write clocks and write data pins must be set to **Fast** or OCT must be used.

- When auto deskew is not enabled (or not supported by the ALTMEMPHY configuration), the output delay chains and output enable delay chains must all be set to the default values applied by Quartus II. These delay chains include the Cyclone III output register and output enable register-to-pin delay chains, and the Stratix III D5 and D6 delay chains.
- The output open drain for all write clocks and write data pins' `IO_OBUF` atom must be set to **Off**. The **Output Open Drain** logic option must not be enabled.
- The weak pull-up for all write clocks and write data pins must be set to **Off**. The **Weak Pull-Up Resistor** logic option must not be enabled.
- The Bus Hold for all write clocks and write data pins must be set to **Off**. The **Enable Bus-Hold Circuitry** logic option must not be enabled.

Cyclone III Devices

For Cyclone III devices the following additional write data assumptions are necessary:

- Write data pins (including the DM pins) must be placed on DQ pins related to the selected DQS pins.
- All write clock pins (DQS/DQS#) must be fed by DDIO output registers.
- All write data pins must be fed by DDIO output registers, V_{CC}, or GND.
- The phase shift of the PLL clock used to generate the write clocks must be 72° to 108° more than the PLL clock used to generate the write data (nominally 90° offset).
- The I/O standard and current strength settings on the write data- and clock-output pins must be as follows:
  - SSTL-2 Class I and 12 mA, or SSTL-2 Class II and 16 mA for DDR SDRAM interfaces
  - SSTL-18 Class I and 8/12 mA, or SSTL-18 Class II and 16 mA for DDR2 SDRAM interfaces

Stratix III Devices

For Stratix III devices the following additional write data assumptions are necessary:

- Differential write clock signals (DQS/DQSn) must be generated using the signal splitter.
- The write data pins (including the DM pins) must be placed in related `DQ` pins associated with the chosen `DQS` pin. The only exception to this rule is for QDRII and QDRII+ ×36 interfaces emulated using two ×18 DQ groups. For such interfaces, all of the write data pins must be placed on the same edge of the device (left, right, top, or bottom). Also, the write clock K/K# pin pair should be placed on one of the `DQS/DQSn` pin pairs on the same edge.

- All write clock pins must have similar circuit structure.
  - For DDR2 SDRAM interfaces and DDR3 SDRAM with leveling interfaces, all DQS/DQS# write strobes must be fed by DDIO output registers clocked by the write-leveling delay chain in the OUTPUT_PHASE_ALIGNMENT block.
  - For DDR and DDR2 SDRAM interfaces, all write clock pins must be fed by DDIO output registers clocked by a global or regional clock network.
- All write data pins must have similar circuit structure.
  - For DDR3 SDRAM interfaces, all write data pins must be fed by either DDIO output registers clocked by the OUTPUT_PHASE_ALIGNMENT block, V_{CC}, or GND.
  - For DDR and DDR2 SDRAM interfaces, all write data pins must be fed by either DDIO output registers clocked by a global or regional clock network, V_{CC}, or GND.
- The write clock output must be 72,° 90°, or 108° more than the write data output.
  - For DDR2 SDRAM and DDR3 SDRAM with leveling interfaces, the write-leveling delay chain in the OUTPUT_PHASE_ALIGNMENT block must implement a phase shift of 72°, 90°, or 108° to center-align write clock with write data.
  - For DDR and DDR2 SDRAM interfaces, the phase shift of the PLL clock used to clock the write clocks must be 72 to 108° more than the PLL clock used to clock the write data clocks to generated center-aligned clock and data.
- The T4 (DDIO_MUX) delay chains must all be set to **3**. When differential DQS (using splitter) is used, T4 must be set to **2**.
- The programmable rise and fall delay chain settings for all memory clock pins must be set to **0**.

Table 11–9 lists I/O standards supported for the write clock and write data signals for each memory type and pin location.

Table 11–9. I/O standards (Part 1 of 2)

Memory Type	Placement	Legal I/O Standards for DQS	Legal I/O Standards for DQ
DDR3 SDRAM	Row I/O	Differential 1.5-V SSTL Class I	1.5-V SSTL Class I
DDR3 SDRAM	Column I/O	Differential 1.5-V SSTL Class I Differential 1.5-V SSTL Class II	1.5-V SSTL Class I 1.5-V SSTL Class II
DDR2 SDRAM	Any	SSTL-18 Class I SSTL-18 Class II Differential 1.8V SSTL Class I Differential 1.8V SSTL Class II	SSTL-18 Class I SSTL-18 Class II
DDR SDRAM	Any	SSTL-2 Class I SSTL-2 Class II	SSTL-2 Class I SSTL-2 Class II

Table 11–9. I/O standards (Part 2 of 2)

Memory Type	Placement	Legal I/O Standards for DQS	Legal I/O Standards for DQ
QDRII and QDR II + SRAM	Any	HSTL-1.5 Class I HSTL-1.8 Class I	HSTL-1.5 Class I HSTL-1.8 Class I
RLDRAM II	Any	HSTL-1.5 Class I HSTL-1.8 Class I	HSTL-1.5 Class I HSTL-1.8 Class I

Read Data Assumptions

To verify that the external memory interface can use the FPGA Sampling Window (SW) input timing specifications, the following assumptions are necessary:

- The read clocks input pins must be placed on DQS pins. DQS/DQS# inputs must be placed on differential DQS/DQSn pins on the FPGA.
- Read data pins (DQ) must be placed on the DQ pins related to the selected DQS pins.
- For QDR II and QDR II+ SRAM interfaces, the complementary read clocks must have a single-ended I/O standard setting of HSTL-18 Class I or HSTL-15 Class I.
- For RLDRAM II interfaces, the differential read clocks must have a single ended I/O standard setting of HSTL 18 Class I or HSTL 15 Class I.

Cyclone III Devices

For Cyclone III devices the following additional read data and mimic pin assumptions are necessary:

- The I/O standard setting on read data and clock input pins must be as follows:
  - SSTL-2 Class I and Class II for DDR SDRAM interface
  - SSTL-18 Class I and Class II for DDR2 SDRAM interfaces
- The read data and mimic input registers (flip-flops fed by the read data pin's input buffers) must be placed in the LAB adjacent to the read data pin. A read data pin can have 0 input registers.
- Specific routing lines from the IOE to core read data/mimic registers must be used. The Quartus II Fitter ensures proper routing unless user-defined placement constraints or LogicLock™ assignments force non-optimal routing. User assignments that prevent input registers from being placed in the LAB adjacent to the IOE must be removed.
- The read data and mimic input pin input pad to core/register delay chain must be set to **0**.
- If all read data pins are on row I/Os or column I/Os, the mimic pin must be placed in the same type of I/O (row I/O for read-data row I/Os, column I/O for read-data column I/Os). For wraparound cases, the mimic pin can be placed anywhere.

Stratix III Devices

For Stratix III devices the following additional read data and mimic pin assumptions are necessary:

- For DDR3, DDR2, and DDR SDRAM interfaces, the read clock pin can only drive a DQS bus clocking a ×4 or ×9 DQ group.
- For QDR II, QDR II+ SRAM, and RLDRAM II interfaces, the read clock pin can only drive a DQS bus clocking a ×9, ×18, or ×36 DQ group.
- For non-wraparound DDR, DDR2, and DDR3 interfaces, the mimic pin, all read clock, and all read data pins must be placed on the same edge of the device (top, bottom, left, or right). For wraparound interfaces, these pins can be placed on adjacent row I/O and column I/O edges and operate at reduced frequencies.
- All read data pins and the mimic pin must feed DDIO_IN registers and their input delay chains D1, D2, and D3 set to the Quartus II default.
- DQS phase-shift setting must be either 72° or 90° (supports only one phase shift for each operating band and memory standard).
- All read clock pins must have the `dqs_ctrl_latches_enable` parameter of its DQS_DELAY_CHAIN WYSIWYG set to false.
- The read clocks pins must have their D4 delay chain set to the Quartus II default value of **0**.
- The read data pins must have their T8 delay chain set to the Quartus II default value of **0**.
- When differential DQS strobes are used (DDR3 and DDR2 SDRAM), the mimic pin must feed a true differential input buffer. Placing the memory clock pin on a `DIFFIO_RX` pin pair allows the mimic path to track timing variations on the DQS input path.
- When single ended DQS strobes are used, the mimic pin must feed a single ended input buffer.

Mimic Path Assumptions

To verify that the ALTMEMPHY-based DDR, DDR2, or DDR3 SDRAM interface's mimic path is configured correctly, the mimic path input must be placed on the `mem_clk[0]` pin.

DLL Assumptions

The following DLL assumptions are necessary:

These assumptions do not apply to Cyclone III devices.

- The DLL must directly feed its `delayctrlout[]` outputs to all DQS pins without intervening logic or inversions.
- The DLL must be in a valid frequency band of operation as defined in the corresponding device data sheet.
- The DLL must have jitter reduction mode and dual-phase comparators enabled.

PLL and Clock Network Assumptions

The PLL and clock network assumptions vary for each device family.

Stratix III Devices

- The PLL that generates the memory output clock signals and write data and clock signals must be set to **No compensation** mode to minimize output clock jitter.
- The reference input clock signal to the PLL must be driven by the dedicated clock input pin located adjacent to the PLL, or from the clock output signal from the adjacent PLL. To minimize output clock jitter, the reference input clock pin to the ALTMEMPHY PLL must not be routed through the core using global or regional clock networks. If the reference clock cascades from another PLL, that upstream PLL must be in **No compensation** mode and **Low bandwidth** mode.
- For DDR3 and DDR2 SDRAM interfaces, use only regional or dual regional clock networks to route PLL outputs that generate the write data, write clock, and memory output clock signals. This requirement ensures that the memory output clocks (CK/CK#) meet the memory device input clock jitter specifications, and that output timing variations or skews are minimized.
- For other memory types, the same clock tree type (global, regional, or dual regional) is recommended for PLL clocks generating the write clock, write data, and memory clock signals to minimize timing variations or skew between these outputs.

Cyclone III Devices

To verify that the memory interface's PLL is configured correctly, the following assumptions are necessary:

- The PLL that generates the memory output clock signals and write data/clock signals must be set to **Normal** compensation mode in Cyclone III devices.
- PLL cascading is not supported.
- The reference input clock signal to the PLL must be driven by the dedicated clock input pin located adjacent to the PLL. The reference input clock pin must not be routed through the core using global or regional clock networks to minimize output clock jitter.

Timing Closure

This section describes common issues and how to optimize timing.

Common Issues

This topic describes potential timing closure issues that can occur when using the ALTMEMPHY or UniPHY IP. For possible timing closure issues with ALTMEMPHY or UniPHY variations, refer to the *Quartus II Software Release Notes* for the software version that you are using. You can solve some timing issues by moving registers or changing the project fitting setting to **Standard** (from **Auto**).

The *Quartus II Software Release Notes* list common timing issues that can be encountered in a particular version of the Quartus II software.

Missing Timing Margin Report

The ALTMEMPHY and UniPHY timing margin reports may not be generated during compilation if the **.sdc** does not appear in the Quartus II project settings.

Timing margin reports are not generated if you specify the ALTMEMPHY or UniPHY variation as the top-level project entity. Instantiate the ALTMEMPHY or UniPHY variation as a lower level module in your user design or memory controller.

Incomplete Timing Margin Report

The timing report may not include margin information for certain timing paths if certain memory interface pins are optimized away during synthesis. Verify that all memory interface pins appear in the <*variation*>**_autodetectedpins.tcl** (ALTMEMPHY) or <*variation*>**_all_pins.txt** (UniPHY) file generated during compilation, and ensure that they connect to the I/O pins of the top-level FPGA design.

Read Capture Timing

In Stratix III and Stratix IV devices, read capture timing may fail if the DQS phase shift selected is not optimal or if the board skew specified is large.

- You can adjust the effective DQS phase shift implemented by the DLL to balance setup and hold margins on the read timing path. The DQS phase shift can be adjusted in coarse PVT-compensated steps of 22.5°, 30°, 36°, or 45° by changing the number of delay buffers (range 1 to 4), or in fine steps using the DQS phase offset feature that supports uncompensated delay addition and subtraction in approximately 12 ps steps.
- To adjust the coarse phase shift selection, determine the supported DLL modes for your chosen memory interface frequency by referencing the DLL and DQS Logic Block Specifications tables in the *Switching Characteristics* section of the device data sheet. For example, a 400 MHz DDR2 interface on a -2 speed grade device can use DLL mode 5 (resolution 36°, range 290 – 450 MHz) to implement a 72° phase shift, or DLL mode 6 (resolution 45°, range 360–560 MHz) to implement a 90° phase shift.

In Cyclone III devices, the read capture is implemented using a calibrated clock, and therefore no clock phase-shift adjustment is possible. Additionally, the capture registers are routed to specific LE registers in the logic array blocks (LABs) adjacent to the IOE using predefined routing. Therefore, no timing optimization is possible for this path. Ensure that you select the correct memory device speed grade for the FPGA speed grade and interface frequency.

Ensure that you specify the appropriate board-skew parameter when you parameterize the controllers in the parameter editor. The default board trace length mismatch used is 20 ps.

Write Timing

Negative timing margins may be reported for write timing paths if the PLL phase shift used to generate the write data signals is not optimal. Adjust the PLL phase shift selection on the write clock PLL output using the PLL MegaWizard Plug-In Manager.

Regenerating the ALTMEMPHY- or UniPHY-based controller overwrites changes made using the PLL MegaWizard Plug-In Manager.

Address and Command Timing

You can optimize the timing margins on the address and command timing path by changing the PLL phase shift used to generate these signals. Modify the **Dedicated Clock Phase** parameter in the **PHY Settings** page of the ALTMEMPHY parameter editor. In the DDR2 or DDR3 SDRAM Controllers with UniPHY IP cores, modify the **Additional CK/CK# phase** and **Additional Address and Command clock phase** parameters.

The DDR2 and DDR3 SDRAM memory controllers use 1T memory timing even in half-rate mode, which may affect the address command margins for DDR2 or DDR3 SDRAM designs that use memory DIMMs. DDR2 SDRAM designs have a greater impact because the address command bus fans out to all the memory devices on a DIMM increasing the loading effect on the bus. Make sure your board designs are robust enough to have the memory clock rising edge within the 1T address command window. You can also use the **Additional Address and Command clock phase** parameter to adjust the phase of the address and command if needed.

The far-end load value and board trace delay differences between address and command and memory clock pins can result in timing failures if they are not accounted for during timing analysis.

The Quartus II Fitter may not optimally set output delay chains on the address and command pins. To ensure that any PLL phase-shift adjustments are not negated by delay chain adjustments, create logic assignments using the Assignment Editor to set all address and command output pin D5 delay chains to 0.

For HardCopy III, HardCopy IV, Stratix III, and Stratix IV devices, some corner cases of device family and memory frequency may require an increase to the address and command clock phase to meet core timing. You can identify this situation, if the DDR timing report shows a `PHY setup` violation with the `phy_clk` launch clock, and the address and command latch clock—clock 0 (half-rate `phy_clk`) or 2 (full-rate `phy_clk`), and 6, respectively.

If you see this timing violation, you may fix it by advancing the address and command clock phase as required. For example, a 200-ps violation for a 400-MHz interface represents 8% of the clock period, or 28.8° . Therefore, advance the address and command phase from 270° to 300° . However, this action reduces the setup and hold margin at the DDR device.

PHY Reset Recovery and Removal

A common cause for reset timing violations in ALTMEMPHY or UniPHY designs is the selection of a global or regional clock network for a reset signal. The ALTMEMPHY or UniPHY IP does not require any dedicated clock networks for reset signals. Only ALTMEMPHY or UniPHY PLL outputs require clock networks, and any other PHY signal using clock networks may result in timing violations.

You can correct such timing violations by:

- Setting the Global Signal logic assignment to **Off** for the problem path (using the Assignment Editor), or
- Adjusting the logic placement (using the Assignment Editor or Chip Planner)

Clock-to-Strobe (for DDR and DDR2 SDRAM Only)

Memory output clock signals and DQS strobes are generated using the same PLL output clock. Therefore, no timing optimization is possible for this path and positive timing margins are expected for interfaces running at or below the FPGA data sheet specifications.

For DDR3 interfaces, the timing margin for this path is reported as **Write Leveling**.

Read Resynchronization and Write Leveling Timing (for SDRAM Only)

These timing paths apply only to Arria II GX, Stratix III, and Stratix IV devices, and are implemented using calibrated clock signals driving dedicated IOE registers. Therefore, no timing optimization is possible for these paths, and positive timing margin is expected for interfaces running at or below the FPGA data sheet specifications.

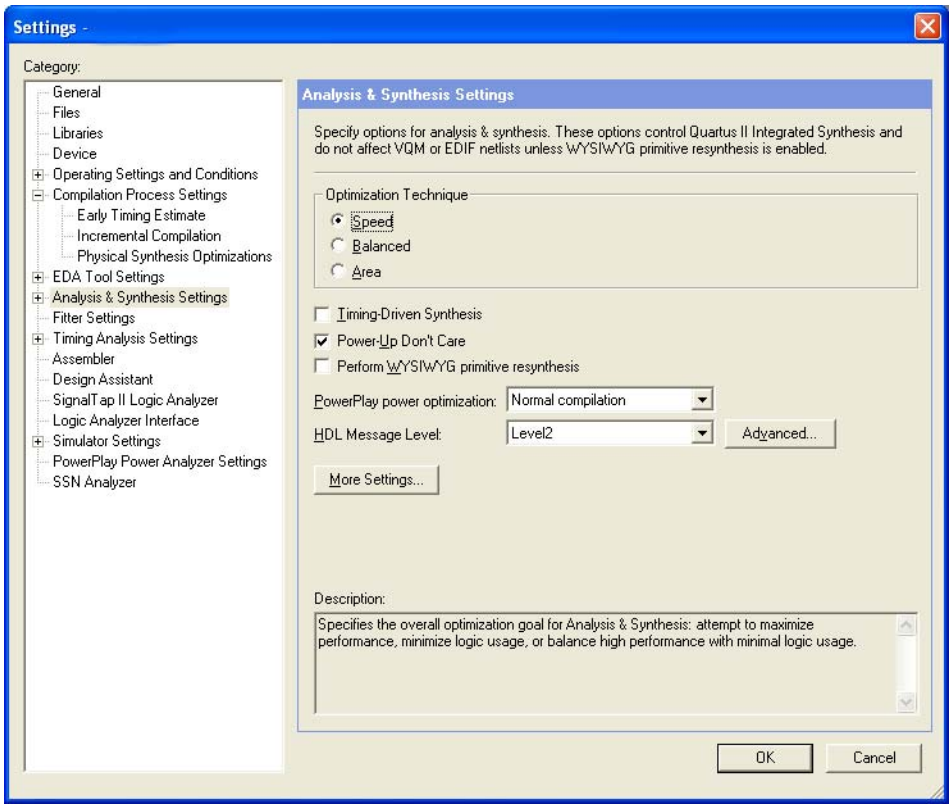
Ensure that you specify the correct memory device timing parameters (t_{DQSCK}, t_{DSS}, t_{DSH}) and board skew (t_{EXT}) in the ALTMEMPHY, DDR, DDR2, and DDR3 SDRAM Controllers with ALTMEMPHY, or DDR2 and DDR3 SDRAM Controllers with UniPHY parameter editor.

Optimizing Timing

For full-rate designs you may need to use some of the Quartus II advanced features, to meet core timing, by following these steps:

1. On the Assignments menu click **Settings**. In the **Category** list, click **Analysis & Synthesis Settings**. For **Optimization Technique** select **Speed** (see Figure 11–18).

Figure 11–18. Optimization Technique



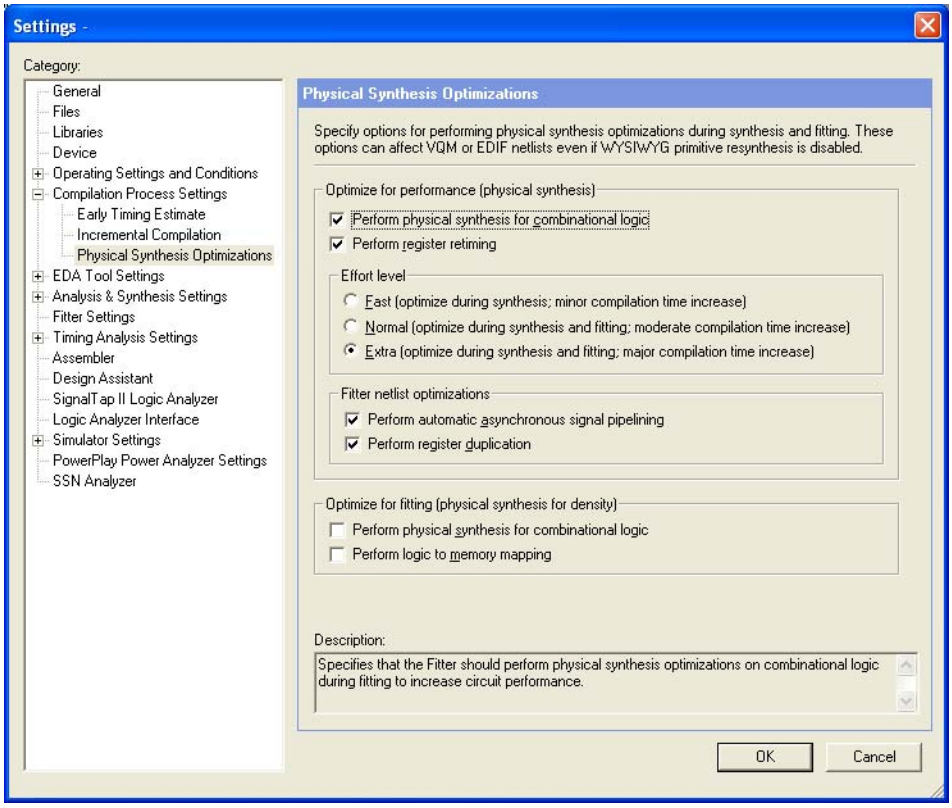
2. In the **Category** list, click **Physical Synthesis Optimizations**. Specify the following options:
   - Turn on **Perform physical synthesis for combinational logic**.

   For more information about physical synthesis, refer to the *Netlist and Optimizations and Physical Synthesis* chapter in the *Quartus II Software Handbook*.

   - Turn on **Perform register retiming**
   - Turn on **Perform automatic asynchronous signal pipelining**
   - Turn on **Perform register duplication**

You can initially select **Normal** for **Effort level**, then if the core timing is still not met, select **Extra** (see Figure 11–19).

Figure 11–19. Physical Synthesis Optimizations



Timing Deration Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs

In a multiple chip select system, each individual rank has its own chip select signal. Consequently, you must change the **Total Memory chip selects**, **Number of chip select** (for discrete components) or **Number of chip select per slot** (DIMMs) in the **Preset Editor** of the ALTMEMPHY- or UniPHY-based parameter editors.

In the **Preset Editor**, you must leave the baseline non-derated ^{t}DS, ^{t}DH, ^{t}IS, ^{t}IH values, because the settings on the **Board Settings** page account for multiple chip select slew rate deration.

This section explains the following two timing deration methodologies for multiple chip-select DDR2 and DDR3 SDRAM designs:

- Timing Deration using the Board Settings
- Timing Deration Using the Excel-Based Calculator

For Arria II GX, Arria II GZ, Stratix IV, and Stratix V devices, the ALTMEMPHY, and ALTMEMPHY- and UniPHY-based controller parameter editors have an option to select multiple chip-select deration.

To perform multiple chip-select timing deration for other Altera devices (for example Cyclone III and Stratix III devices), Altera provides an Excel-based calculator available from the Altera website.

Timing deration in this section applies to either discrete components or DIMMs.

You can derate DDR SDRAM multiple chip select designs by using the DDR2 SDRAM section of the Excel-based calculator, but Altera does not guarantee the results.

This section assumes you know how to obtain data on PCB simulations for timing deration from HyperLynx or any other PCB simulator.

Multiple Chip Select Configuration Effects

A DIMM contains one or several RAM chips on a small PCB with pins that connect it to another system such as a motherboard or router.

Nonregistered (unbuffered) DIMMs (or UDIMMs) connect address and control buses directly from the module interface to the DRAM on the module.

Registered DIMMs (RDIMMs) and load-reduced DIMMs (LRDIMMs) improve signal integrity (and hence potential clock rates and/or overall memory size) by electrically buffering address and command signals as well as the data bus (for LRDIMMs) at a cost of additional latency. Both RDIMMs and LRDIMMs use parity on the address and command bus for increased robustness.

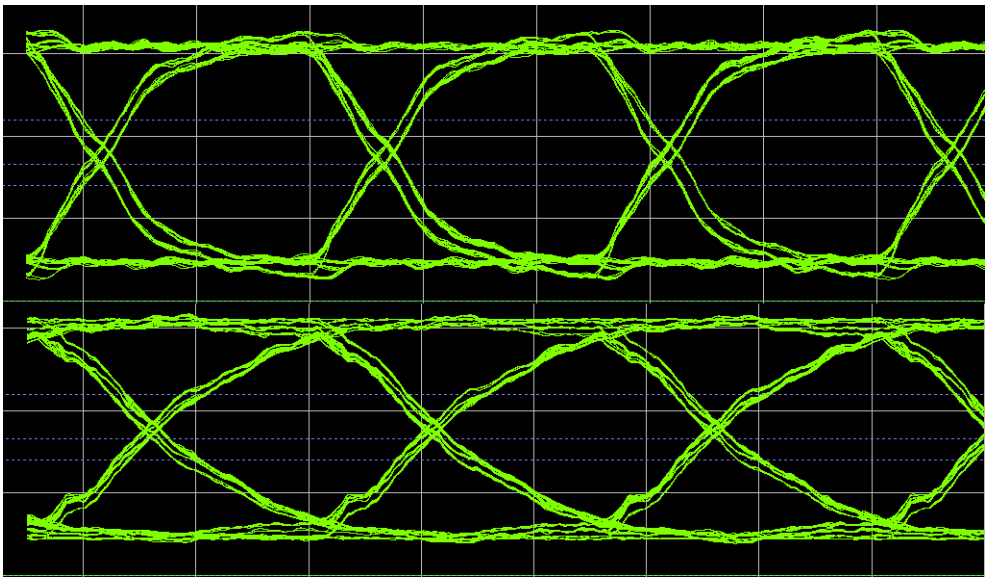
Multiple chip select configurations allow for one set of data pins (and address pins for UDIMMs) to be connected to two or more memory ranks. Multiple chip select configurations have a number of effects on the timing analysis including the intersymbol interference (ISI) effects, board effects, and calibration effects.

ISI Effects

With multiple chip selects and possible slots loading the far end of the pin, there may be ISI effects on a signal causing the eye openings for DQ, DQS, and address and command signals to be smaller than for single-rank designs (Figure 11–20). Figure 11–20 shows the eye shrinkage for DQ signal of a single rank system (top) and multiple chip select system (bottom). The ISI eye reductions reduce the timing window available for both the write path and the address and command path analysis. You must specify them as output delay constraints in the **.sdc**.

Extra loading from the additional ranks causes the slew rate of signals from the FPGA to be reduced. This reduction in slew rate affects some of the memory parameters including data, address, command and control setup and hold times (^{t}DS, ^{t}DH, ^{t}IS, and ^{t}IH).

Figure 11–20. Eye Shrinkage for DQ Signal



Calibration Effects

In addition to the SI effects, multiple chip select topologies change the way that the FPGA calibrates to the memories. In single-rank situations with leveling, the calibration algorithms set delay chains in the FPGA such that specific DQ and DQS pin delays from the memory are equalized (only for ALTMEMPHY-based designs at 401 MHz and above) so that the write-leveling and resynchronization timing requirements are met. In single rank without leveling situations, the calibration algorithm centers the resynchronization or capture phase such that it is optimum for the single rank. When there are two or more ranks in a system, the calibration algorithms must calibrate to the average point of the ranks.

Board Effects

Unequal length PCB traces result in delays reducing timing margins. Furthermore, skews between different memory ranks can further reduce the timing margins in multiple chip select topologies. Board skews can also affect the extent to which the FPGA can calibrate to the different ranks. If the skew between various signals for different ranks is large enough, the timing margin on the fully calibrated paths such as write leveling and resynchronization changes.

To account for all these board effects for Arria II GX, Arria II GZ, Arria V, Cyclone V, Stratix IV, and Stratix V devices, refer to the **Board Settings** page in the ALTMEMPHY- or UniPHY-based controller parameter editors.

To perform multiple chip select timing deration for other Altera devices (for example Cyclone III and Stratix III devices), use the Excel-based calculator available from the Altera website.

Timing Deration using the Board Settings

When you target Arria II GX, Arria II GZ, Arria V, Cyclone V, Stratix IV, or Stratix V devices, the ALTMEMPHY- or UniPHY-based parameter editors include the **Board Settings** page, to automatically account for the timing deration caused by the multiple chip selects in your design.

When you target Cyclone III or Stratix III devices, you can derate single chip-select designs using the parameter editors to account for the skews, ISI, and slew rates in the **Board Settings** page.

If you are targeting Cyclone III or Stratix III devices you see the following warning:

```
"Warning: Calibration performed on all chip selects, timing analysis
only performed on first chip select. Manual timing derating is
required"
```

You must perform manual timing deration using the Excel-based calculator.

The **Board Settings** page allows you to enter the parameters related to the board design including skews, signal integrity, and slew rates. The **Board Settings** page also includes the board skew setting parameter, **Addr/Command to CK skew**, (previously on the **PHY Settings** tab).

Slew Rates

You can obtain the slew rates in one of the following ways:

- Altera performs PCB simulations on internal Altera boards to compute the output slew rate and ISI effects of various multiple chip select configurations. These simulation numbers are prepopulated in the **Slew Rates** based on the number of ranks selected. The address and command setup and hold times (tDS, tDH, tIS, tIH) are then computed from the slew rate values and the baseline nonderated tDS, tDH, tIS, tIH numbers entered in the **Preset Editor**. The parameter editor shows the computed values in **Slew Rates**. If you do not have access to a simulator to obtain accurate slew rates for your specific system, Altera recommends that you use these prepopulated numbers for an approximate estimate of your actual board parameters.

- Alternatively, you can update these default values, if dedicated board simulation results are available for the slew rates. Custom slew rates cause the ^{t}DS, ^{t}DH, ^{t}IS, ^{t}IH values to be updated. Altera recommends performing board level simulations to calculate the slew rate numbers that accounts for accurate board-level effects for your board.
- You can modify the auto-calculated tDS, tDH, tIS, tIH values with more accurate dedicated results direct from the vendor data sheets, if available.

Slew Rate Setup, Hold, and Derating Calculation

Slew rate is calculated based on the nominal slew rate for setup and hold times. The total tIS (setup time) and tIH (hold time) required is calculated by adding the Micron data sheet tIS (base) and tIH (base) values to the delta tIS and delta tIH derating values, respectively.

For more information about slew rate calculation, setup, hold, and derating values, download the data sheet specifications from the following vendor websites:

- Micron (www.micron.com)
  For example, refer to *Command and Address Setup, Hold, and Derating* section in the Micron DDR3 data sheet.
- JEDEC (www.jedec.org)
  For example, refer to the DDR2 SDRAM Standard data sheet.

The following section describes the timing derating algorithms and shows you where to obtain the setup, hold, and derating values in the Micron datasheet.

The slew rate derating process uses the following timing derating algorithms, which is similar to the JEDEC specification:

$$tDS = tDS(base) + delta\ tDS + (V_{IHAC} - V_{REF})/(DQ\ slew\ rate)$$

$$^{t}DH = tDH(base) + delta\ tDH + (V_{IHDC} - V_{REF})/(DQ\ slew\ rate)$$

$$tIS = tIS\ (base) + delta\ tIS + (V_{IHAC} - V_{REF})/(Address/Command\ slew\ rate)$$

$$tIH = tIH(base) + delta\ tIH + (V_{IHDC} - V_{REF})/(Address/Command\ slew\ rate)$$

where:

a. The setup and hold values for tDS(base), tDH(base), tIS(base), and tIH(base) are obtained from the Micron datasheet.
Figure 11–21 shows a screenshot example of the values from the Micron datasheet.

Figure 11–21. Setup and Hold Values from Micron Datasheet

Parameter		Symbol	DDR3-800 Min	DDR3-800 Max	DDR3-1066 Min	DDR3-1066 Max
DQ Input Timing						
Data setup time to DQS, DQS#	Base (specification)	^{t}DS AC175	75	–	25	–
	V_{REF} @ 1 V/ns		250	–	200	–
Data setup time to DQS, DQS#	Base (specification)	^{t}DS AC150	125	–	75	–
	V_{REF} @ 1 V/ns		275	–	250	–
Data setup time to DQS, DQS#	Base (specification)	^{t}DS AC135	–	–	–	–
	V_{REF} @ 1 V/ns		–	–	–	–
Data hold time from DQS, DQS#	Base (specification)	^{t}DH DC100	150	–	100	–
	V_{REF} @ 1 V/ns		250	–	200	–
Minimum data pulse width		^{t}DIPW	600	–	490	–
Command and Address Timing						
DLL locking time		^{t}DLLK	512	–	512	–
CTRL, CMD, ADDR setup to CK,CK#	Base (specification)	^{t}IS AC175	200	–	125	–
	V_{REF} @ 1 V/ns		375	–	300	–
CTRL, CMD, ADDR setup to CK,CK#	Base (specification)	^{t}IS AC150	350	–	275	–
	V_{REF} @ 1 V/ns		500	–	425	–
CTRL, CMD, ADDR hold from CK,CK#	Base (specification)	^{t}IH DC100	275	–	200	–
	V_{REF} @ 1 V/ns		375	–	300	–
Minimum CTRL, CMD, ADDR pulse width		^{t}IPW	900	–	780	–

b. The JEDEC defined logic trip points for DDR3 SDRAM memory standard are as follow:

- $V_{IHAC} = V_{REF} + 0.175$ V
- $V_{IHDC} = V_{REF} + 0.1$ V
- $V_{ILAC} = V_{REF} - 0.175$ V
- $V_{ILDC} = V_{REF} - 0.1$ V

c. The derating values for delta tIS, tIH, tDH, and tDS are obtained from the Micron data sheet.
Figure 11–22 shows the screenshot of the derating values from the Micron data sheet.

Figure 11–22. Derating Values from Micron Datasheet

Δ^tIS, Δ^tIH Derating (ps) – AC/DC-Based AC175 Threshold: $V_{IH(AC)} = V_{REF(DC)} + 175mV$, $V_{IL(AC)} = V_{REF(DC)} - 175$												
CMD/ ADDR Slew Rate V/ns	CK, CK# Differential Slew Rate											
	4.0 V/ns		3.0 V/ns		2.0 V/ns		1.8 V/ns		1.6 V/ns		1.4 V/ns	
	Δ^tIS	Δ^tIH	Δ^tIS	Δ^tIH	Δ^tIS	Δ^tIH	Δ^tIS	Δ^tIH	Δ^tIS	Δ^tIH	Δ^tIH	Δ^tIH
2.0	88	50	88	50	88	50	96	58	104	66	112	74
1.5	59	34	59	34	59	34	67	42	75	50	83	58
1.0	0	0	0	0	0	0	8	8	16	16	24	24
0.9	–2	–4	–2	–4	–2	–4	6	4	14	12	22	20
0.8	–6	–10	–6	–10	–6	–10	2	–2	10	6	18	14
0.7	–11	–16	–11	–16	–11	–16	–3	–8	5	0	13	8
0.6	–17	–26	–17	–26	–17	–26	–9	–18	–1	–10	7	–2
0.5	–35	–40	–35	–40	–35	–40	–27	–32	–19	–24	–11	–16
0.4	–62	–60	–62	–60	–62	–60	–54	–52	–46	–44	–38	–36

Shaded cells indicate slew rate combinations not supported

Δ^tDS, Δ^tDH Derating (ps) – AC/DC-Based												
DQ Slew Rate V/ns	DQS, DQS# Differential Slew Rate											
	4.0 V/ns		3.0 V/ns		2.0 V/ns		1.8 V/ns		1.6 V/ns		1.4 V/ns	
	Δ^tDS	Δ^tDH	Δ^tDS	Δ^tDH	Δ^tDS	Δ^tDH	Δ^tDS	Δ^tDH	Δ^tDS	Δ^tDH	Δ^tDS	Δ^tDH
2.0	88	50	88	50	88	50						
1.5	59	34	59	34	59	34	67	42				
1.0	0	0	0	0	0	0	8	8	16	16		
0.9			–2	–4	–2	–4	6	4	14	12	22	20
0.8					–6	–10	2	–2	10	6	18	14
0.7							–3	–8	5	0	13	8
0.6									–1	–10	7	–2
0.5											–11	–16
0.4												

Intersymbol Interference

ISI parameters are similarly auto-populated based on the number of ranks you select with Altera's PCB simulations. You can update these autopopulated typical values, if more accurate dedicated simulation results are available.

Altera recommends performing board-level simulations to calculate the slew rate and ISI deltas that account for accurate board level effects for your board. You can use HyperLynx or similar simulators to obtain these simulation numbers. The default values have been computed using HyperLynx simulations for Altera boards with multiple DDR2 and DDR3 SDRAM slots.

For DQ and DQS ISI there is one textbox for the total ISI, which assumes symmetric setup and hold. For address and command, there are two textboxes: one for ISI on the leading edge, and one for the lagging edge, to allow for asymmetric ISI.

The wizard writes these parameters for the slew rates and the ISI into the **.sdc** and they are used during timing analysis.

Board Skews

Table 11–10 lists the types of board skew.

Table 11–10. Board Skews

Board Skew		Description
ALTMEMPHY	**UniPHY**	
Minimum CK/DQS skew to DIMM	—	The largest negative skew that exists between the CK signal and any DQS signal when arriving at any rank. This value affects the write leveling margin for DDR3 SDRAM DIMM interfaces in multiple chip select configurations only.
Maximum CK/DQS skew to DIMM	—	The maximum skew (or largest positive skew) between the CK signal and any DQS signal when arriving at any rank. This value affects the write leveling margin for DDR3 SDRAM DIMM interfaces in multiple chip select configurations.
Maximum skew between DIMMs	Maximum delay difference between DIMMs/devices	The largest skew or propagation delay between ranks (especially for different ranks in different slots). This value affects the resynchronization margin for DDR2 and DDR3 SDRAM interfaces in multiple chip select configurations.
Maximum skew within DQS group	Maximum skew within DQS group	The largest skew between DQ pins in a DQS group. This value affects the read capture and write margins for DDR2 and DDR3 SDRAM interfaces.
Maximum skew between DQS groups	Maximum skew between DQS groups	The largest skew between DQS signals in different DQS groups. This value affects the resynchronization margin in non-leveled memory interfaces such as DDR2 and DDR3 SDRAM.
Address and command to CK skew	Maximum delay difference between Address/Command and CK	The skew (or propagation delay) between the CK signal and the address and command signals. Positive values represent address and command signals that are longer than CK signals; negative values represent address and command signals that are shorter than CK signals. The Quartus II software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins for DDR2 and DDR3 SDRAM interfaces.
—	Maximum skew within Address/Command bus	The largest skew between the Address/Command signals.

Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time

This section describes how to measure eye reduction for address/command, DQ, and DQS.

Address/Command

The setup and hold times for address/command eye reduction is measured by comparing both the multirank and single rank address/command timing window as shown in Figure 11–24. Relative to the single rank address/command timing window, the reduction of the eye opening on the left side of the window denotes the setup time, while the reduction of the eye opening on the right side denotes the hold time.

To obtain the address/command eye reduction (setup time), measure the $V_{IL(AC)}$ or $V_{IH(AC)}$ difference between the single rank and multirank timing window, denoted by A in Figure 11–23 and Figure 11–24.

To obtain the address/command eye reduction (hold time), measure the $V_{IL(DC)}$ or $V_{IH(DC)}$ difference between the single rank and multirank timing window, denoted by B in Figure 11–23 and Figure 11–24.

Figure 11–23. Difference between Single Rank and Multirank Timing Window for Address/Command Eye Reduction (Setup)

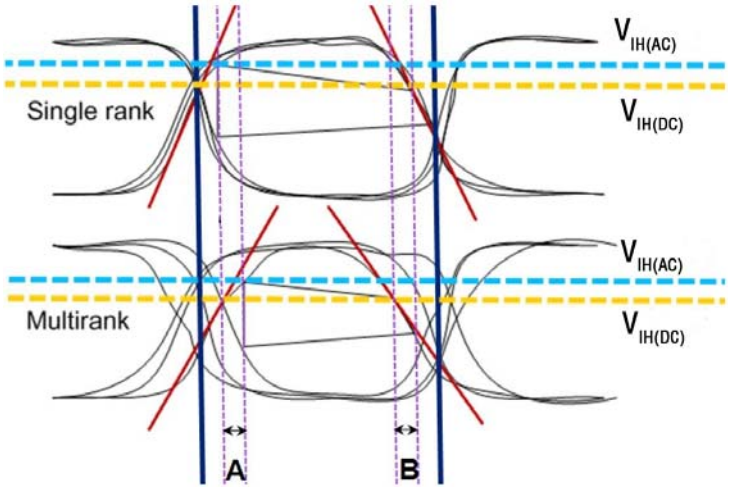
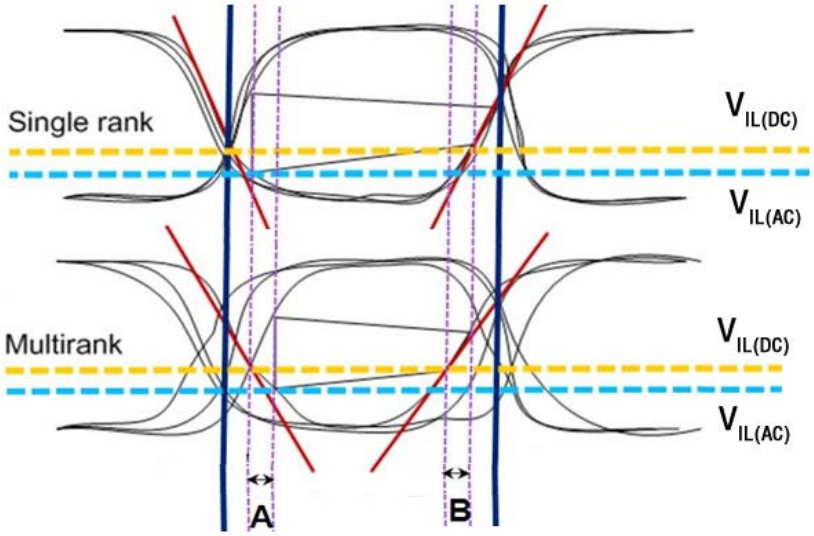


Figure 11–24. Difference between Single Rank and Multirank Timing Window for Address/Command Eye Reduction (Hold)



For signals with multiple loads, look at the measurements for all the target locations and pick the worst case eye width in all cases. For example, if pin A7 is the worst case eye width from pins A0 to A14, then the A7 measurement is used for the address signal. In general, look for eye reduction for the worst-pin in single-rank as compared to the worst-pin in multirank regardless of whether the pin is the same pin or a different pin.

DQ

The method to measure the DQ eye reduction is similar to the method you use to measure the command/address eye reduction. To measure the DQ eye reduction hold time, compare $V_{IH(DC)}$ or $V_{IL(DC)}$ between the single rank and multirank timing window. To measure the DQ eye reduction setup time, compare $V_{IH(AC)}$ or $V_{IL(AC)}$ between the single rank and multirank timing window.

DQS

DQS arrival time is the jitter before and after the single-rank timing window that you must enter in the GUI. The DQS arrival time is indicated by the DQS signal eye reduction between the signal rank system and multiple chip selects. Therefore, the method to measure the DQS arrival time is similar to the method you use to measure the command/address and DQ eye reduction.

ISI and Board Skew

Skews are systematic effects that are not time-varying, while ISI values are time- and pattern-dependent varying margin reduction.

In the **.sdc**, the address/command eye reduction is applied as an output delay constraint on the setup side and on the hold side, respectively.

For the write analysis, the eye reduction in DQ is applied as an output delay constraint, with half on the setup side and half on the output side. Similarly, the extra variation in the DQS is also applied as an output delay constraint with half

removed from the setup side and half removed from the hold side.

The board skews are included in the timing margin on the fully calibrated path such as write-leveling and resynchronization. Both the ISI and board skews values are entered to ensure that the interfaces are not over constraint.

Timing Deration Using the Excel-Based Calculator

To perform multiple chip select timing deration for other Altera devices (for example Stratix III and Cyclone III devices), use the Excel-based calculator, which is available from the Altera web site. You can also derate single chip-select cases using the Excel based calculator for devices that do not have the board settings panel provided you have the board simulation data to account for the ISI, skew and slew rate information.

The Excel-based calculator requires data like the slew rate, eye reduction, and the board skews of your multiple chip select system as inputs and outputs the final result based on built-in formula.

The calculator requires the Quartus II timing results (report DDR section) from the single rank version of your system. Two simulations are also required for the slew rate and ISI information required by the calculator: a baseline single rank version of your system and your multiple chip select system. The calculator uses the timing deltas of these simulation results for the signals of interest (DQ, DQS, CK/CK#, address and command, and CS). You must enter board skews for your specific board. The calculator outputs the final derated timing margins, which provides a full analysis of your multiple chip select system's performance.

The main assumption for this flow is that you have board trace models available for the single rank version of your system. If you have these board trace models, the Quartus II software automatically derates the effects for the single rank case correctly. Hence the Excel-based calculator provides the deration of the supported single-rank timing analysis, assuming that the single rank version has provided an accurate baseline.

You must ensure that the single rank board trace models are included in your Quartus II project so that the baseline timing analysis is accurate. If you do not have board trace models, follow the process described at the end of this section.

Before You Use the Excel-based Calculator for Timing Deration

Ensure you have the following items before you use the Excel-based calculator for timing deration:

1. A Quartus II project with your instantiated memory IP. Always use the latest version of the Quartus II software, for the most accurate timing models.
2. The board trace models for the single rank version of your system.

If you do not have board trace models, refer to "Using the Excel-based Calculator for Timing Deration (Without Board Trace Models)" on page 11–55.

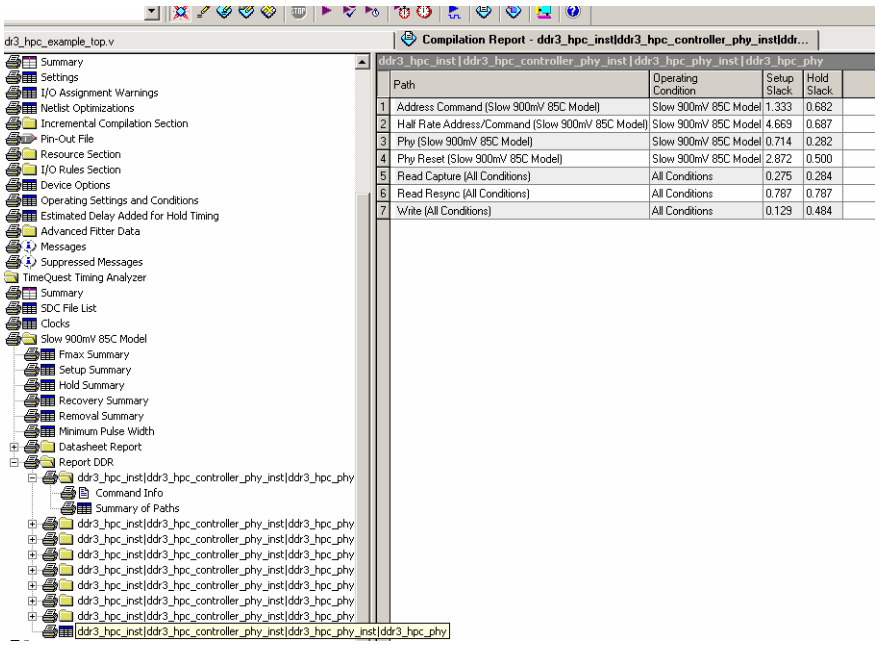
Using the Excel-Based Calculator

To obtain derated timing margins for multiple chip select designs using the Excel-based calculator, follow these steps:

1. Create a memory interface design in the Quartus II software.
2. Ensure board trace models are specified for your single rank system. Extract Quartus II reported timing margins into the Excel-based calculator.
3. Use the slow 85C model timing results (Figure 11–25).

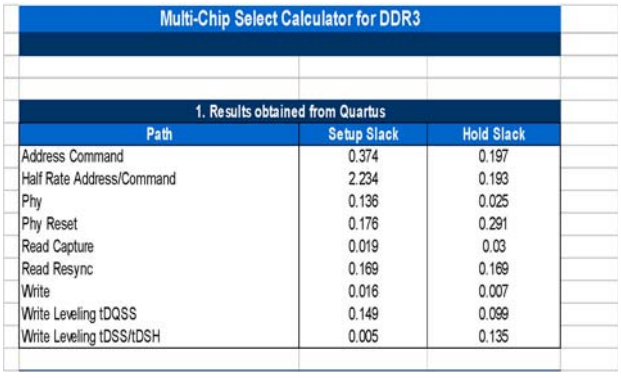
Use the worst-case values from all corners, which means that some values may come from different corners. For example, a setup value may come from the slow 85C model, while the hold value for the same metric may come from the fast 0C model. In most cases, using the slow 85C model should be accurate enough.

Figure 11–25. Quartus II Report DDR Section Timing Results for the Slow 85C Model



4. Enter the Report DDR results from Quartus II timing analysis into the Excel-based calculator (Figure 11–26).

Figure 11–26. Calculator



5. Perform PCB SI simulations to get the following values:
   - Single rank eye width and topology eye width for data, strobe, and address and command signals.
   - Multiple chip select topology slew rates for clock, address and command, and data and strobe signals.

   Table 11–11 lists the data rates and recommended stimulus patterns for various signals and memory interface types.

   Use a simulation tool (for example, HyperLynx), if you have access to the relevant input files that the tool requires, or use prelayout line simulations if the more accurate layout information is not available.

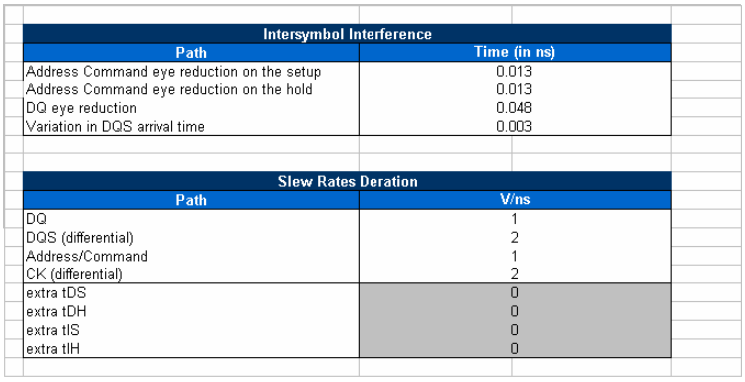
Table 11–11. Data Rates and Stimulus Patterns

Memory Interface	CLK and DQS Toggling Pattern (MHz)	DQ PRBS Pattern (MHz)	Address and Command PRBS Pattern (MHz)
DDR2 SDRAM (with a half-rate controller)	400	400	100
DDR2 SDRAM (with a full-rate controller)	300	300	150
DDR3 SDRAM (with a half-rate controller)	533	533	133

6. Calculate the deltas to enter into the Excel-based calculator. For example, if DQ for the single rank case is 853.682 ps and DQ for the dual rank case is 805.137 ps, enter 48 ps into the calculator (Figure 11–27).

   For signals with multiple loads, look at the measurements at all the target locations and pick the worst case eye width in all cases. For example, for the address bus, if A7 is the worst case eye width from pins A0 to A14, use that measurement for the address signal.

Figure 11–27. ISI and Slew Rate Values



Intersymbol Interference	
Path	Time (in ns)
Address Command eye reduction on the setup	0.013
Address Command eye reduction on the hold	0.013
DQ eye reduction	0.048
Variation in DQS arrival time	0.003

Slew Rates Deration	
Path	V/ns
DQ	1
DQS (differential)	2
Address/Command	1
CK (differential)	2
extra tDS	0
extra tDH	0
extra tIS	0
extra tIH	0

7. Enter the topology slew rates into the slew rate deration section. The calculator calculates the extra tDS, tDH, tIS, tIH values.

8. Obtain the board skew numbers for your PCB from either your board simulation or from your PCB vendor and enter them into the calculator (Figure 11–28).

Figure 11–28. Board Skew Values

Board Skews	
Path	Time (in ns)
Maximum skew between DIMMs	0.05
Minimum CK/DQS to one DIMM	0.01
Maximum CK/DQS to one DIMM	0.03

The Excel-based calculator then outputs the final derated numbers for your multiple chip select design.

Figure 11–29. Derated Setup and Hold Values

Final Multi Chip Select Results		
Path	Setup Slack	Hold Slack
Address Command	0.361	0.184
Half Rate Address/Command	2.228	0.187
Phy	0.136	0.025
Phy Reset	0.176	0.291
Read Capture	0.019	0.030
Read Resync	0.119	0.119
Write	-0.010	-0.019
Write Leveling tDQSS	0.126	0.076
Write Leveling tDSS/tDSH	-0.018	0.112

These values are an accurate calculation of your multiple chip select design timing, assuming the simulation data you provided is correct. In this example, there is negative slack on some of the paths, so this design does not pass timing. You have the following four options available:

- Try to optimize margins and see if it improves timing (for example modify address and command phase setting)
- Lower the frequency of your design
- Lower the loading (change the topology of your interface to lower the loading and skew)
- Use a faster DIMM

Using the Excel-based Calculator for Timing Deration (Without Board Trace Models)

If board trace models are not available for any of the signals of the single rank system, follow these steps:

1. Create a new Quartus II Project with the Stratix III or Cyclone III device that you are targeting and instantiate a High-Performance SDRAM Controller for your memory interface.
2. Do not enter the board trace models (assumes a 0-pf load) and compile the Quartus II design.
3. Enter the Report DDR setup and hold slack numbers into the Excel-based calculator.

4. Perform a prelayout line simulation of a 0-pf load simulation and obtain eye width and slew rate numbers. Perform multiple chip select simulations of your topology and use the Excel-based calculator.

Performing I/O Timing Analysis

For accurate I/O timing analysis, the Quartus II software must be made aware of the board trace and loading information. This information must be derived and refined during your PCB development process of pre-layout (line) and post-layout (board) simulations.

For external memory interfaces that use memory modules (DIMMs), the board trace and loading information must include the trace and loading information of the module in addition to the main and host platform, which you can obtain from your memory vendor.

You can use the following I/O timing analysis methods for your memory interface:

- Perform I/O Timing Analysis with 3rd Party Simulation Tools
- Perform Advanced I/O Timing Analysis with Board Trace Delay Model

Perform I/O Timing Analysis with 3rd Party Simulation Tools

Altera recommends that you perform I/O timing analysis using the 3rd party simulation tool flow because this flow involves post layout simulation that can capture more accurate I/O timing. This method is also easier because it only requires you to enter the slew rates, board skews, and ISI values in the ALTMEMPHY or UniPHY IP parameter editor.

To perform I/O timing analysis using 3rd party simulation tools, follow these steps:

1. Use a 3rd party simulation tool such as HyperLynx to simulate the full path for DQ, DQS, CK, Address, and Command signals.
2. Under the **Board Settings** tab of the ALTMEMPHY or UniPHY parameter editor, enter the slowest slew rate, ISI, and board skew values.

Perform Advanced I/O Timing Analysis with Board Trace Delay Model

You should use this method only if you are unable to perform post-layout simulation on the memory interface signals to obtain the slew rate parameters, and/or when no simulation tool is available.

To perform I/O timing analysis using board trace delay model, follow these steps:

1. After the instantiation is complete, analyze and synthesize your design.
2. Add pin and DQ group assignment by running the <*variation_name*>**_p0_pin_assignments.tcl script**.

   The naming of the pin assignment file may vary depending on the Quartus II software version that you are using.

3. Enter the pin location assignments.
4. Assign the virtual pins, if necessary.

5. Enter the board trace model information. To enter board trace model information, follow these steps:
   a. In the Pin Planner, select the pin or group of pins for which you want to enter board trace parameters.
   b. Right-click and select **Board Trace Model**.
6. Compile your design. To compile the design, on the Processing menu, click **Start Compilation**.
7. After successfully compiling the design, perform timing analysis in the TimeQuest timing analyzer. To perform timing analysis, follow these steps:
   a. In the Quartus II software, on the Tools menu, click **TimeQuest Timing Analyzer**.
   b. On the **Tasks** pane, click **Report DDR**.
   c. On the **Report** pane, select **Advanced I/O Timing>Signal Integrity Metrics**.
   d. In the **Signal Integrity Metrics** window, right-click and select **Regenerate** to regenerate the signal integrity metrics.
   e. In the **Signal Integrity Metrics** window, note the 10–90% rise time (or fall time if fall time is worse) at the far end for CK/CK#, address, and command, DQS/DQS#, and DQ signals.
   f. In the DDR3 SDRAM controller parameter editor, in the **Board Settings** tab, type the values you obtained from the signal integrity metrics.
   g. For the board skew parameters, set the maximum skew within DQS groups of your design. Set the other board parameters to 0 ns.
   h. Compile your design.

Document Revision History

Table 11–12 lists the revision history for this document.

Table 11–12. Document Revision History

Date	Version	Changes
November 2012	4.2	Changed chapter number from 10 to 11.
June 2012	4.1	■ Added Feedback icon.
November 2011	4.0	■ Added Arria V and Cyclone V information. ■ Added Performing I/O Timing Analysis section. ■ Added Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time section.
June 2011	3.0	Updated for 11.0 release.
December 2010	2.1	Added Arria II GZ and Stratix V, updated board skews table.
July 2010	2.0	Added information about UniPHY-based IP and controllers.
January 2010	1.2	Corrected minor typos.
December 2009	1.1	Added *Timing Deration* section.
November 2009	1.0	First published.

12. Debugging Memory IP

EMI_DG_011-4.2

This chapter describes the process of debugging hardware and the tools to debug any external memory interface. The concepts discussed can be applied to any IP but focus on the debug of issues using the Altera® DDR, DDR2, DDR3, QDRII, QDRII+, and RLDRAM II IP.

Increases in external memory interface frequency results in the following issues that increase the challenges of debugging interfaces:

- More complex memory protocols
- Increased features and functionality
- More critical timing
- Increased complexity of calibration algorithm

Before the in-depth debugging of any issue, gather and confirm all information regarding the issue.

Memory IP Debugging Issues

Debug issues may not be directly related to interface operation. Issues can also arise at the Quartus® II Fitter stage, or complex designs may have timing analysis issues.

Memory debugging issues can be catagorized as follows:

- Resource and Planning Issues
- Interface Configuration Issues
- Functional Issues
- Timing Issues

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Resource and Planning Issues

Typically, single stand-alone interfaces should not present any Quartus II Fitter or timing issues. You may find that fitter, timing, and hardware operation can sometimes become a challenge, as multiple interfaces are combined into a single project, or as the device utilization increases. In such cases, the interface configuration is not the issue, the placement and total device resource requirements create problems.

Resource Issue Evaluation

External memory interfaces typically require the following resource types, which you must consider when trying to manually place, or perhaps use additional constraints to force the placement or location of external memory interface IP:

- Dedicated IOE DQS group resources and pins
- Dedicated DLL resources
- Specific PLL resources
- Specific global, regional, and dual-regional clock net resources

Dedicated IOE DQS Group Resources and Pins

Fitter issues can occur with even a single interface, if you do not size the interface to fit within the specified constraints and requirements. A typical requirement includes containing assignments for the interface within a single bank or possibly side of the chosen device.

Such a constraint requires that the chosen device meets the following conditions:

- Sufficient DQS groups and sizes to support the required number of common I/O (CIO) or separate I/O (SIO) data groups.
- Sufficient remaining pins to support the required number of address, command, and control pins.

Failure to evaluate these fundamental requirements can result in suboptimal interface design, if the chosen device cannot be modified. The resulting wraparound interfaces or suboptimal pseudo read and write data groups artificially limit the maximum operating frequency.

Multiple blocks of IP further complicate the issue, if other IP has either no specified location constraints or incompatible location constraints.

The Quartus II fitter may first place other components in a location required by your memory IP, then error at a later stage because of an I/O assignment conflict between the unconstrained IP and the constrained memory IP.

Your design may require that one instance of IP is placed anywhere on one side of the device, and that another instance of IP is placed at a specific location on the same side.

While the two individual instances may compile in isolation, and the physical number of pins may appear sufficient for both instances, issues can occur if the instance without placement constraints is placed before the instance with placement constraints.

In such circumstances, Altera recommends manually placing each individual pin, or at least try using more granular placement constraints.

For more information about the pin number and DQS group capabilities of your chosen device, refer to device data sheets or the Quartus II pin planner.

Dedicated DLL Resources

Altera devices typically use DLLs to enhance data capture at the FPGA.

While multiple external memory interfaces can usually share DLL resources, fitter issues can occur when there is insufficient planning before HDL coding. If DLL sharing is required, Altera gives the following recommendations for each instance of the IP that shares the DLL resources:

- Must have compatible DLL requirements—same frequency and mode.
- Exports its autogenerated DLL instance out of its own dedicated PHY hierarchy and into the top-level design file. This procedure allows easy comparison of the generated DLL's mode, and allows you to explicitly show the required DLL sharing between two IP blocks in the HDL

The Quartus II fitter does not dynamically merge DLL instances.

Specific PLL Resources

When only a single interface resides on one side or one quadrant of a device, PLL resources are typically not an issue. However if multiple interfaces or IP are required on a single side or quadrant, consider the specific PLL used by each IP, and the sharing of any PLL resources.

The Quartus II software automerges PLL resources, but not for any dynamically controlled PLL components. Use the following PLL resource rules:

- Ensure that the PLL located in the same bank or side of the device is available for your memory controller.
- If multiple PLLs are required for multiple controllers that cannot be shared, ensure that enough PLL resources are available within each quadrant to support your interface number requirements.
- Try to limit multiple interfaces to a single quadrant. For example, if two complete same size interfaces can fit on a single side of the device, constrain one interface entirely in one bank of that side, and the other controller in the other bank.

For more information about using multiple PHYs or controllers, refer to the design tutorials on the List of designs using Altera External Memory IP page of the Altera Wiki website.

Specific Global, Regional and Dual-Regional Clock Net Resources

Memory PHYs typically have specific clock resource requirements for each PLL clock output. For example because of characterization data, the PHY may require that the `phy_clk` is routed on a global clock net. The remaining clocks may all be routed on a global or a regional clock net. However, they must all be routed on the same type. Otherwise, the operating frequency of the interface is lowered, because of the increased uncertainty between two different types of clock nets. The design may still fit, but not meet timing.

Planning Issue Evaluation

Plan the total number of DQS groups and total number of other pins required in your shared area. Use the Pin Planner to assist with this activity.

Decide which PLLs or clock networks can be shared between IP blocks, then ensure that sufficient resources are available. For example, if an IP core requires a regional clock network, a PLL located on the opposite side of the device cannot be used.

Calculate the number of total clock networks and types required when trying to combine multiple instances of IP.

You must understand the number of quadrants that the IP uses and if this number can be reduced. For example, an interface may be autoplaced across an entire side of the device, but may actually be constrained to fit in a single bank.

Optimizing the physical placement ensures that when possible, regional clock networks are used as opposed to dual-regional clock networks, hence clock net resources are maintained and routing is simplified.

As device utilization increases, the Quartus II software may have difficulty placing the core. To optimize design utilization, follow these steps:

- Review any fitter warning messages in multiple IP designs to ensure that clock networks or PLL modes are not modified to achieve the desired fit.
- Use the Quartus II Fitter resource section to compare the types of resources used in a successful standalone IP implementation to those used in an unreliable multiple IP implementation.
- Use this information to better constrain the project to achieve the same results as the standalone project.
- Use the Chip Planner (Floorplan and Chip Editor) to compare the placement of the working stand-alone design to the multiple IP project. Then use LogicLock™ or Design Partitions to better guide the Quartus II software to the required results.
- When creating LogicLock regions, ensure that they encompass all required resources. For example, if constraining the read and write datapath hierarchy, ensure that your LogicLock region includes the IOE blocks used for your datapath pin out.

Interface Configuration Issues

This topic describes the performance (f_{MAX}), efficiency (latency and transaction efficiency) and bottleneck (the limiting factor) issues that you can encounter in any design.

Performance Issues

There are a large number of interface combinations and configurations possible in an Altera design, therefore it is impractical for Altera to explicitly state the achievable f_{MAX} for every combination. Altera seeks to provide guidance on typical performance, but this data is subject to memory component timing characteristics, interface widths, depths directly affecting timing deration requirements, and the achieved skew and timing numbers for a specific PCB.

FPGA timing issues should generally not be affected by interface loading or layout characteristics. In general, the Altera performance figures for any given device family and speed-grade combination should usually be achievable.

To resolve FPGA (PHY and PHY reset) timing issues, refer to the *Analyzing Timing of Memory IP* chapter.

Achievable interface timing (address and command, half-rate address and command, read and write capture) is directly affected by any layout issues (skew), loading issues (deration), signal integrity issues (crosstalk timing deration), and component speed grades (memory timing size and tolerance). Altera performance figures are typically stated for the default (single rank, unbuffered DIMM) case. Altera provides additional expected performance data where possible, but the f_{MAX} is not achievable in all configurations. Altera recommends that you optimize the following items whenever interface timing issues occur:

- Improve PCB layout tolerances
- Use a faster speed grade of memory component
- Ensure that the interface is fully and correctly terminated
- Reduce the loading (reduce the deration factor)

Bottleneck and Efficiency Issues

Depending on the transaction types, efficiency issues can exist where the achieved data rate is lower than expected. Ideally, these issues should be assessed and resolved during the simulation stage because they are sometimes impossible to solve later without rearchitecting the product.

Any interface has a maximum theoretical data rate derived from the clock frequency, however, in practise this theoretical data rate can never be achieved continuously due to protocol overhead and bus turnaround times.

Simulate your desired configuration to ensure that you have specified a suitable external memory family and that your chosen controller configuration can achieve your required bandwidth.

Efficiency can be assessed in several different ways, and the primary requirement is an achievable continuous data rate. The local interface signals combined with the memory interface signals and a command decode trace should provide adequate visibility of the operation of the IP to understand whether your required data rate is sufficient and the cause of the efficiency issue.

To show if under ideal conditions the required data rate is possible in the chosen technology, follow these steps:

1. Use the memory vendors own testbench and your own transaction engine.
2. Use either your own driver, or modify the provided example driver, to replicate the transaction types typical of your system.
3. Simulate this performance using your chosen memory controller and decide if the achieved performance is still acceptable.

Observe the following points that may cause efficiency or bottleneck issues at this stage:

- Identify the memory controller rate (full, half, or quarter) and commands, which may take two or four times longer than necessary
- Determine whether the memory controller is starved for data by observing the appropriate request signals.
- Determine whether the memory controller processor transactions at a rate sufficient to meet throughput requirements by observing appropriate signals, including the local ready signal.

Altera has several versions and types of memory controller, and where possible you can evaluate different configurations based on the results of the first tests.

Consider using either a faster interface, or a different memory type to better align your data rate requirements to the IP available directly from Altera.

Altera also provides stand-alone PHY configurations so that you may develop custom controllers or use third-party controllers designed specifically for your requirements.

Functional Issues

This topic discusses functional issues that occur at all frequencies (using the same conditions) and are not altered by speed grade, temperature, or PCB changes.

Functional Issue Evaluation

Functional issues should be evaluated using functional simulation.

The Altera IP includes the option to autogenerate a testbench specific to your IP configuration, which provides an easy route to functional verification.

The following issues should be considered when trying to debug functional issues in a simulation environment.

Correct Combination of the Quartus II Software and ModelSim-Altera Device Models

When running any simulations, ensure that you are using the correct combination of the Quartus II software and device models. Altera only tests each release of software and IP with the aligned release of device models. Failure to use the correct RTL and model combination may result in unstable simulation environments.

The ModelSim®-Altera edition of the ModelSim simulator comes precompiled with the Altera device family libraries included. You must always install the correct release of ModelSim-Altera to align with your Quartus II software and IP release.

If you are using a full version of ModelSim-SE or PE, or any other supported simulation environment, ensure that you are compiling the current Quartus II supplied libraries. These libraries are located in the <*Quartus II install path*>**/quartus/eda/sim_lib/** directory.

Altera IP Memory Model

Altera memory IP autogenerates a generic simplified memory model that works in all cases. This simple read and write model is not designed or intended to verify all entered IP parameters or transaction requirements.

The Altera-generated memory model may be suitable to evaluate some limited functional issues, but it does not provide comprehensive functional simulation.

Vendor Memory Model

Contact the memory vendor directly as many additional models are available from the vendors support system.

When using memory vendor models, ensure that the model is correctly defined for the following characteristics:

- Speed grade
- Organization
- Memory allocation
- Maximum memory usage
- Number of ranks on a DIMM
- Buffering on the DIMM
- ECC

Refer to the **readme.txt** file supplied with the memory vendor model, for more information about how to define this information for your configuration.

During simulation vendor models output a wealth of information regarding any device violations that may occur because of incorrectly parameterized IP.

Refer to Transcript Window Messages, for more information.

Out of PC Memory Issues

If you are running the ModelSim-Altera simulator, the limitation on memory size, may mean that you have insufficient memory to run your simulation. Or, if you are using a 32-bit operating system, your PC may have insufficient memory.

Typical simulation tool errors include: "Iteration limit reached" or "Error out of memory".

When using either the Altera generic memory model, or a vendor specific model quite large memory depths can be required if you do not specify your simulation carefully.

For example, if you simulate an entire 4-GB DIMM interface, the hardware platform that performs that simulation requires at least this amount of memory just for the simulation contents storage.

Refer to Memory Allocation and Max Memory Usage in the vendor's **readme.txt** files for more information.

Transcript Window Messages

When debugging a functional issue in simulation, vendor models typically provide a much more detailed checks and feedback regarding the interface and their operational requirements than the Altera generic model.

In general, you should use a vendor-supplied model whenever one is available. Consider using second-source vendor models in preference to the Altera generic model.

Many issues can be traced to incorrectly configured IP for the specified memory components. Component data sheets usually contain settings information for several different speed grades of memory. Be aware data sheet specify parameters in fixed units of time, frequencies, or clock cycles.

The Altera generic memory model always matches the parameters specified in the IP, as it is generated using the same engine. Because vendor models are independent of the IP generation process, they offer a more robust IP parameterization check.

During simulation, review the transcript window messages and do not rely on the Simulation Passed message at the end of simulation. This message only indicates that the example driver successfully wrote and then read the correct data for a single test cycle.

Even if the interface functionally passes in simulation, the vendor model may report operational violations in the transcript window. These reported violations often specifically explain why an interface appears to pass in simulation, but fails in hardware.

Vendor models typically perform checks to ensure that the following types of parameters are correct:

- Burst length
- Burst order
- tMRD
- tMOD
- tRFC
- tREFPDEN
- tRP
- tRAS
- tRC
- tACTPDEN
- tWR
- tWRPDEN
- tRTP
- tRDPDEN
- tINIT
- tXPDLL

- tCKE
- tRRD
- tCCD
- tWTR
- tXPR
- PRECHARGE
- CAS length
- Drive strength
- AL
- tDQS
- CAS_WL
- Refresh
- Initialization
- tIH
- tIS
- tDH
- tDS

If a vendor model can verify all these parameters are compatible with your chosen component values and transactions, it provides a specific insight into hardware interface failures.

Simulation

Passing simulation means that the interface calibrates and successfully completes a single test complete cycle without asserting pass not fail (`pnf`). It does not take into account any warning messages that you may receive during simulation. If you are debugging an interface issue, review and, if necessary, correct any warning messages from the transcript window before continuing.

Modifying the Example Driver to Replicate the Failure

Often during debugging, you may discover that the example driver design works successfully, but that your custom logic is observing data errors. This information indicates that the issue is either:

- Related to the way that the local interface transactions are occurring. Altera recommends you probe and compare using the SignalTap™ II analyzer.
- Related to the types or format of transactions on the external memory interface. Altera recommends you modify the example design to replicate the issue.

Typical issues on the local interface side include:

- Incorrect local address to memory address translation causing the word order to be different than expected. Refer to *Burst Definition* in your memory vendor data sheet.

- Incorrect timing on the local interface. When your design requests a transaction, the local side must be ready to service that transaction as soon as it is accepted without any pause.

  For more information, refer to the *Avalon® Interface Specification*.

The default example driver only performs a limited set of transaction types, consequently potential bus contention or preamble and postamble issues can often be masked in its default operation. For successful debugging, isolate the custom logic transaction types that are causing the read and write failures and modify the example driver to demonstrate the same issue. Then, you can try to replicate the failure in RTL simulation with the modified driver.

An issue that you can replicate in RTL simulation indicates a potential bug in the IP. You should recheck the IP parameters. An issue that you can not replicate in RTL simulation indicates a timing issue on the PCB. You can try to replicate the issue on an Altera development platform to rule out a board issue.

Ensure that all PCB timing, loading, skew, and deration information is correctly defined in the Quartus II software, as the timing report is inaccurate if this initial data is not correct.

Functional simulation allows you to identify any issues with the configuration of either the Altera memory controller and or PHY. You can then easily check the operation against both the memory vendor data sheet and the respective JEDEC specification. After you resolve functional issues, you can start testing hardware.

For more information about simulation, refer to the *Simulating Memory IP* chapter.

Timing Issues

The Altera PHY and controller combinations autogenerate timing constraint files to ensure that the PHY and external interface are fully constrained and that timing is analyzed during compilation. However, timing issues can still occur. This topic discusses how to identify and resolve any timing issues that you may encounter.

Timing Issue Characteristics

Timing issues typically fall into two distinct categories:

- FPGA core timing reported issues
- External memory interface timing issues in a specific mode of operation or on a specific PCB

TimeQuest reports timing issues in two categories: core to core and core to IOE transfers. These timing issues include the PHY and PHY reset sections in the TimeQuest Report DDR subsection of timing analysis. External memory interface timing issues are specifically reported in the TimeQuest Report DDR subsection, excluding the PHY and PHY reset. The Report DDR PHY and PHY reset sections only include the PHY, and specifically exclude the controller, core, PHY-to-controller and local interface. Quartus II timing issues should always be evaluated and corrected before proceeding to any hardware testing.

PCB timing issues are usually Quartus II timing issues, which are not reported in the Quartus II software, if incorrect or insufficient PCB topology and layout information is not supplied. PCB timing issues are typically characterized by calibration failure, or failures during user mode when the hardware is heated or cooled. Further PCB timing issues are typically hidden if the interface frequency is lowered.

Timing Issue Evaluation

Try to fix and identify timing issues in the Quartus II software. Consider the following issues when resolving timing issues.

FPGA Timing Issues

In general, you should not have any timing issues with Altera-provided IP unless you running the IP outside of Altera's published performance range or are using a version of the Quartus II software with preliminary timing model support for new devices. However, timing issues can occur in the following circumstances:

- The **.sdc** files are incorrectly added to the Quartus II project
- Quartus II analysis and synthesis settings are not correct
- Quartus II Fitter settings are not correct

For all of these issues, refer to the correct user guide for more information about recommended settings and follow these steps:.

1. Ensure that the IP generated **.sdc** files are listed in the Quartus II TimeQuest Timing Analyzer files to include in the project window.
2. Ensure that **Analysis and Synthesis Settings** are set to **Optimization Technique Speed**.
3. Ensure that **Fitter Settings** are set to **Fitter Effort Standard Fit**.
4. Use **TimeQuest Report Ignored Constraints**, to ensure that **.sdc** files are successfully applied.
5. Use **TimeQuest Report Unconstrained Paths**, to ensure that all critical paths are correctly constrained.

More complex timing issues can occur if any of the following conditions are true:

- The design includes multiple PHY or core projects
- Devices where the resources are heavily used
- The design includes wide, distributed, maximum performance interfaces in large die sizes

Any of these conditions can lead to suboptimal placement results when the PHY or controller are distributed around the FPGA. To evaluate such issues, simplify the design to just the autogenerated example top-level file and determine if the core meets timing and you see a working interface. Failure implies that a more fundamental timing issue exists. If the standalone design passes core timing, evaluate how this placement and fit is different than your complete design.

Use LogicLock regions, or design partitions to better define the placement of your memory controllers. When you have your interface standalone placement, repeat for additional interfaces, combine, and finally add the rest of your design.

Additionally, use fitter seeds and increase the placement and router effort multiplier.

External Memory Interface Timing Issues

External memory interface timing issues are not directly related to the FPGA timing but are actually derived from the FPGA input and output characteristics, PCB timing, and the memory component characteristics.

The FPGA input and output characteristics tend to be a predominately fixed value, as the IOE structure of the devices is fixed. Optimal PLL characteristics and clock routing characteristics do have an effect. Assuming the IP is correctly constrained with the autogenerated assignments, and you follow implementation rules, the design should reach the stated performance figures.

The memory component characteristics are fixed for any given component or DIMM. However, consider using faster components or DIMMs in marginal cases when PCB skew may be suboptimal, or your design includes multiple ranks when deration may be causing read capture or write timing challenges. Using faster memory components typically reduces the memory data output skew and uncertainty easing read capture, and lowering the memory's input setup and hold requirement, which eases write timing.

Increased PCB skew reduces margins on address, command, read capture and write timing. If you are narrowly failing timing on these paths, consider reducing the board skew (if possible), or using faster memory. Address and command timing typically requires you to manually balance the reported setup and hold values with the dedicated address and command phase in the IP.

Refer to the respective IP user guide for more information.

Multiple-slot multiple-rank UDIMM interfaces can place considerable loading on the FPGA driver. Typically a quad rank interface can have thirty-six loads. In multiple-rank configurations, Altera's stated maximum data rates are not likely to be achievable because of loading deration. Consider using different topologies, for example registered DIMMs, so that the loading is reduced.

Deration because of increased loading, or suboptimal layout may result in a lower than desired operating frequency meeting timing. You should close timing in the Quartus II software using your expected loading and layout rules before committing to PCB fabrication.

Ensure that any design with an Altera PHY is correctly constrained and meets timing in the Quartus II software. You must address any constraint or timing failures before testing hardware.

For more information about timing constraints, refer to the *Analyzing Timing of Memory IP* chapter.

Verifying Memory IP Using the SignalTap II Logic Analyzer

The SignalTap II logic analyzer shows read and write activity in the system.

For more information about using the SignalTap II logic analyzer, refer to *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Software Handbook*

To add the SignalTap II logic analyzer, follow these steps:

1. On the Tools menu click **SignalTap II Logic Analyzer**.
2. In the **Signal Configuration** window next to the **Clock** box, click **…** (Browse Node Finder).
3. Type the memory interface system clock (typically `*phy_clk`) in the **Named** box, for **Filter** select **SignalTap II: pre-synthesis** and click **List**.
4. Select the memory interface system clock (**<***variation name***>_example_top|<***variation name***>:<***variation name***>_inst|<***variation name***>_controller_phy:<***variation name***>_controller_phy_inst|phy_clk|phy_clk)** in **Nodes Found** and click **>** to add the signal to **Selected Nodes**.
5. Click **OK**.
6. Under Signal Configuration, specify the following settings:
   - For **Sample depth**, select **512**
   - For **RAM type**, select **Auto**
   - For **Trigger flow control**, select **Sequential**
   - For **Trigger position**, select **Center trigger position**
   - For **Trigger conditions**, select **1**
7. On the Edit menu, click **Add Nodes**.
8. Search for specific nodes by typing `*local*` in the **Named** box, for **Filter** select **SignalTap II: pre-synthesis** and click **List**.

9. Select the following nodes in **Nodes Found** and click **>** to add to **Selected Nodes**:
   - **local_address**
   - **local_rdata**
   - **local_rdata_valid**
   - **local_read_req**
   - **local_ready**
   - **local_wdata**
   - **local_wdata_req**
   - **local_write_req**
   - **pnf**
   - **pnf_per_byte**
   - **test_complete** (trigger)
   - **ctl_cal_success**
   - **ctl_cal_fail**
   - **ctl_wlat**
   - **ctl_rlat**

   ☞ Do not add any memory interface signals to the SignalTap II logic analyzer. The load on these signals increases and adversely affects the timing analysis.

10. Click **OK**.
11. To reduce the SignalTap II logic size, turn off **Trigger Enable** on the following bus signals:
    - `local_address`
    - `local_rdata`
    - `local_wdata`
    - `pnf_per_byte`
    - `ctl_wlat`
    - `ctl_rlat`
12. Right-click **Trigger Conditions** for the `test_complete` signal and select **Rising Edge**.
13. On the File menu, click **Save**, to save the SignalTap II **.stp** file to your project.

    ☞ If you see the message **Do you want to enable SignalTap II file "stp1.stp" for the current project**, click **Yes**.

14. Once you add signals to the SignalTap II logic analyzer, recompile your design, on the Processing menu, click **Start Compilation**.

15. When the design compiles, ensure that TimeQuest timing analysis passes successfully. In addition to this FPGA timing analysis, check your PCB or system SDRAM timing. To run timing analysis, run the ***_phy_report_timing.tcl** script.
    a. On the Tools menu, click **Tcl Scripts**.
    b. Select <*variation name*>**_phy_report_timing.tcl** and click **Run**.
16. Connect the development board to your computer.
17. On the Tools menu, click **SignalTap II Logic Analyzer**.
18. Add the correct <*your project name*>**.sof** file to the SOF Manager:
    a. Click **...** to open the **Select Program Files** dialog box.
    b. Select <*your project name*>**.sof**.
    c. Click **Open**.
    d. To download the file, click the **Program Device** button.
19. When the example design including SignalTap II successfully downloads to your development board, click **Run Analysis** to run once, or click **Autorun Analysis** to run continuously.

Monitoring Signals with the SignalTap II Logic Analyzer

The following sections detail the memory controller signals you should consider analyzing for different memory interfaces. The list is not exhaustive, but is a starting point.

For a description of each signal, refer to *Volume 3: Reference Material* of the *External Memory Interface Handbook*.

DDR, DDR2, and DDR3 ALTMEMPHY Designs

Monitor the following signals for DDR, DDR2, and DDR3 SDRAM ALTMEMPHY designs:

- `Local_* -example_driver` (all the local interface signals)
- `Pnf -example_driver`
- `Pnf_per_byte -example_driver`
- `Test_complete -example_driver`
- `Test_status -example_driver`
- `Ctl_cal_req -phy_inst`
- `Ctl_init_fail -phy_inst`
- `Ctl_init_success -phy_inst`
- `Ctl_cal_fail -phy_inst`
- `Ctl_cal_success -phy_inst`
- `Cal_codvw_phase * -phy_inst`
- `Cal_codvw_size * -phy_inst`

- Codvw_trk_shift * -phy_inst
- Ctl_rlat * -phy_inst
- Ctl_wlat * -phy_inst
- Locked -altpll_component
- Phasecounterselect * -altpll_component
- Phaseupdown -altpll_component
- Phasestep -altpll_component
- Phase_done -altpll_component
- Flag_done_timeout -seq_inst:ctrl
- Flag_ack_timeout -seq_inst:ctrl
- Proc_ctrl.command_err -seq_inst:ctrl
- Proc_ctrl.command_result * -seq_inst:ctrl
- dgrb_ctrl.command_err -seq_inst:ctrl
- dgrb_ctrl.command_result * -seq_inst:ctrl
- dgwb_ctrl.command_err -seq_inst:ctrl
- dgwb_ctrl.command_result * -seq_inst:ctrl
- admin_ctrl.command_err -seq_inst:ctrl
- admin_ctrl.command_result * -seq_inst:ctrl
- setup_ctrl.command_err -seq_inst:ctrl
- setup_ctrl.command_result * -seq_inst:ctrl
- state.s_phy_initialise -seq_inst:ctrl
- state.s_init_dram -seq_inst:ctrl
- state.s_write_ihi -seq_inst:ctrl
- state.s_cal -seq_inst:ctrl
- state.s_write_btp -seq_inst:ctrl
- state.s_write_mtp -seq_inst:ctrl
- state.s_read_mtp -seq_inst:ctrl
- state.s_rrp_reset -seq_inst:ctrl
- state.s_rrp_sweep -seq_inst:ctrl
- state.s_rrp_seek -seq_inst:ctrl
- state.s_rdv -seq_inst:ctrl
- state.s_poa -seq_inst:ctrl
- state.s_was -seq_inst:ctrl
- state.s_adv_rd_lat -seq_inst:ctrl
- state.s_adv_wr_lat -seq_inst:ctrl

- state.s_prep_customer_mr_setup -seq_inst:ctrl
- state.s_tracking_setup -seq_inst:ctrl
- state.s_tracking -seq_inst:ctrl
- state.s_reset -seq_inst:ctrl
- state.s_non_operational -seq_inst:ctrl
- state.s_operational -seq_inst:ctrl
- dqs_delay_ctrl_export * -phy_inst
- * = Disable Trigger Enable

UniPHY Designs

Monitor the following signals for UniPHY designs:

- avl_addr
- avl_rdata
- avl_rdata_valid
- avl_read_req
- avl_ready
- avl_wdata
- avl_write_req
- fail
- pass
- afi_cal_fail
- afi_cal_success
- test_complete
- be_reg (QDRII only)
- pnf_per_bit
- rdata_reg
- rdata_valid_reg
- data_out
- data_in
- written_data_fifo|data_out
- usequencer|state*
- usequencer|phy_seq_rdata_valid
- usequencer|phy_seq_read_fifo_q
- usequencer|phy_read_increment_vfifo*
- usequencer|phy_read_latency_counter
- uread_datapath|afi_rdata_en

- uread_datapath|afi_rdata_valid
- uread_datapath|ddio_phy_dq
- qvld_wr_address*
- qvld_rd_address*

Hardware Debugging Guidelines

Before starting to debug, confirm the design followed the Altera recommended design flow.

Refer to the *Design Flow* chapter in volume 1 of the *External Memory Interface Handbook*.

Always keep a record of tests, to avoid repeating the same tests later. To start debugging the design, perform the following initial steps.

Create a Simplified Design that Demonstrates the Same Issue

To help debugging create a simple design that replicates the issue. A simple design compiles faster and is much easier to understand. Altera's external memory interface IP generates an example top-level file that is ideal for debugging. The example top-level file uses all the same parameters, pin-outs, and so on.

Measure Power Distribution Network

Ensure you take measurements of the various power supplies on their hardware development platform over a suitable time base and with a suitable trigger using an appropriate probe and grounding scheme. In addition, take the measurements directly on the pins or vias of the devices in question, and with the hardware operational.

Measure Signal Integrity and Setup and Hold Margin

Measure the signals on their PCB to ensure that everything looks correct. This information can be vital. When measuring any signal, consider the edge rate of the signal, not just its frequency. Modern FPGA devices have very fast edge rates, therefore you must use a suitable oscilloscope, probe, and grounding scheme when you measure the signals.

You can take measurements to capture the setup and hold time of key signal classes with respect to their clock or strobe. Ensure that the measured setup and hold margin is at least better than that reported in the Quartus II software. A worse margin indicates that a timing discrepancy exists somewhere in the project. However, this timing issue may not be the cause of your problem.

Vary Voltage

Try and vary the voltage of your system, if you suspect a marginality issue. Increasing the voltage typically causes devices to operate faster and also usually provides increased noise margin.

Use Freezer Spray and Heat Gun

If you have an intermittent marginal issue, cool or heat the interface to try and stress the issue. Cooling down ICs causes them to run faster, which makes timing easier. Conversely heating up ICs causes them to run slower, which makes timing more difficult.

If cooling or heating fixes the issue, you are probably looking at a timing issue.

Operate at a Lower Speed

Test the interface at a lower speed. If the interface works, the interface is correctly pinned out and functional. However, if the interface fails at a lower speed, determine if the test is valid. Many high-speed memory components have a minimal operating frequency, or require subtly different configurations when operating at a lower speeds.

For example, DDR, DDR2, or DDR3 SDRAM typically requires modification to the following parameters if you want operate the interface a lower speeds:

- t_{MRD}
- t_{WTR}
- CAS latency and CAS write latency

Find Out if the Issue Exists in Previous Versions of Software

Hardware that works before an update to either the Quartus II software or the memory IP indicates that the development platform is not the issue. However, the previous generation IP may be less susceptible to a PCB issue, masking the issue.

Find out if the Issue Exists in the Current Version of Software

Designs are often tested using previous generations of Altera software or IP. Projects do not always get upgraded for the following reasons:

- Multiple engineers are on the same project. To ensure compatibility, a common release of Altera software is used by all engineers for the duration of the product development. The design may be several releases behind the current Quartus II software version.
- Many companies delay before adopting a new release of software so that they can first monitor Internet forums to get a feel for how successful other users say the software is.
- Many companies never use the latest version of any software, preferring to wait until the first service pack is released that fixes the primary issues.
- Some users may only have a license for the older version of the software and can only use that version until their company makes the financial decision to upgrade.
- The local interface specification from Altera IP to the customer's logic sometimes changes from software release to software release. If you have already spent resources designing interface logic, you may be reluctant to repeat this exercise. If a block of code is already signed off, you may be reluctant to modify it to upgrade to newer IP from Altera..

In all of these scenarios, you must determine if the issue still exists in the latest version of the Altera software. Bugs are fixed and enhancements are added to the Altera IP every release. Depending on the nature of the bug or enhancement, it may not always be clearly documented in the release notes.

Finally, if the latest version of the software resolves the issue, it may be easier to debug the version of software that you are using.

Try A Different PCB

If you are using the same Altera IP on a number of different hardware platforms; find out if the issue occurs on all of these hardware platforms, or just one. Multiple instances of the same PCB, or multiple instances of the same interface, on physically different hardware platforms may exhibit different behavior. You can determine if the configuration is fundamentally not working, or if some form of marginality is involved in the issue.

Issues are often reported on the alpha build of a development platform. These are produced in very limited numbers and often have received limited BBT (bare board testing), or FT (functional testing). Hence, these early boards are often more unreliable than production quality PCBs.

Additionally, if the IP is from a previous project to help save development resources, find out if this specific IP configuration works on a previous platform.

Try Other Configurations

Designs are typically quite large, using multiple blocks of IP in many different combinations. Find out if any other configurations work correctly on the development platform. The full project may have multiple external memory controllers in the same device, or may have configurations where only half the memory width or frequency is required. Find out what does and does not work to help the debugging of the issue.

Debugging Checklist

The following checklist is a good starting point when debugging an external memory interface. This chapter discusses all of the items in the checklist.

Check	Item
☐	Try a different fit.
☐	Check IP parameters at the operating frequency (t_{MRD}, t_{WTR} for example).
☐	Ensure you have constrained your design with proper timing deration and have closed timing.
☐	Simulate the design. If it fails in simulation, it will fail in hardware.
☐	Analyze timing.
☐	Place and assign R_{UP} and R_{DN} (OCT).
☐	Measure the power distribution network (PDN).
☐	Measure signal integrity.
☐	Measure setup and hold timing.

Check	Item
☐	Measure FPGA voltages.
☐	Vary voltages.
☐	Heat and cool the PCB.
☐	Operate at a lower or higher frequency.
☐	Check board timing and trace Information.
☐	Check LVDS and clock sources, I/O voltages and termination.
☐	Check PLL clock source, specification, and jitter.
☐	Ensure the correct number of PLL phase steps take place during calibration. If the number stated in the IP does not match the number, you may have manually altered the PLL.
☐	Retarget to a smaller interface width or a single bank.

Catagorizing Hardware Issues

The following topic catagorizises issues. Identifying which category or groups of category an issue may be classified within allows you to focus on the cause of the issue.

Signal Integrity Issues

Many design issues, even ones that you find at the protocol layer, can often be traced back to signal integrity issues. Hence, you must check circuit board construction, power systems, command, and data signaling to determine if they meet specifications. If infrequent, random errors exist in the memory subsystem, product reliability suffers. As such, electrical verification is vital. Check the bare circuit board or PCB design file. Circuit board errors can cause poor signal integrity, signal loss, signal timing skew, and trace impedance mismatches. Differential traces with unbalanced lengths or signals that are routed too closely together can cause crosstalk.

Characteristics

Signal integrity issues often appear when the performance of the hardware design is marginal. The design may not always initialize and calibrate correctly, or may exhibit occasional bit errors in user mode. Severe signal integrity issues can result in total failure of an interface at certain data rates, and sporadic component failure because of electrical stress. PCB component variance and signal integrity issues often show up as failures on one PCB, but not on another identical board. Timing issues can have a similar characteristic. Multiple calibration windows or significant differences in the calibration results from one calibration to another can also indicate signal integrity issues.

Evaluating Signal Integrity Issues

Signal integrity issues can only really be evaluated in two ways, direct measurement using suitable test equipment like an oscilloscope and probe, or simulation using a tool like HyperLynx or Allegro PCB SI. Signals should be compared against the respective electrical specification. You should look for overshoot and undershoot, non-monotonicity, eye height and width, and crosstalk.

Skew

Ensure that all clocked signals, commands, addresses, and control signals arrive at the memory inputs at the same time. Trace length variations cause data valid window variations between the signals reducing margin. For example, DDR2-800 at 400 MHz has a data valid window that is smaller than 1,250 ps. Trace length skew or crosstalk can reduce this data valid window further, making it difficult to design a reliably operating memory interface. Ensure that the skew figure previously entered into the Altera IP matches that actually achieved on the PCB, otherwise Quartus II timing analysis of the interface is accurate.

Crosstalk

Crosstalk is another issue that is best evaluated early in the memory design phase. Check the clock-to-data strobes, as these are bidirectional. Measure the crosstalk at both ends of the line. Check the data strobes to clock, as the clocks are unidirectional, these only need checking at the memory end of the line.

Power System

Some memory interfaces tend to draw current in spikes from their power delivery system as SDRAMs are based on capacitive memory cells. Rows are read and refreshed one at a time, which causes dynamic currents that can stress any power distribution network (PDN). The various power rails should be checked either at or as close as possible to the SDRAM pins power pins. Ideally, a real-time oscilloscope set to fast glitch triggering should be used for this activity.

Clock Signals

The clock signal quality is important for any external memory system. Measurements include frequency, digital core design (DCD), high width, low width, amplitude, jitter, rise, and fall times.

Read Data Valid Window and Eye Diagram

The memory generates the read signals. Take measurements at the FPGA end of the line. To ease read diagram capture, modify the example driver to mask writes or modify the PHY to include a signal that you can trigger on when performing reads.

Write Data Valid Window and Eye Diagram

The FPGA generates the write signals. Take measurements at the memory device end of the line. To ease write diagram capture, modify the example driver to mask reads or modify the PHY export a signal that is asserted when performing writes.

OCT and ODT Usage

Modern external memory interface designs typically use OCT for the FPGA end of the line, and ODT for the memory component end of the line. If either the OCT or ODT are incorrectly configured or enabled, signal integrity issues exist. If the design is using OCT, R_{UP} or R_{DN} pins must be placed correctly for the OCT to work. If you do not place these pins, the Quartus II software allocates them automatically with the following warning:

```
Warning: No exact pin location assignment(s) for 2 pins of 110 total pins

Info: Pin termination_blk0~_rup_pad not assigned to an exact location
on the device

Info: Pin termination_blk0~_rdn_pad not assigned to an exact location
on the device
```

If you see these warnings, the R_{UP} and R_{DN} pins may have been allocated to a pin that does not have the required external resistor present on the board. This allocation renders the OCT circuit faulty, resulting in unreliable UniPHY and ALTMEMPHY calibration and or interface behavior. The pins with the required external resistor must be specified in the Quartus II software.

For the FPGA, ensure that follow these actions:

- Specify the R_{UP} and R_{DN} pins in either the projects HDL port list, or in the assignment editor (`termination_blk0~_rup_pad/ termination_blk0~_rdn_pad`).
- Connect the R_{UP} and R_{DN} pins to the correct resistors and pull-up and pull-down voltage in the schematic or PCB.
- Contain the R_{UP} and R_{DN} pins within a bank of the device that is operating at the same VCCIO voltage as the interface that is terminated.
- Check that only the expected number of R_{UP} and R_{DN} pins exists in the project pin-out file. Look for `Info: Created on-chip termination` messages at the fitter stage for any calibration blocks not expected in your design.
- Review the Fitter Pin-Out file for R_{UP} and R_{DN} pins to ensure that they are on the correct pins, and that only the correct number of calibration blocks exists in your design.
- Check in the fitter report that the input, output, and bidirectional signals with calibrated OCT all have the termination control block applicable to the associated R_{UP} and R_{DN} pins.

For the memory components, ensure that you follow these actions:

- Connect the required resistor to the correct pin on each and every component, and ensure that it is pulled to the correct voltage.
- Place the required resistor close to the memory component.
- Correctly configure the IP to enable the desired termination at initialization time.
- Check that the speed grade of memory component supports the selected ODT setting.
- Check that the second source part that may have been fitted to the PCB, supports the same ODT settings as the original

Hardware and Calibration Issues

When you resolve functional, timing, and signal integrity issues, assess the operation of the PHY and its interface calibration.

Hardware and Calibration Issue Characteristics

Hardware and calibration issues have the following definitions:

- Calibration issues result in calibration failing, which typically results in the design asserting the `ctl_cal_fail` signal.
- Hardware issues result in read and write failures, which typically results in the design asserting the pass not fail (`pnf`) signal

Ensure that functional, timing, and signal integrity issues are not the direct cause of your hardware issue, as functional, timing or signal integrity issues are usually the cause of any hardware issue.

Evaluating Hardware and Calibration Issues

Use the following methods to evaluate hardware and calibration issues:

- Evaluate hardware issues using the SignalTap II logic analyzer to monitor the local side read and write interface with the pass or fail or error signals as triggers
- Evaluate calibration issues using the SignalTap II logic analyzer to monitor the various calibration, configuration with the pass or fail or error signals as triggers, but also use the debug toolkit and system consoles when available

For more information about debug toolkits and the type of signals for debugging external memory interfaces, refer to the *ALTMEMPHY External Memory Interface Debug Toolkit* and *UniPHY External Memory Interface Debug Toolkit* chapters in volume 3 of the *External Memory Interface Handbook*.

Consider adding core noise to your design to aggravate margin timing and signal integrity issues. Steadily increase the stress on the interface in the following order:

1. Increase the interface utilization by modifying the example driver to focus on the types of transactions that exhibit the issue.
2. Increase the SNN or aggressiveness of the data pattern by modifying the example driver to output in synchronization PRBS data patterns, or hammer patterns.
3. Increase the stress on the PDN by adding more and more core noise to your system. Try sweeping the fundamental frequency of the core noise to help identify resonances in your power system.

Steadily increasing the stress on the external memory interface is an ideal way to assess and understand the cause of any previously intermittent failures that you may observe in your system. Using the SignalTap II probe tool can provide insights into the source or cause of operational failure in the system.

Additionally, steadily increasing stress on the external memory interface allows you to assess and understand the impact that such factors have on the amount of timing margin and resynchronization window. Take measurements with and without the additional stress factor to allow evaluation of the overall effect.

Write Timing Margin

Determine the write timing margin by phase sweeping the write clock from the PLL. Use sources and probes to dynamically control the PLL phase offset control, to increase and decrease the write clock phase adjustment so that the write window size may be ascertained.

Remember that when sweeping PLL clock phases, the following two factors may cause operational failure:

- The available write margin.
- The PLL phase in a multi-clock system.

The following code achieves this adjustment. You should use sources and probes to modify the respective output of the PLL. Ensure that the example driver is writing and reading from the memory while observing the `pnf_per_byte` signals to see when write failures occur:

```
/////////////////
wire [7:0] Probe_sig;
wire [5:0] Source_sig;
PhaseCount PhaseCounter (
 .resetn (1'b1),
 .clock (pll_ref_clk),
 .step (Source_sig[5]),
 .updown (Source_sig[4]),
 .offset (Probe_sig)
 );
CheckoutPandS freq_PandS (
 .probe (Probe_sig),
 .source (Source_sig)
 );
ddr2_dimm_phy_alt_mem_phy_pll_siii pll (
 .inclk0 (pll_ref_clk),
 .areset (pll_reset),
 .c0 (phy_clk_1x), // hR
.c1 (mem_clk_2x), // FR
.c2 (aux_clk), // FR
 .c3 (write_clk_2x), // FR
 .c4 (resync_clk_2x), // FR
 .c5 (measure_clk_1x), // hR
 .c6 (ac_clk_1x), // hR
 .phasecounterselect (Source_sig[3:0]),
 .phasestep (Source_sig[5]),
 .phaseupdown (Source_sig[4]),
 .scanclk (scan_clk),
```

```
.locked (pll_locked_src),
.phasedone (pll_phase_done)
);
```

Read Timing Margin

Similarly, assess the read timing margin by using sources and probes to manually control the DLL phase offset feature. Open the autogenerated DLL using ALT_DLL and add the additionally required offset control ports. This action allows control and observation of the following signals:

```
dll_delayctrlout[5:0], // Phase output control from DLL to DQS pins (Gray Coded)
dll_offset_ctrl_a_addnsub, // Input add or subtract the phase offset value
dll_offset_ctrl_a_offset[5:0], // User Input controlled DLL offset value (Gray Coded)
dll_aload, // User Input DLL load command
dll_dqsupdate, // DLL Output update required signal.
```

In examples where the applied offset applied results in the maximum or minimum `dll_delayctrlout[5:0]` setting without reaching the end of the read capture window, regenerate the DLL in the next available phase setting, so that the full capture window is assessed.

Modify the example driver to constantly perform reads (mask writes). Observe the `pnf_per_byte` signals while the DLL capture phase is manually modified to see when failures begin, which indicates the edge of the window.

A resynchronization timing failure can indicate failure at that capture phase, and not a capture failure. You should recalibrate the PHY with the calculated phase offset to ensure that you are using the true read-capture margin.

Address and Command Timing Margin

You set the address and command clock phase directly in the IP. Assuming you enter the correct board trace model information into the Quartus II software, the timing analysis should be correct. However, if you want to evaluate the address and command timing margin, use the same process as in "Write Timing Margin", only phase step the address and command PLL output (c6 `ac_clk_1x`). You can achieve this effect using the debug toolkit or system console.

Refer to the *ALTMEMPHY External Memory Interface Debug Toolkit* and *UniPHY External Memory Interface Debug Toolkit* chapters in volume 3 of the *External Memory Interface Handbook.*

Resynchronization Timing Margin

Observe the size and margins, available for resynchronization using the debug toolkit or system console.

Refer to the *ALTMEMPHY External Memory Interface Debug Toolkit* and *UniPHY External Memory Interface Debug Toolkit* chapters in volume 3 of the *External Memory Interface Handbook.*

Additionally for PHY configurations that use a dedicated PLL clock phase (as opposed to a resynchronization FIFO buffer), use the same process as described in "Write Timing Margin", to dynamically sweep resynchronization margin (c4 `resynch_clk_2x`).

Postamble Timing Issues and Margin

The postamble timing is set by the PHY during calibration. You can diagnose postamble issues by viewing the `pnf_per_byte` signal from the example driver. Postamble timing issues mean only read data is corrupted during the last beat of any read request.

Intermittent Issues

Intermittent issues are typically the hardest type of issue to debug—they appear randomly and are hard to replicate.

Intermittent Issue Evaluation

Errors that occur during run-time indicate a data related issue, which you can identify by the following actions:

- Add the SignalTap II logic analyzer and trigger on the post-trigger `pnf`
- Use a stress pattern of data or transactions, to increase the probability of the issue
- Heat up or cool down the system
- Run the system at a slightly faster frequency

If adding the SignalTap II logic analyzer or modifying the project causes the issue to go away, the issue is likely to be placement or timing related.

Errors that occur at start-up indicate that the issue is related to calibration, which you can identify by the following actions:

- Modify the design to continually calibrate and reset in a loop until the error is observed
- Where possible, evaluate the calibration margin either from the debug toolkit or system console.

  Refer to the *ALTMEMPHY External Memory Interface Debug Toolkit* and *UniPHY External Memory Interface Debug Toolkit* chapters in volume 3 of the *External Memory Interface Handbook.*

- Capture the calibration error stage or error code, and use this information with whatever specifically occurs at that stage of calibration to assist with your debug of the issue.

Debug Toolkit

The debug toolkit is an interface that runs on your PC and enables you to debug your external memory interface design on the circuit board, retrieve calibration status, and perform margining activities. Debug toolkit uses a JTAG connection to a Windows PC.

Altera provides the following types of debug toolkits:

- ALTMEMPHY Debug Toolkit
- UniPHY EMIF Debug Toolkit

ALTMEMPHY Debug Toolkit Overview and Usage Flow

The ALTMEMPHY Debug Toolkit supports the following Altera AFI-based IP:

- ALTMEMPHY megafunction
- DDR2 and DDR3 SDRAM High-Performance Controller and High Performance Controller II

The debug toolkit does not support the QDR II and II+ SRAM, RLDRAM II with UniPHY controllers.

The ALTMEMPHY Debug Toolkit lists and indicates whether calibration stages are successful, states error specific to calibration failure and provides possible causes. However, the debug toolkit does not fix a failing design. You can run the debug toolkit and the SignalTap II logic analyzer at the same time.

The ALTMEMPHY Debug Toolkit usage flow involves the following steps:

1. Before using the debug toolkit, modify the design example top-level file by regenerating the IP with the JTAG Avalon-MM port enabled.
2. Add additional debug and sequencer signals to indicate the location of calibration failure, resyncronization margin, read and write latency, and PLL status.
3. Recompile the design.
4. Connect your pc's download cable (for example, ByteBlaster™ II download cable) to the JTAG port on the development board.
5. Program the device with the debug enabled in your design using the SignalTap II logic analyzer.
6. Run analysis and interpret calibration results using the ALTMEMPHY Debug Toolkit with the SignalTap II logic analyzer.

For more information about the ALTMEMPHY debug toolkit and calibration stages, refer to the *ALTMEMPHY External Memory Interface Debug Toolkit* chapter in volume 3 of the *External Memory Interface Handbook*.

UniPHY EMIF Debug Toolkit Overview and Usage Flow

The UniPHY EMIF Debug Toolkit is a Tcl-based interface and consists of the following parts:

- DDR2 and DDR3 SDRAM Controllers with UniPHY
- Avalon Memory-Mapped (Avalon-MM) slave interface
- JTAG Avalon master

The EMIF toolkit allows you to display information about your external memory interface and generate calibration and margining reports. The toolkit can aid in diagnosing the type of failure that may be occurring in your external memory interface, and help identify areas of reduced margin that might be potential failure points.

The UniPHY Debug Toolkit usage flow involves the following steps:

1. (Optional) Generate your IP core with the CSR port enabled and with the CSR communication interface type properly set.
2. Recompile the design.
3. Connect your pc's download cable (for example, ByteBlaster II download cable) to the JTAG port on the development board.
4. Program the device.
5. Specify project settings using the UniPHY EMIF Debug Toolkit.
6. Generate calibration report and interpret calibration results using the UniPHY EMIF Debug Toolkit.

For more information about the UniPHY EMIF debug toolkit and calibration stages, refer to the *UniPHY External Memory Interface Debug Toolkit* chapter in volume 3 of the *External Memory Interface Handbook*.

Document Revision History

Table 12–1 lists the revision history for this document.

Table 12–1. Document Revision History (Part 1 of 2)

Date	Version	Changes
November 2012	4.2	Changed chapter number from 11 to 12.
June 2012	4.1	Added Feedback icon.
November 2011	4.0	Added Debug Toolkit section.
June 2011	3.0	Removed leveling information from *ALTMEMPHY Calibration Stages* and *UniPHY Calibration Stages* chapter.
December 2010	2.1	■ Added new chapter: *UniPHY Calibration Stages*. ■ Added new chapter: *DDR2 and DDR3 SDRAM Controllers with UniPHY EMIF Toolkit*.
July 2010	2.0	Updated for 10.0 release.
January 2010	1.2	Corrected minor typos.

Table 12–1. Document Revision History (Part 2 of 2)

Date	Version	Changes
December 2009	1.1	Added *Debug Toolkit for DDR2 and DDR3 SDRAM High-Performance Controllers* chapter and *ALTMEMPHY Calibration Stages* chapter.
November 2009	1.0	First published.

Section II. Miscellaneous Guidelines

This section provides information about HardCopy migration for UniPHY-based designs, ways to increase the efficiency of the controller and the PHY, and describes the power estimation methods.

This section includes the following chapters:

- Chapter 13, HardCopy Design Migration Guidelines
- Chapter 14, Optimizing the Controller
- Chapter 15, PHY Considerations
- Chapter 16, Power Estimation Methods for External Memory Interface Designs

For information about the revision history for chapters in this section, refer to "Document Revision History" in each individual chapter.

13. HardCopy Design Migration Guidelines

EMI_DG_012-2.2

This chapter discusses HardCopy® migration guidelines for UniPHY-based designs. If you want to migrate your ALTMEMPHY-based designs to HardCopy, Altera recommends that you first upgrade your design to UniPHY.

For information about upgrading an ALTMEMPHY-based design to UniPHY, refer to the *Upgrading to UniPHY-based Controllers from ALTMEMPHY-based Controllers* chapter in volume 3 of the *External Memory Interface Handbook*.

HardCopy Migration Design Guidelines

If you intend to target your design to a HardCopy device, you should select both your prototyping FPGA device and target HardCopy device at the start of your project, to avoid any late difficulties that might occur due to differences in the UniPHY IP between FPGA and HardCopy implementations.

You must migrate your design from an FPGA to a HardCopy companion device; you cannot target HardCopy directly.

Ensure you use the following design guidelines when migrating your design:

- On the **PHY Settings** page of the parameter editor, turn on **HardCopy Compatibility Mode**, and then specify whether the **Reconfigurable PLL Location** is **Top_Bottom** or **Left_Right**.

  Altera recommends that you set the **Reconfigurable PLL Location** to the same side as your memory interface.

  When turned on, the **HardCopy Compatibility Mode** option enables a ROM loader and run-time reconfiguration for all phase-locked loops (PLLs) and delay-locked loops (DLLs) instantiated in memory interfaces. In this mode, all the necessary PLL and DLL reconfiguration and ROM loader signals are brought to the top level of the design.

- Enable run-time reconfiguration mode for all PLLs and DLLs instantiated in interfaces that are configured in PLL and DLL slaves.

  For information about PLL megafunctions, refer to the *Phase-Locked Loop (ALTPLL) Megafunction User Guide* and the *Phase-Locked Loops Reconfiguration (ALTPLL_RECONFIG) Megafunctions User Guide*. For information about DLL megafunctions, refer to the *ALTDLL and ALTDQ_DQS Megafunctions User Guide*.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

- Ensure that you place all memory interface pins close together. If, address pins are located far away from data pins, for example, closing timing might be difficult.
- For DDR2 and DDR3 (and RLDRAM II when using the Nios® II -based sequencer) UniPHY-based designs, ensure that you have external nonvolatile ROM or flash memory on your circuit board for storing the Nios II instruction code. (QDR II and QDR II+ SRAM with UniPHY designs support only the RTL-based sequencer in HardCopy migration.) The UniPHY IP instantiates a ROM loader to load Nios II instruction code from external ROM.

  For ROM loader connection guidelines, refer to "ROM Loader for Designs Using Nios II Sequencer" on page 13–3.
- In the wraparound interface design, open the *<variation_name>*_**p0_new_io_pads.v** file in an editor and locate the following line:

```
.dll_offsetdelay_in((i < 0) ?
hc_dll_config_dll_offset_ctrl_offsetctrlout :
hc_dll_config_dll_offset_ctrl_offsetctrlout),
```

  In the preceding line, first change the second `hc_dll_config_dll_offset_ctrl_offsetctrlout` to `hc_dll_config_dll_offset_ctrl_b_offsetctrlout`, and then change the numeral **0** to the number of DQS groups located in the top or bottom I/O edge. For example, changing 0 to 3 would mean that DQS groups 0 to 2 are connected to the output of the first DLL offset control block. DQS group 3 and above are connected to the output of the second DLL offset control block.

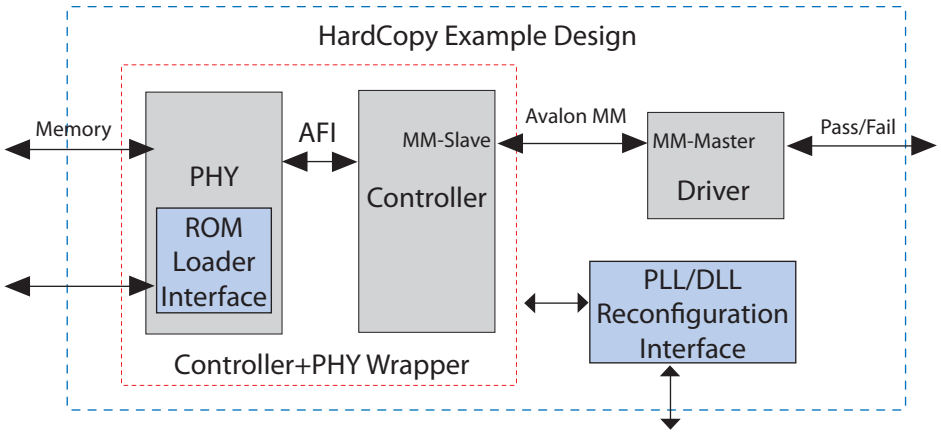
You can use the example top-level project that is generated when you turn on **HardCopy Migration** as a guide to help you connect the necessary signals in your design.

Differences in UniPHY IP Generated with HardCopy Migration Support

When you generate a UniPHY memory interface for HardCopy device support, certain features in the IP are enabled that do not exist when you generate the IP core for only the FPGA. This section discusses those additional enabled features.

Figure 13–1 shows the additional blocks enabled when **HardCopy Compatibility Mode** is turned on.

Figure 13–1. HardCopy UniPHY Example Design



ROM Loader for Designs Using Nios II Sequencer

An additional ROM loader is instantiated in the design for UniPHY designs that use the Nios II sequencer. The Nios II sequencer instruction code resides in RAM on either the HardCopy or FPGA device.

When you target only an FPGA device, the RAM is initialized when the device is programmed; however, HardCopy devices are not programmable and therefore the RAM cannot be initialized in this fashion. Instead, the Nios II sequencer instruction code must be stored in an external, non-volatile memory and must be loaded to the Nios II sequencer RAM through a ROM loader. You must create a subsystem to load the Nios II sequencer code from the external nonvolatile memory to the ROM loader.

The instruction code varies according to memory protocols and memory parameterization in UniPHY. You can share the same sequencer instruction code content for multiple interface designs if the memory protocol and settings are the same for each interface. You can also store different Nios II code in the same external, nonvolatile memory and use a single subsystem to load the Nios II codes to the corresponding Nios II sequencer RAMs.

For more information about the ROM loader, refer to the *RAM Initializer (ALTMEM_INIT) Megafunction User Guide*.

Table 13–1 lists the ports exposed at the top level of the PHY+Controller wrapper to expose the ROM loader utilized by the Nios II-based sequencer within the RLDRAM II, DDR2, or DDR3 PHY.

Table 13–1. Top-level Ports that Connect to External ROM for Loading Nios II Code Memory

Port Name	Direction	Size	Description
hc_rom_config_clock	Input	1 bit	Write clock for the ROM loader. This clock is the write clock for the Nios II code memory.
hc_rom_config_datain	Input	32 bits	Data input from external ROM.
hc_rom_config_rom_data_ready	Input	1 bit	Asserts to the code memory loader that the word of memory is ready to be loaded.
hc_rom_config_init	Input	1 bit	Signals that the Nios II code memory is being loaded from the external ROM.
hc_rom_config_init_busy	Output	1 bit	Remains asserted throughout initialization and becomes inactive when initialization is complete. soft_reset_n can be issued after hc_rom_config_init_busy is deasserted.
hc_rom_config_rom_rden	Output	1 bit	Read-enable signal that connects to the external ROM.
hc_rom_config_rom_address	Output	12 bits	ROM address that connects to the external ROM.

You can load the Nios II instruction code in several ways. You can connect the ROM loader directly to the dedicated external, nonvolatile memory; alternatively, you may reuse the FPGA configuration interface with flash memory to load the Nios II instruction code using the ROM loader. The configuration flash memory is used to configure the FPGA design, but device configuration is not required in HardCopy designs, conserving resources and board space. You can reuse existing configuration pins and flash memory for interfacing with the ROM loader without any extra I/O pins or flash memory.

Three FPGA configuration schemes are available with flash memory: passive serial (PS) configuration, active serial (AS) configuration, and fast passive parallel (FPP) configuration. Only active serial (AS) and fast passive parallel (FPP) configuration interfaces are suitable for interfacing with the ROM loader when the device is in user mode.

For more information about FPGA configuration schemes, refer to the *Configuration Handbook*.

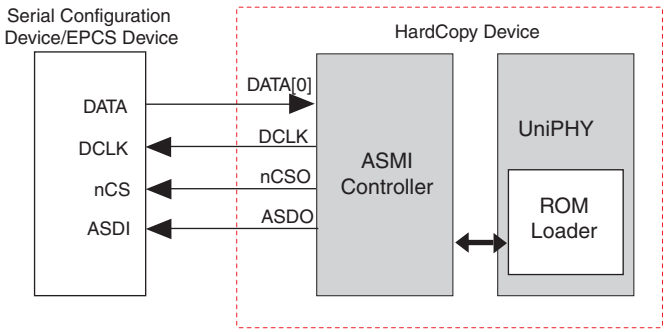
Passive Serial (PS) Configuration Scheme

In the passive serial configuration scheme, CONF_DONE, nSTATUS, nCE, nCONFIG, DATA[0], and DCLK are used for FPGA configuration. Only the DATA[0] signal is a dual-purpose configuration pin which is used as a normal I/O pin in user mode. Thus, only a single DATA[0] pin is available in the HardCopy device to use as an I/O pin. Interfacing with the ROM loader requires more than one I/O pin; therefore this configuration scheme cannot be used for interfacing with the ROM loader. Altera recommends using other configuration schemes in UniPHY-based designs that you intend to migrate to HardCopy devices.

Active Serial (AS) Configuration Scheme

The active serial configuration scheme uses four configuration pins (DATA[0], DCLK, ASDO, and nCSO) to configure the FPGA. You can directly access the content of the flash memory through these four configuration pins in FPGA user mode, or in the HardCopy device using the active serial memory interface (ASMI) controller. You must add the ASMI controller to your HardCopy design to interface with the ROM loader. Figure 13–2 shows an example connection between a ROM loader with ASMI controller and EPCS flash memory.

Figure 13–2. ROM Loader Connection in Active Serial Configuration Scheme



For more information about the ASMI controller, refer to the *Active Serial Memory Interface (ALTASMI_PARALLEL) Megafunction User Guide*.

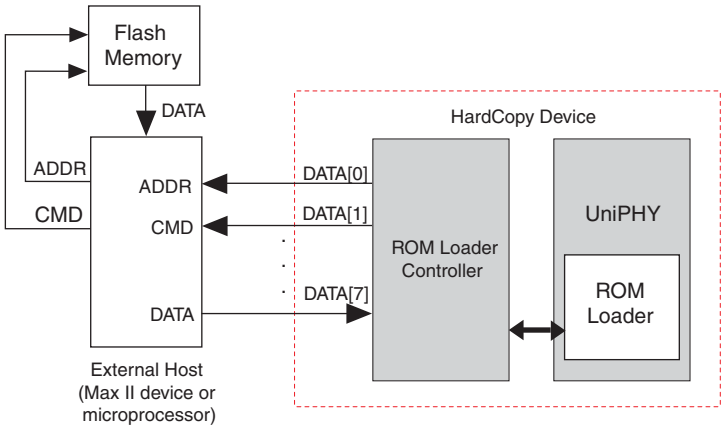
Fast Passive Parallel (FPP) Configuration Scheme

In the fast passive parallel configuration scheme, only the DATA[0...7] configuration pins can be used as normal I/Os in a HardCopy device; eight DATA pins are available with this scheme.

The data pins can be used as clock, data, address, and command pins to interface with the external host for loading the flash memory content to Nios II through the ROM loader. You must ensure the flash memory data pins are connected to the external host or controller before connecting to the FPGA during FPGA configuration, in order to reuse these data pins.

FPGA configuration uses an external host which you must configure to interface with the ROM loader and to read the content of the flash memory. Due to limited FPP interface I/O count, you must create two controllers to serialize and deserialize the address (12 bits) and data (23 bits) signals of the ROM loader. One controller resides in the HardCopy device and one resides in the external host device. Figure 13–3 shows an example of a ROM loader configured in the FPP configuration scheme.

Figure 13–3. ROM Loader Connection in Fast Passive Parallel Configuration Scheme



PLL/DLL Run-time Reconfiguration

The PLLs and DLLs in the HardCopy design have run-time reconfiguration enabled—provided that they are not in PLL/DLL slave mode.

When the PLLs and DLLs are generated with reconfiguration enabled, extra signals must be connected and driven by user logic. In the example design generated during IP core generation, the PLL/DLL reconfiguration signals are brought to the top level and connected to constants.

For information about PLL megafunctions and reconfiguration, refer to the *Phase-Locked Loop (ALTPLL) Megafunction User Guide* and the *Phase-Locked Loops Reconfiguration (ALTPLL_RECONFIG) Megafunctions User Guide*.

Table 13–2 lists the DLL reconfiguration ports exposed at the top level of the Controller and PHY wrapper.

Table 13–2. DLL Reconfiguration Ports Exposed at Top-Level of Controller+PHY Wrapper (Part 1 of 2)

Port Name	Direction	Size	Description
`hc_dll_config_dll_offset_ctrl_addnsub` (1)	Input	1 bit	Addition/subtraction control port for the DLL. This port controls whether the delay-offset setting on `hc_dll_config_dll_offset_ctrl_offset` is added or subtracted.
`hc_dll_config_dll_offset_ctrl_offset` (1)	Input	6 bits	Offset input setting for the PLL. This is a Gray-coded offset that is added or subtracted from the current value of the DLL's delay chain.
`hc_dll_config_dll_offset_ctrl_offsetctrlout`	Output (2)	6 bits	The registered and gray-coded value of the current delay-offset setting for the DLL offset control block that controls DQS pins on the top or bottom I/O edge.

Table 13–2. DLL Reconfiguration Ports Exposed at Top-Level of Controller+PHY Wrapper (Part 2 of 2)

Port Name	Direction	Size	Description
hc_dll_config_dll_offset_ctrl_b_offsetctrlout	Output (2)	6 bits	The registered and gray-coded value of the current delay offset setting for the DLL offset control block that controls DQS pins on left or right I/O edge.

Note:
(1) Available only in DLL nonsharing mode and DLL master sharing mode.
(2) Functions as an output in DLL nonsharing mode and DLL master sharing mode, and as an input in DLL slave mode.

Table 13–3 lists the PLL reconfiguration ports exposed at the top level of the Controller and PHY wrapper.

Table 13–3. PLL Reconfiguration Ports Exposed at the Top-Level of Controller+PHY Wrapper (1)

Port Name	Direction	Size	Description
hc_pll_config_configupdate	Input	1 bit	Control signal to enable PLL reconfiguration. (Applies to RLDRAMII and QDRII only; the phase reconfiguration feature for DDR2/3 is included in the CSR port.)
hc_pll_config_phasecounterselect	Input	4 bits	Specifies the counter select for dynamic phase adjustment. (Applies to RLDRAMII and QDR II only.)
hc_pll_config_phasestep	Input	1 bit	Specifies the phase step for dynamic phase shifting. (Applies to RLDRAMII and QDR II only.)
hc_pll_config_phaseupdown	Input	1 bit	Specifies if the phase adjustment should be up or down. (Applies to RLDRAMII and QDR II only.)
hc_pll_config_scanclk	Input	1 bit	PLL reconfiguration scan chain clock.
hc_pll_config_scanclkena	Input	1 bit	Clock enable port of the hc_pll_config_scanclk clock.
hc_pll_config_scandata	Input	1 bit	Serial input data for the PLL reconfiguration scan chain.
hc_pll_config_phasedone	Output	1 bit	When asserted, this signal indicates to core logic that phase adjustment is completed and that the PLL is ready to act on a possible second adjustment pulse.
hc_pll_config_scandataout	Output	1 bit	The data output of the serial scan chain.
hc_pll_config_scandone	Output	1 bit	This signal is asserted when the scan chain write operation is in progress and is deasserted when the write operation is complete.

Note:
(1) Inputs and outputs are available only in PLL nonsharing mode and PLL master sharing mode. No inputs or outputs are available in PLL slave mode.

To facilitate placement and timing closure and to help compensate for PLLs adjacent to I/Os and vertical I/O overhang issues that can occur when targeting HardCopy III and HardCopy IV devices, an additional pipeline stage is added to the write path in the RTL when you turn on **HardCopy Compatibility**. The additional pipeline stage is added in all cases, except when CAS write latency equals 2 (for DDR3) or CAS latency equals 3 (for DDR2), where the additional pipeline stage is not required to meet timing requirements. The additional pipeline stage does not affect the overall latency of the controller, because the controller's internal latency is reduced by 1 to compensate for the extra pipeline stage.

In DDR2 and DDR3 designs, at a certain frequency the DLL length changes when you generate the IP with **HardCopy Compatibility** enabled; this allows the DLL to work in both FPGA and HardCopy devices at the requested frequency. In addition, because the memory clock uses the global clock network, the write clock changes from a regional clock network to a global clock network, for reduced skew between the write clock and memory clock, resulting in improved leveling timing.

For information about HardCopy issues such as vertical I/O overhang, PLLs adjacent to I/Os, and timing closure, refer to the *I/O Features for HardCopy III Devices* chapter in volume 1 of the *HardCopy III Device Handbook* and *I/O Features for HardCopy IV Devices* chapter in volume 1 of the *HardCopy IV Device Handbook*.

Document Revision History

Table 13–4 lists the revision history for this document.

Table 13–4. Document Revision History

Date	Version	Changes
November 2012	2.2	■ Changed chapter number from 12 to 13.
June 2012	2.1	■ Added Feedback icon.
November 2011	2.0	■ Reorganized HardCopy design migration information into an individual chapter. ■ Updated the HardCopy Migration Design Guidelines section.
June 2011	1.0	Initial release.

EMI_DG_013-3.1

Understanding how to increase the efficiency and bandwidth of the memory controller is important when you design any external memory interface. This section discusses factors that affect controller efficiency and ways to increase the efficiency of the controller.

Controller Efficiency

Controller efficiency varies depending on data transaction. The best way to determine the efficiency of the controller is to simulate the memory controller for your specific design.

You express controller efficiency as:

Efficiency = number of active cycles of data transfer/total number of cycles

The total number of cycles includes the number of cycles required to issue commands or other requests.

You calculate the number of active cycles of data transfer in terms of local clock cycles. For example, if the number of active cycles of data transfer is 2 memory clock cycles, you convert that to the local clock cycle which is 1.

The following cases are based on a DDR2 SDRAM high-performance controller design targeting a Stratix® IV device that has a CAS latency of 3, and burst length of 4 on the memory side (2 cycles of data transfer), with accessed bank and row in the memory device already open. The Stratix IV device has a command latency of 9 cycles in half-rate mode. The `local_ready` signal is high.

- Case 1: The controller performs individual reads.

  Efficiency = 1/(1 + CAS + command latency) = 1/(1+1.5+9) = 1/11.5 = 8.6%

- Case 2: The controller performs 4 back to back reads.

  In this case, the number of data transfer active cycles is 8. The CAS latency is only counted once because the data coming back after the first read is continuous. Only the CAS latency for the first read has an impact on efficiency. The command latency is also counted once because the back to back read commands use the same bank and row.

  Efficiency = 4/(4 + CAS + command latency) = 4/(4+1.5+9) = 1/14.5 = 27.5%

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Factors Affecting Efficiency

The two main factors that affect controller efficiency are the interface standard specified by the memory vendor, and the way you transfer data.

The following sections discuss these two factors in detail.

Interface Standard

Complying with certain interface standard specifications affects controller efficiency. When interfacing the memory with the DDR2 or DDR3 SDRAM controllers, you must follow certain timing specifications and perform the following bank management operations:

- Activate

  Before you issue any read (RD) or write (WR) commands to a bank within a DDR2 SDRAM device, you must open a row in that bank using the activate (ACT) command. After you open a row, you can issue a read or write command to that row based on the t_{RCD} specification. Reading or writing to a closed row has negative impact on the efficiency as the controller has to first activate that row and then wait until t_{RCD} time to perform a read or write.

- Precharge

  To open a different row in the same bank, you must issue a precharge (PCH) command. The precharge command deactivates the open row in a particular bank or the open row in all banks. Switching a row has a negative impact on the efficiency as you must first precharge the open row, then activate the next row and wait t_{RCD} time to perform any read or write operation to the row.

- Device CAS latency

  The higher the CAS latency, the less efficient an individual access. The memory device has its own read latency, which is about 12 ns to 20 ns regardless of the actual frequency of the operation. The higher the operating frequency, the longer the CAS latency is in number of cycles.

- Refresh

  A refresh, in terms of cycles, consists of the precharge command and the waiting period for the auto refresh. Based on the memory datasheet, these components require the following values:

  - t_{RP} = 12 ns, 3 clock cycles for a 200-MHz operation (5 ns period for 200 MHz)
  - t_{RFC} = 75 ns, 15 clock cycles for a 200-MHz operation.

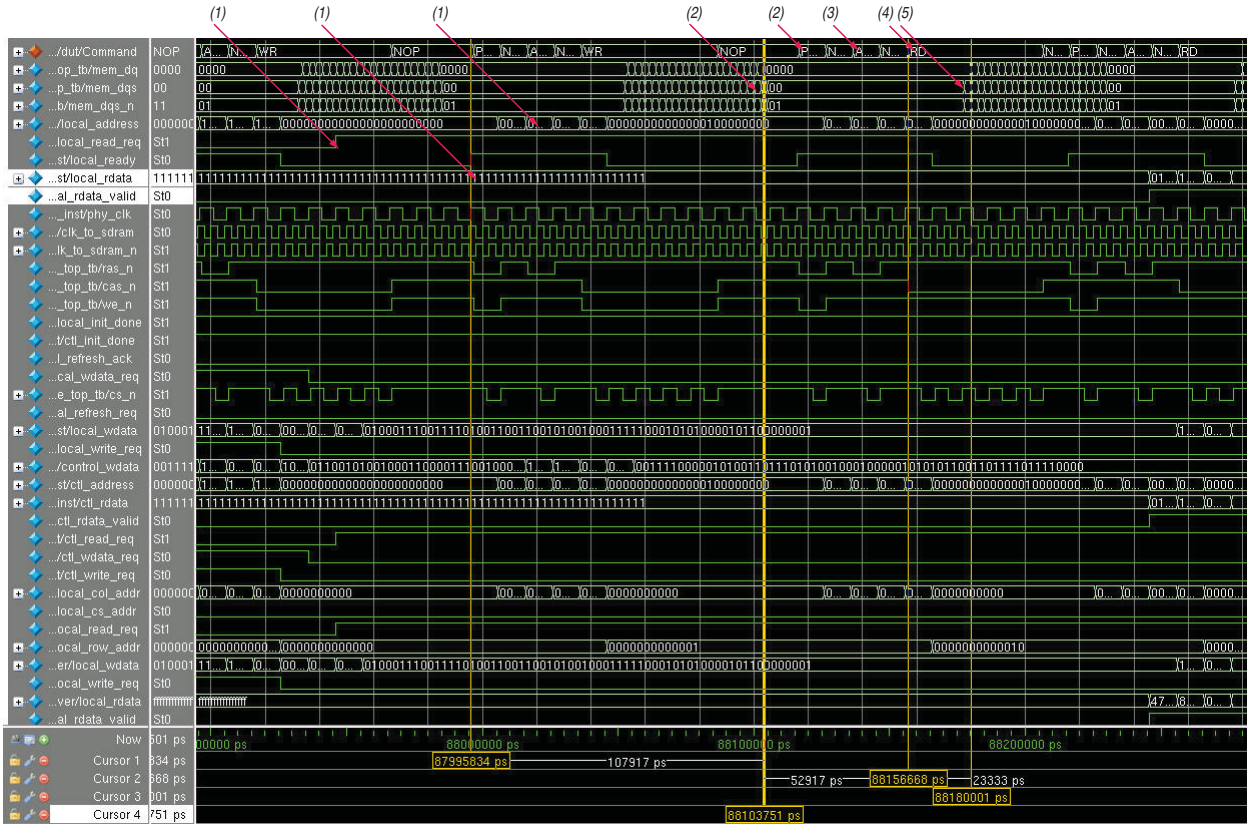
  Based on this calculation, a refresh pauses read or write operations for 18 clock cycles. So, at 200 MHz, you lose 1.15% (18 x 5 ns/7.8 us) of the total efficiency.

Bank Management Efficiency

Figure 14–1 and Figure 14–2 show some examples of how the bank management operations affect controller efficiency. Figure 14–1 shows a read operation in which you have to change a row in a bank. This figure shows how CAS latency and precharge and activate commands affect efficiency.

Figure 14–1 illustrates a read-after-write operation. The controller changes the row address after the write-to-read from a different row.

Figure 14–1. Read Operation—Changing A Row in A Bank



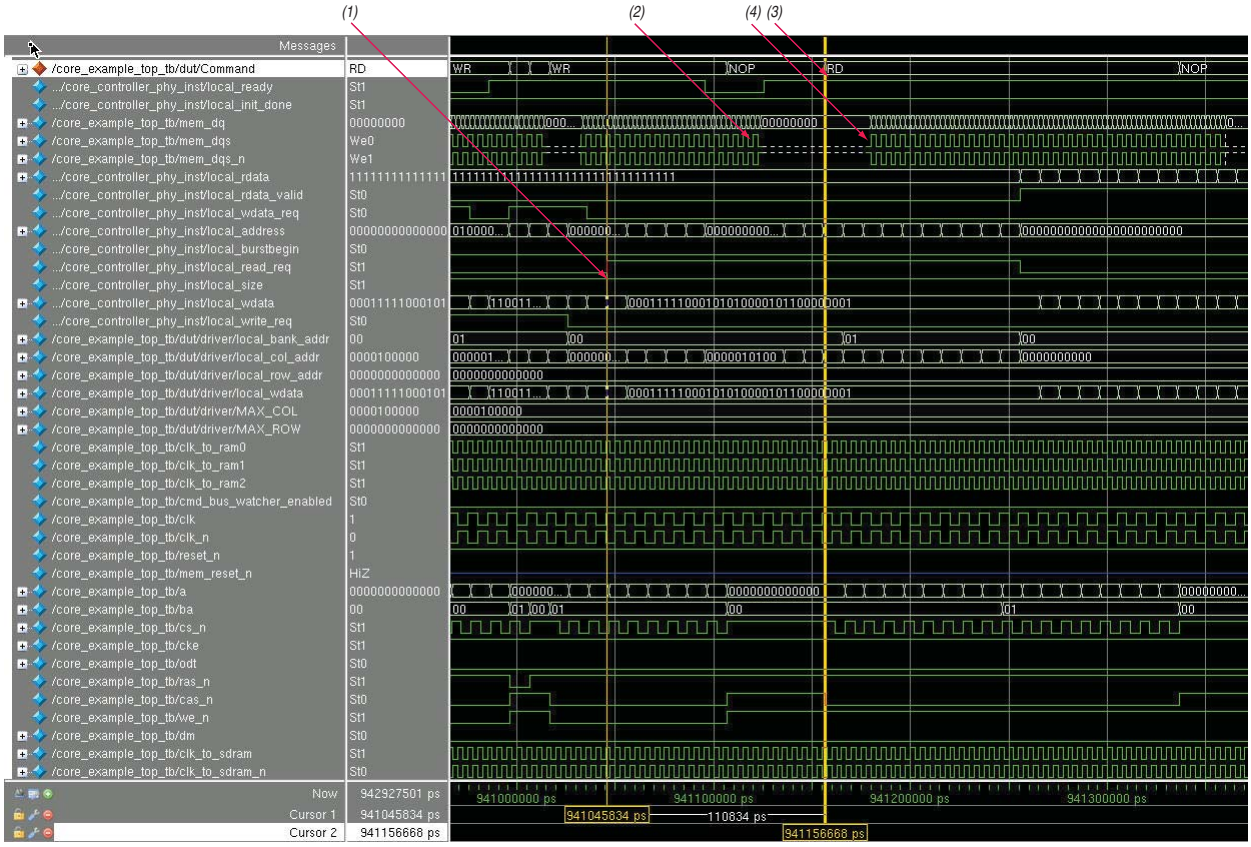
The following sequence of events describes Figure 14–1:

1. The `local_read_req` signal goes high, and when the `local_ready` signal goes high, the controller accepts the read request along with the address.
2. After the memory receives the last write data, the row changes for read. Now you require a precharge command to close the row opened for write. The controller waits for t_{WR} time (3 memory clock cycles) to give the precharge command after the memory receives the last write data.
3. After the controller issues the precharge command, it must wait for t_{RP} time to issue an activate command to open a row.
4. After the controller gives the activate command to activate the row, it needs to wait t_{RCD} time to issue a read command.
5. After the memory receives the read command, it takes the memory some time to provide the data on the pin. This time is known as CAS latency, which is 3 memory clock cycles in this case.

For this particular case, you need approximately 17 local clock cycles to issue a read command to the memory. Because the row in the bank changes, the read operation takes a longer time, as the controller has to issue the precharge and activate commands first. You do not have to take into account t_{WTR} for this case because the precharge and activate operations already exceeded t_{WTR} time.

Figure 14–2 shows the case where you use the same the row and bank address when the controller switches from write to read. In this case, the read command latency is reduced.

Figure 14–2. Changing From Write to Read—Same Row and Bank Address



The following sequence of events describes Figure 14–2:

1. The `local_read_req` signal goes high and the `local_ready` signal is high already. The controller accepts the read request along with the address.
2. When switching from write to read, the controller has to wait t_{WTR} time before it gives a read command to the memory.
3. The SDRAM device receives the read command.
4. After the SDRAM device receives the read command, it takes some time to give the data on the pin. This time is called CAS latency, which is 3 memory clock cycles in this case.

For the case illustrated in Figure 14–2, you need approximately 11 local clock cycles to issue a read command to the memory. Because the row in the bank remains the same, the controller does not have to issue the precharge and activate commands, which speeds up the read operation and in turn results in a better efficiency compared to the case in Figure 14–1.

Similarly, if you do not switch between read and write often, the efficiency of your controller improves significantly.

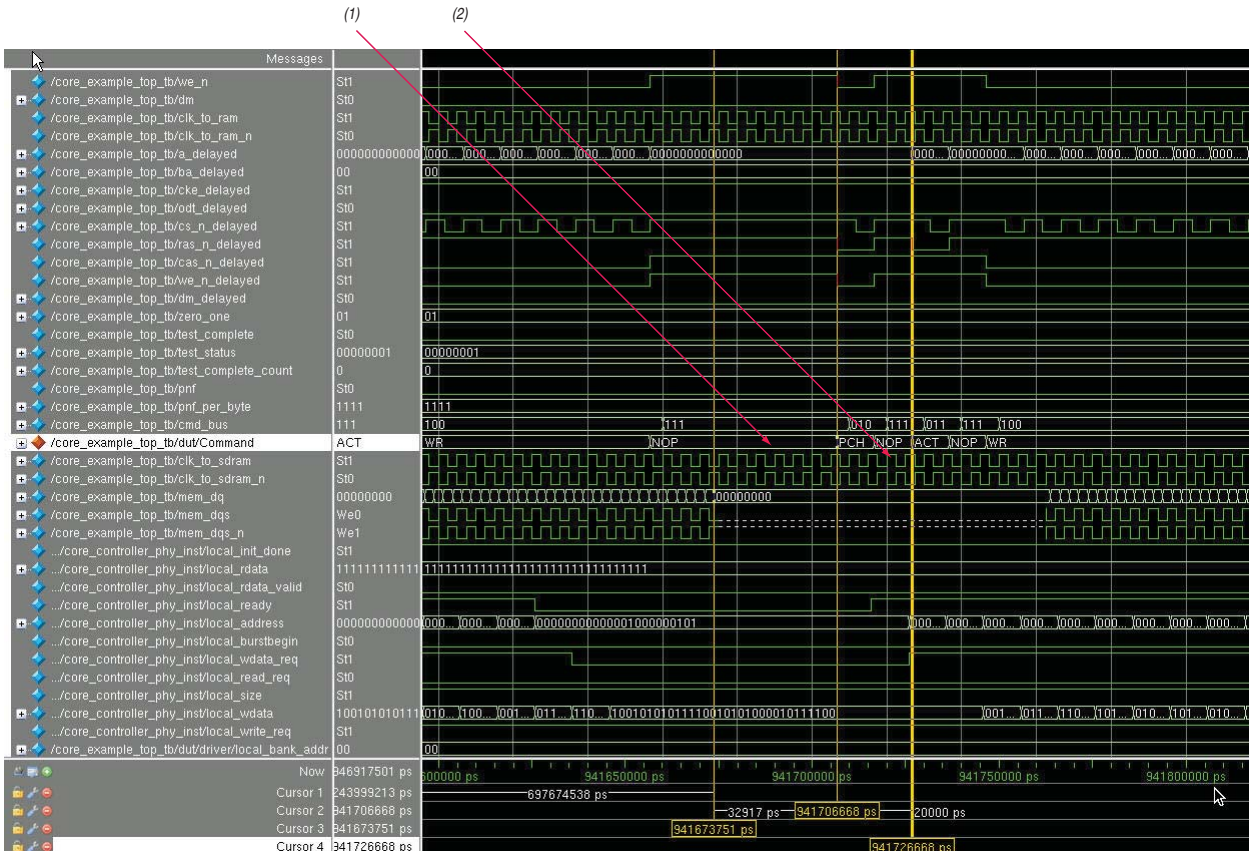
Data Transfer

The following methods of data transfer reduce the efficiency of your controller:

- Performing individual read or write accesses is less efficient.
- Switching between read and write operation has a negative impact on the efficiency of the controller.
- Performing read or write operations from different rows within a bank or in a different bank—if the bank and a row you are accessing is not already open—also affects the efficiency of your controller.

Figure 14–3 shows an example of changing the row in the same bank.

Figure 14–3. Changing Row in the Same Bank



The following sequence of events describes Figure 14–3:

1. You have to wait t_{WR} time before giving the precharge command
2. You then wait t_{RP} time to give the activate command.

Ways to Improve Efficiency

To improve the efficiency of your controller, you can use the following methods:

- DDR2 SDRAM Controller
- Auto-Precharge Commands
- Additive Latency
- Bank Interleaving
- Command Queue Look-Ahead Depth
- Additive Latency and Bank Interleaving
- User-Controlled Refresh
- Frequency of Operation
- Burst Length
- Series of Reads or Writes

The following sections discuss these methods in detail.

DDR2 SDRAM Controller

The DDR2 SDRAM controller maintains up to eight open banks; one row in each bank is open at a time. Maintaining more banks at one time helps avoid bank management commands. Ensure that you do not change a row in a bank frequently, because changing the row in a bank causes the bank to close and reopen to open another row in that bank.

Auto-Precharge Commands

The auto-precharge read and write commands allow you to indicate to the memory device that this read or write command is the last access to the currently opened row. The memory device automatically closes or auto-precharges the page it is currently accessing, so that the next access to the same bank is quicker. This command is useful when performing fast random memory accesses.

The Timer bank pool (TBP) block supports the dynamic page policy, where depending on user input on local autoprecharge input would keep a page open or close. In close-page policy, a page is always closed after it is accessed with auto-precharge command. With the data pattern consists of repeated reads or writes to addresses not within the same page, the optimal system shall achieve the maximum efficiency allowed by continuous page miss limited access. There shall be no other efficiency losses other than that associated with activating and refreshing. An efficiency of 10-20% should be expected for this close page policy.

In open-page policy, the page is always kept open after accessing it for incoming commands. With the data pattern consists of repeated reads or writes to sequential addresses within the same page, the optimal system will achieve 100% efficiency (after ignoring the effects of periodic refreshes, which typically consume around 2-3% of total efficiency) for page open transactions, with minimum latency for any highest priority single transactions.

If you turn on **Enable Auto-Precharge Control**, you can request the controller to issue an autoprecharge read or write command. The next time you need to use that bank, the access could be quicker as the controller does not need to precharge the bank before activating the row you wish to access.

The controller-derived autoprecharge logic evaluates the pending commands in the command buffer and determines the most efficient autoprecharge operation to perform, reordering commands if necessary. When all TBP are occupied due to tracking an open page, TBP will employ a scheme called on-demand flush where it will stop tracking a page to give space to incoming command.

Figure 14–4 shows a comparison of auto-precharge with and without look-ahead support.

Figure 14–4. Comparison With and Without Look-ahead Auto-Precharge

Without Look-ahead Auto-Precharge			Look-ahead Auto-Precharge		
Cycle	Command	Data	Cycle	Command	Data
1	WRITE		1	WRITE with AP	
2	NOP	DATA0 (Burst 0, Burst 1)	2	NOP	DATA0 (Burst 0, Burst 1)
3	ACT	DATA0 (Burst 2, Burst 3)	3	ACT	DATA0 (Burst 2, Burst 3)
4	NOP	DATA0 (Burst 4, Burst 5)	4	NOP	DATA0 (Burst 4, Burst 5)
5	WRITE	DATA0 (Burst 6, Burst 7)	5	WRITE	DATA0 (Burst 6, Burst 7)
6	NOP	DATA1 (Burst 0, Burst 1)	6	NOP	DATA1 (Burst 0, Burst 1)
7	ACT	DATA1 (Burst 2, Burst 3)	7	ACT	DATA1 (Burst 2, Burst 3)
8	NOP	DATA1 (Burst 4, Burst 5)	8	NOP	DATA1 (Burst 4, Burst 5)
9	WRITE	DATA1 (Burst 6, Burst 7)	9	WRITE	DATA1 (Burst 6, Burst 7)
10	NOP	DATA2 (Burst 0, Burst 1)	10	NOP	DATA2 (Burst 0, Burst 1)
11	PCH	DATA2 (Burst 2, Burst 3)	11	ACT	DATA2 (Burst 2, Burst 3)
12	NOP	DATA2 (Burst 4, Burst 5)	12	NOP	DATA2 (Burst 4, Burst 5)
13	ACT	DATA2 (Burst 6, Burst 7)	13	WRITE	DATA2 (Burst 6, Burst 7)
14	NOP	Wasted Cycle	14	NOP	DATA3 (Burst 0, Burst 1)
15	WRITE	Wasted Cycle	15	NOP	DATA3 (Burst 2, Burst 3)
16	NOP	DATA3 (Burst 0, Burst 1)	16	NOP	DATA3 (Burst 4, Burst 5)
17	NOP	DATA3 (Burst 2, Burst 3)	17	NOP	DATA3 (Burst 6, Burst 7)
18	NOP	DATA3 (Burst 4, Burst 5)			
19	NOP	DATA3 (Burst 6, Burst 7)			

Command	Bank	Row	Condition
Write	Bank 0	Row 0	
Write	Bank 1	Row 0	Activate required
Write	Bank 2	Row 0	Activate required
Write	Bank 0	Row 1	Precharge required

Without the look-ahead auto-precharge support, the controller will precharge to close and then open the row before the write or read burst for every row change. With the look-ahead precharge feature in controller, Controller decides whether to do auto-precharge read/write by looking ahead the incoming command, and subsequent reads or writes to same bank/different row only require an activate command. As shown in figure, the controller perform an auto-precharge for the write command to bank 0 at cycle 1 as it detects the next write at cycle 13 is to different row in bank 0, and hence save 2 valuable data cycle.

Compare the following efficiency results for Figure 14–4:

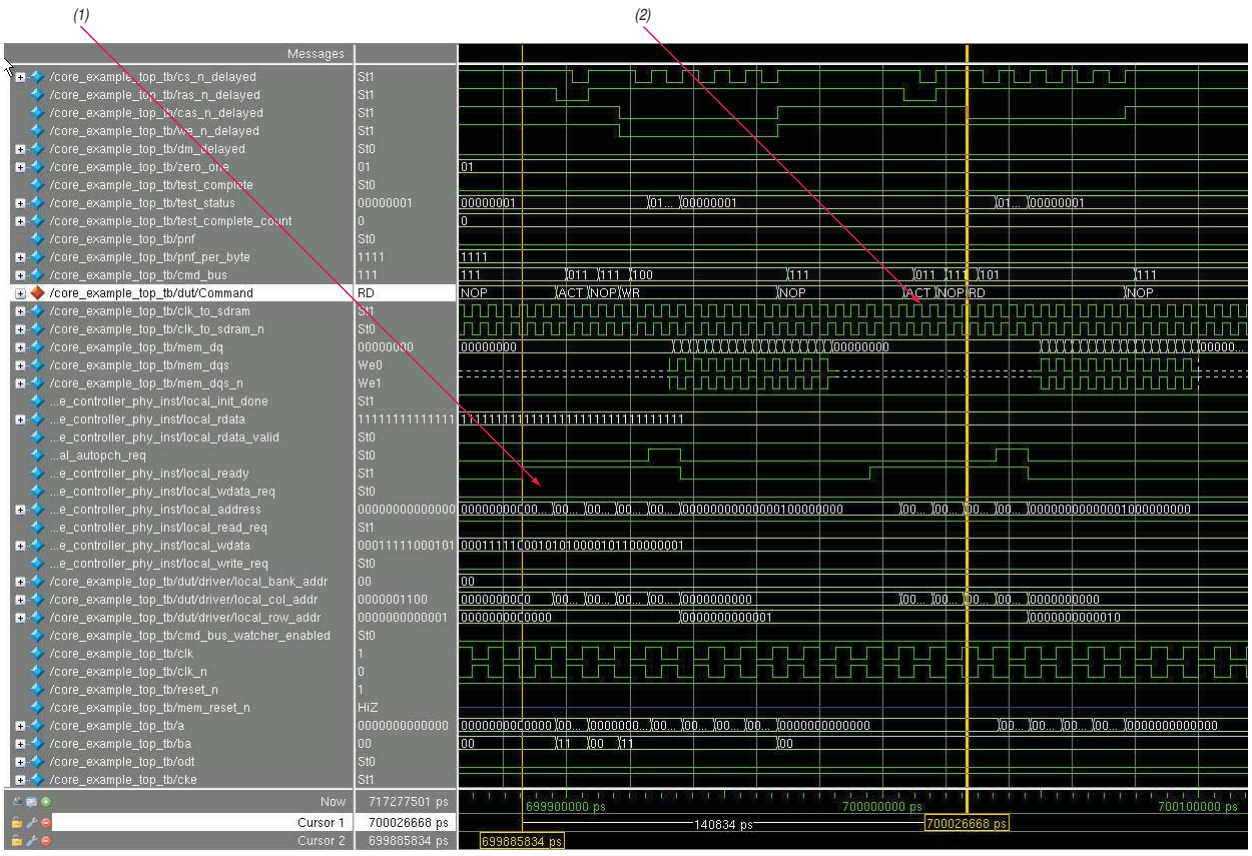
Table 14–1. Compatative Efficiencies With and Without Look-Ahead Auto-Precharge Feature

	Without Look-ahead Auto-precharge	With Look-ahead Auto-precharge
Active cycles of data transfer	16	16
Total number of cycles	19	17
Approximate efficiency	84%	94%

The look-ahead auto-precharge used increases efficiency by approximately 10%.

Figure 14–5 shows how you can improve controller efficiency using the auto-precharge command.

Figure 14–5. Improving Efficiency Using Auto-Precharge Command



The following sequence of events describes Figure 14–5:

1. The controller accepts a read request from the local side as soon as the `local_ready` signal goes high.
2. The controller gives the activate command and then gives the read command. The read command latency is approximately 14 clock cycles for this case as compared to the similar case with no auto precharge which had approximately 17 clock cycles of latency (described in Figure 14–3).

When using the auto-precharge option, note the following guidelines:

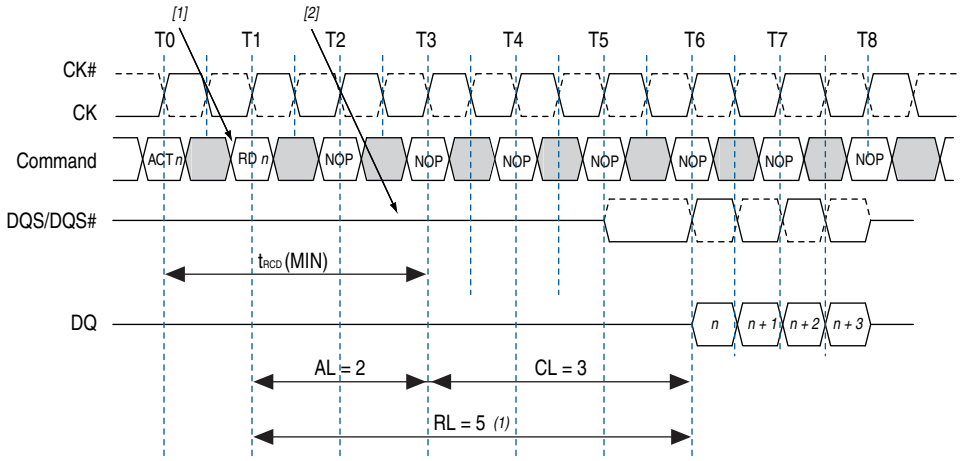
- Use the auto-precharge command if you know the controller is issuing the next read or write to a particular bank and a different row.
- Auto-precharge does not improve efficiency if you auto-precharge a row and immediately reopen it.

Additive Latency

Additive latency increases the efficiency of the command and data bus for sustainable bandwidths. You may issue the commands externally but the device holds the commands internally for the duration of additive latency before executing, to improve the system scheduling. The delay helps to avoid collision on the command bus and gaps in data input or output bursts. Additive latency allows the controller to issue the row and column address commands—activate, and read or write—in consecutive clock cycles, so that the controller need not hold the column address for several (t_{RCD}) cycles. This gap between the activate and the read or write command can cause bubbles in the data stream.

Figure 14–6 shows an example of additive latency.

Figure 14–6. Additive Latency—Read



The following sequence of events describes Figure 14–6:

1. The controller issues a read or write command before the t_{RCD} (MIN) requirement— additive latency less than or equal to t_{RCD} (MIN).

2. The controller holds the read or write command for the time defined by additive latency before issuing it internally to the SDRAM device.

   Read latency = additive latency + CAS latency

   Write latency = additive latency + CAS latency – t_{CK}

Bank Interleaving

You can use bank interleaving to sustain bus efficiency when the controller misses a page, and that page is in a different bank.

Page size refers to the minimum number of column locations on any row that you access with a single activate command. For example:

For a 512Mb x8 DDR3 SDRAM with 1024 column locations (column address A[9:0]), page size = 1024 columns x 8 = 8192 bits = 8192/8 bytes = 1024 bytes (1 KB)

Without interleaving, the controller sends the address to the SDRAM device, receives the data requested, and then waits for the SDRAM device to precharge and reactivate before initiating the next data transaction, thus wasting several clock cycles.

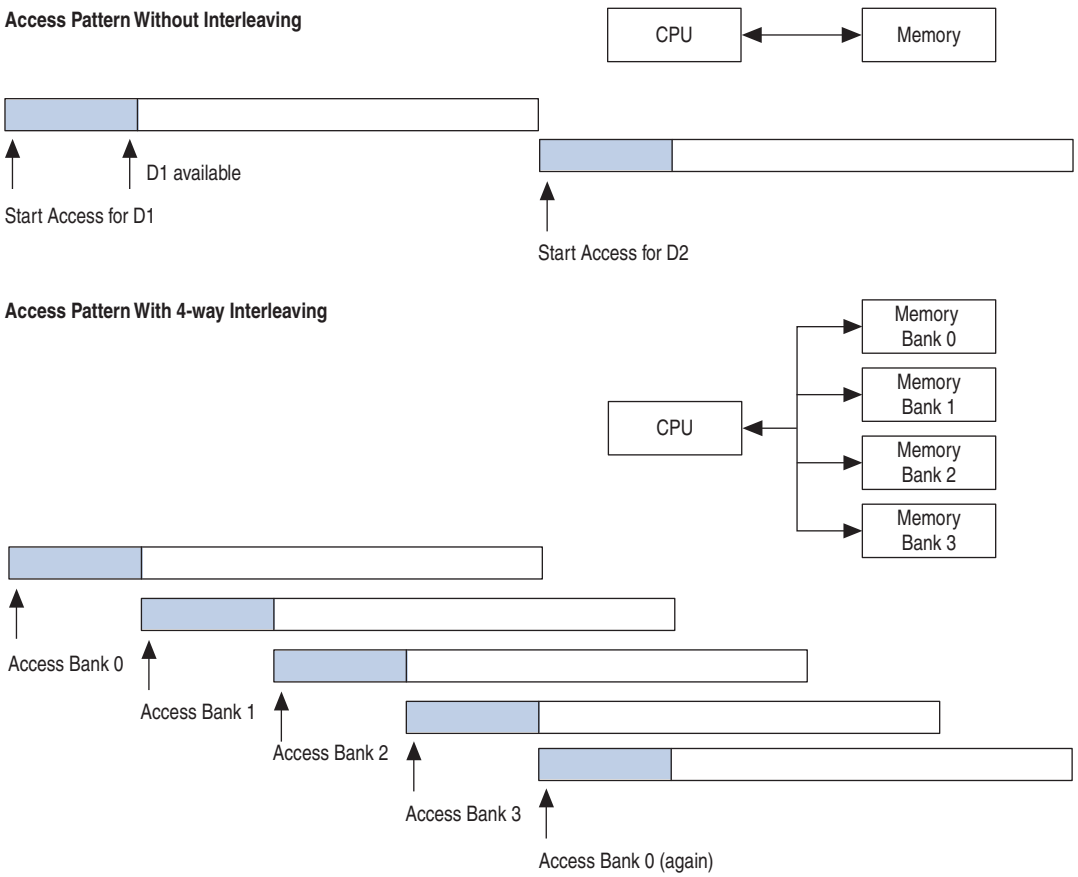
Interleaving allows banks of the SDRAM device to alternate their background operations and access cycles. One bank undergoes its precharge/activate cycle while another is being accessed. By alternating banks, the controller improves its performance by masking the precharge/activate time of each bank. If there are four banks in the system, the controller can ideally send one data request to each of the banks in consecutive clock cycles.

For example, in the first clock cycle, the CPU sends an address to Bank 0, and then sends the next address to Bank 1 in the second clock cycle, before sending the third and fourth addresses to Banks 2 and 3 in the third and fourth clock cycles respectively. The sequence is as follows:

1. Controller sends address 0 to Bank 0.
2. Controller sends address 1 to Bank 1 and receives data 0 from Bank 0.
3. Controller sends address 2 to Bank 2 and receives data 1 from Bank 1.
4. Controller sends address 3 to Bank 3 and receives data 2 from Bank 2.
5. Controller receives data 3 from Bank 3.

Figure 14–7 shows how you can use interleaving to increase bandwidth.

Figure 14–7. Using Interleaving to Increase Bandwidth



The Altera controller supports three interleaving options:

Chip-Bank-Row-Col – This is a noninterleaved option. Select this option to improve efficiency with random traffic

Chip-Row-Bank-Col – This option uses bank interleaving without chip select interleaving. Select this option to improve efficiency with sequential traffic, by spreading smaller data structures across all banks in a chip.

Row-Chip-Bank-Col - This option uses bank interleaving with chip select interleaving. Select this option to improve efficiency with sequential traffic and multiple chip selects. This option allows smaller data structures to spread across multiple banks and chips.

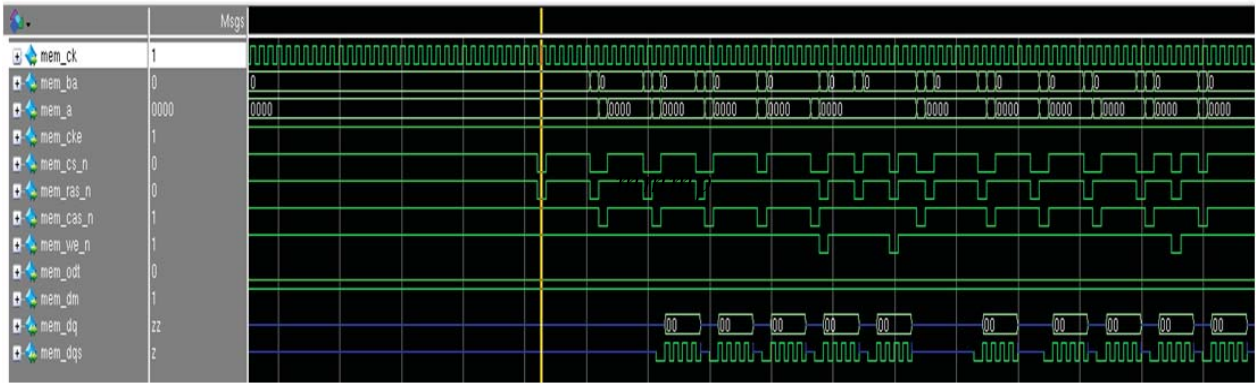
Bank interleaving is a fixed pattern of data transactions, enabling best-case bandwidth and latency, and allowing for sufficient interleaved transactions between opening banks to completely hide t_{RC}. An optimal system can achieve 100% efficiency for bank interleave transactions with 8 banks. A system with less than 8 banks is unlikely to achieve 100%.

Command Queue Look-Ahead Depth

The command queue look-ahead depth value determines the number of read or write requests that the look-ahead bank management logic examines. The command queue look-ahead depth value also determines how many open pages the High-Performance Controller II (HPC II) can track.

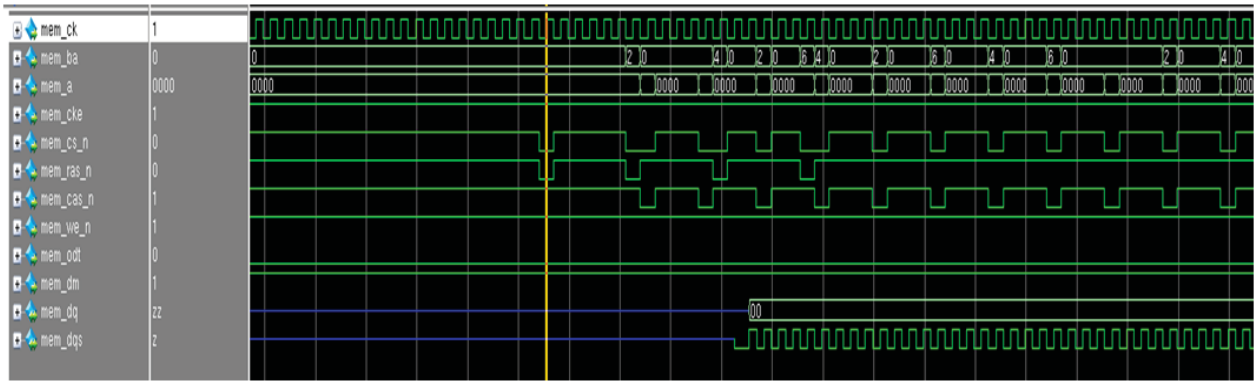
For example, if you set the command queue look-ahead depth value to 4, the HPC II controller can track 4 open pages. In a 4-bank interleaving case, HPCII will receive repeated commands with addresses of bank A, bank B, bank C, and bank D. To receive the next set of commands, the controller issues a precharge command to exit the current page and then issues an activate command to track the new incoming page, leading to a drop in efficiency.

Figure 14–8. Simulation with Command Queue Look-ahead Depth of 4



With the command queue look-ahead depth set to 8, the controller can track 8 open pages and overall efficiency is much improved relative to a command queue look-ahead value of 4.

Figure 14–9. Simulation with Command Queue Look-ahead Depth of 8



There is a trade-off between efficiency and resource usage. Higher command queue look-ahead values are likely to increase bank management efficiency, but at the cost of higher resource usage. Smaller command queue look-ahead values may be less efficient, but also consume fewer resources. Also, a command queue look-ahead value greater than 4 may cause timing violations for interfaces approaching their maximum frequency.

If you set Command Queue Look-ahead depth to a value greater than 4, you may not be able to run the interface at maximum frequency.

To achieve an optimized balance of controller efficiency versus resource usage and frequency, you must understand your traffic patterns. You should simulate your design with a variety of controller settings to observe the results of different settings.

User-selectable Command Queue Look-ahead depth is available only when using the soft memory controller. For the hard memory controller, the Command Queue Look-ahead depth value is hard-coded to 8.

Additive Latency and Bank Interleaving

Using additive latency together with bank interleaving increases the bandwidth of the controller.

Figure 14–10 shows an example of bank interleaving in a read operation without additive latency.

Figure 14–10. Bank Interleaving—Without Additive Latency

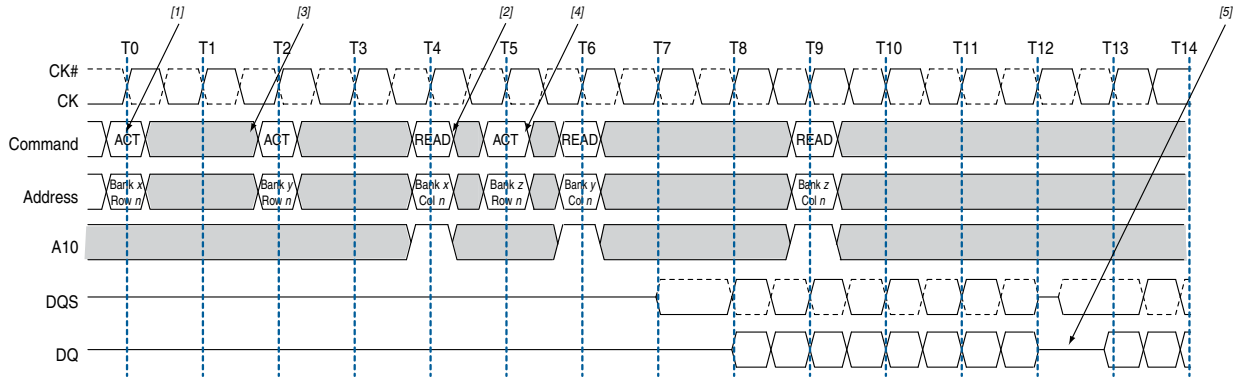


Figure 14–10 illustrates an example of DDR2 SDRAM bank interleave reads with CAS latency of 4, and burst length of 4.

The following sequence of events describes Figure 14–10:

1. The controller issues an activate command to open the bank, which activates bank *x* and the row in it.
2. After t_{RCD} time, the controller issues a read with auto-precharge command to the specified bank.

3. Bank *y* receives an activate command after t_{RRD} time.
4. The controller cannot issue an activate command to bank *z* at its optimal location because it must wait for bank *x* to receive the read with auto-precharge command, thus delaying the activate command for one clock cycle.
5. The delay in activate command causes a gap in the output data from the DDR2 SDRAM device.

If you use additive latency of 1, the latency affects only read commands and not the timing for write commands.

Figure 14–11 shows an example of bank interleaving in a read operation with additive latency. In this configuration, the controller issues back-to-back activate and read with auto-precharge commands.

Figure 14–11. Bank Interleaving—With Additive Latency

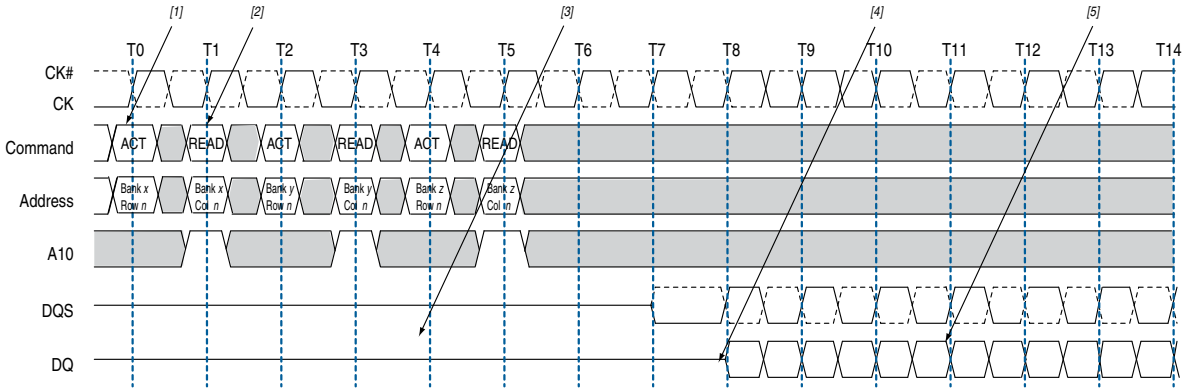


Figure 14–11 illustrates an example of a DDR2 SDRAM bank interleave reads with additive latency of 3, CAS latency of 4, and burst length of 4.

The following sequence of events describes Figure 14–11:

1. The controller issues an activate command to bank *x*.
2. The controller issues a read with auto precharge command to bank *x* right after the activate command, before waiting for the t_{RCD} time.
3. The controller executes the read with auto-precharge command t_{RCD} time later on the rising edge T4.
4. 4 cycles of CAS latency later, the SDRAM device issues the data on the data bus.
5. For burst length of 4, you need 2 cycles for data transfer. With 2 clocks of giving activate and read with auto-precharge commands, you get a continuous flow of output data.

Compare the following efficiency results in Figure 14–10 and Figure 14–11:

- DDR2 SDRAM bank interleave reads with no additive latency, CAS latency of 4, and burst length of 4 (Figure 14–10),

  Number of active cycles of data transfer = 6.

  Total number of cycles = 15

  Efficiency = 40%

- DDR2 SDRAM bank interleave reads with additive latency of 3, CAS latency of 4, and burst length of 4 (Figure 14–11),

  Number of active cycles of data transfer = 6.

  Total number of cycles = 14

  Efficiency = approximately 43%

The interleaving reads used with additive latency increases efficiency by approximately 3%.

Additive latency improves the efficiency of back-to-back interleaved reads or writes, but not individual random reads or writes.

User-Controlled Refresh

The requirement to periodically refresh memory contents is normally handled by the memory controller; however, the **User Controlled Refresh** option allows you to determine when memory refresh occurs. With specific knowledge of traffic patterns, you can time the refresh operations so that they do not interrupt read or write operations, thus improving efficiency.

If you enable the auto-precharge control, you must ensure that the average periodic refresh requirement is met, because the controller does not issue any refreshes until you instruct it to.

Frequency of Operation

Certain frequencies of operation give you the best possible latency based on the memory parameters. The memory parameters you specify through the parameter editor in the MegaWizard™ Plug-In Manager are converted to clock cycles and rounded up.

If you are using a memory device that has t_{RCD} = 20 ns and running the interface at 100 MHz, you get the following results:

- For full-rate implementation (t_{Ck} = 10 ns):

  t_{RCD} convert to clock cycle = 20/10 = 2.

- For half rate implementation (t_{Ck} = 20 ns):

  t_{RCD} convert to clock cycle = 20/20 = 1

This frequency and parameter combination is not easy to find because there are many memory parameters and frequencies for the memory device and the controller to run. Memory device parameters are optimal for the speed at which the device is designed to run, so you should run the device at that speed.

In most cases, the frequency and parameter combination is not optimal. If you are using a memory device that has t_{RCD} = 20 ns and running the interface at 133 MHz, you get the following results:

- For full-rate implementation (t_{Ck} = 7.5 ns):

  t_{RCD} convert to clock cycle = 20/7.5 = 2.66, rounded up to 3 clock cycles or 22.5 ns.

- For half rate implementation (t_{Ck} = 15 ns):

  t_{RCD} convert to clock cycle = 20/15 = 1.33, rounded up to 2 clock cycles or 30 ns.

There is no latency difference for this frequency and parameter combination.

Burst Length

Burst length affects the efficiency of the controller. A burst length of 8 provides more cycles of data transfer, compared to a burst length of 4.

For a half-rate design that has a command latency of 9 half-rate clock cycles, and a CAS latency of 3 memory clock cycles or 1.5 half rate local clock cycles, the efficiency is 9% for burst length of 4, and 16% for burst length of 8.

- Burst length of 4 (2 memory clock cycles of data transfer or 1 half-rate local clock cycle)

  Efficiency = number of active cycles of data transfer/total number of cycles

  Efficiency = 1/(1 + CAS + command latency) = 1/(1 + 1.5 + 9) = 1/11.5 = 8.6% or approximately 9%

- Burst length of 8 (4 memory clock cycles of data transfer or 2 half-rate local clock cycles)

  Efficiency = number of active cycles of data transfer/total number of cycles

  Efficiency = 2/(2 + CAS + command latency) = 2/(2 + 1.5 + 9) = 2/12.5 = 16%

Series of Reads or Writes

Performing a series of reads or writes from the same bank and row increases controller efficiency.

The case shown in Figure 14–2 on page 14–4 demonstrates that a read performed from the same row takes only 14.5 clock cycles to transfer data, making the controller 27% efficient.

Do not perform random reads or random writes. When you perform reads and writes to random locations, the operations require row and bank changes. To change banks, the controller must precharge the previous bank and activate the row in the new bank. Even if you change the row in the same bank, the controller has to close the bank (precharge) and reopen it again just to open a new row (activate). Because of the precharge and activate commands, efficiency can decrease by as much as 3–15%, as the controller needs more time to issue a read or write.

If you must perform a random read or write, use additive latency and bank interleaving to increase efficiency.

Controller efficiency depends on the method of data transfer between the memory device and the FPGA, the memory standards specified by the memory device vendor, and the type of memory controller.

Reordering

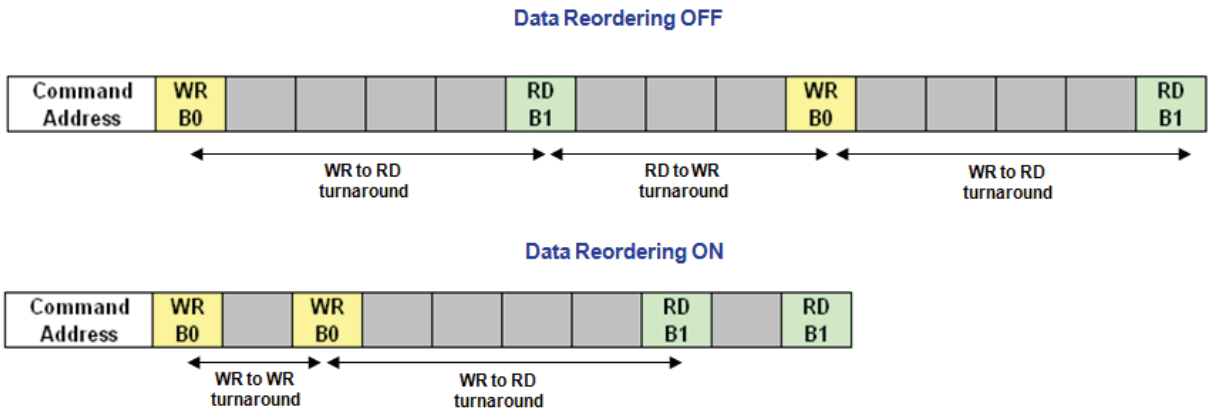
Data reordering and command reordering can both contribute towards achieving controller efficiency.

Data Reordering

The Data Reordering feature allows the single-port memory controller to change the order of read and write commands to achieve highest efficiency. You can enable data reordering by turning on **Enable Reordering** on the **Controller Settings** tab of the parameter editor.

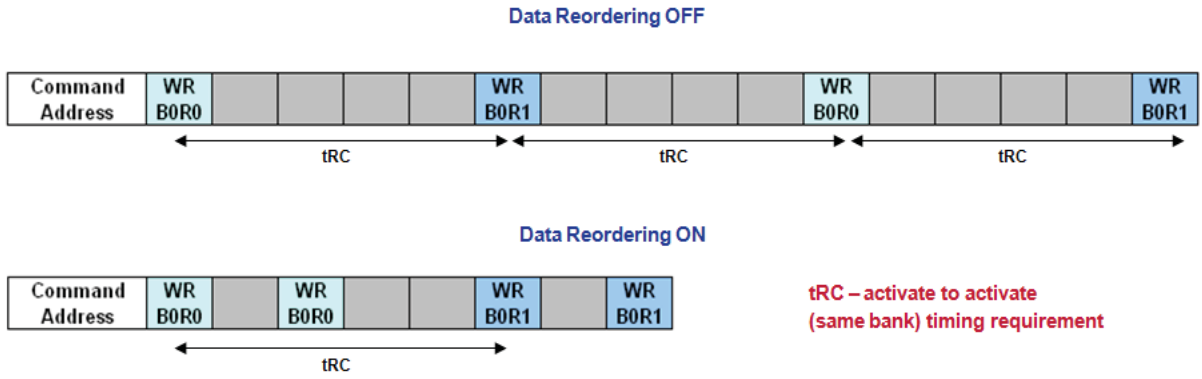
In the soft memory controller, inter-bank data reordering serves to minimize bus turnaround time by optimizing the ordering of read and write commands going to different banks; commands going to the same bank address are not reordered.

Figure 14–12. Data Reordering for Minimum Bus Turnaround



In the hard memory controller, inter-row data reordering serves to minimize t_{RC} by reordering commands going to different bank and row addresses; command going to the same bank and row address are not reordered. Inter-row data reordering inherits the minimum bus turnaround time benefit from inter-bank data reordering.

Figure 14–13. Data Reordering for Minimum t_{RC}



Starvation Control

The controller implements a starvation counter to ensure that lower-priority requests are not forgotten as higher-priority requests are reordered for efficiency. In starvation control, a counter is incremented for every command served. You can set a starvation limit, to ensure that a waiting command is served immediately upon the starvation counter reaching the specified limit.

For example, if you set a starvation limit of 10, a lower-priority command will be treated as high priority and served immediately, after ten other commands are served before it.

Command Reordering

DDR protocols are naturally inefficient, because commands are fetched and processed sequentially. The DDRx command and DQ bus are not fully utilized as few potential cycles are wasted and degrading the efficiency

The command reordering feature, or look-ahead bank management feature, allows the controller to issue bank management commands early based on incoming patterns, so that when the command reaches the memory interface, the desired page in memory is already open.

The command cycles during the t_{RCD} period are idle and the bank-management commands are issued to next access banks. When the controller is serving the next command, the bank is already precharged. The command queue look-ahead depth is configurable from 1-16, to specify how many read or write requests the look-ahead bank management logic examines. With the look-ahead command queue, if consecutive write or read requests are to a sequential address with same row, same

bank, and column incremental by 1, the controller merges the write or read requests at the memory transaction into a single burst.

Figure 14–14. Comparison With and Without Look-Ahead Bank Mangement Feature

Compare the following efficiency results for Figure 14–14:

Table 14–2. Efficiency Results for Figure 14–14

	Without Look-ahead Bank Management	With Look-ahead Bank Management
Active cycles of data transfer	12	12
Total number of cycles	20	16
Approximate efficiency	60%	75%

In Table 14–2, the use of look-ahead bank management increases efficiency by 15%. The bank look-ahead pattern verifies that the system is able to completely hide the bank precharge and activation for specific sequences in which the minimum number of page-open transactions are placed between transactions to closed pages to allow bank look-ahead to occur just in time for the closed pages. An optimal system would completely hide bank activation and precharge performance penalties for the bank look-ahead traffic pattern and achieve 100% efficiency, ignoring refresh.

Bandwidth

Bandwidth depends on the efficiency of the memory controller controlling the data transfer to and from the memory device.

You can express bandwidth as follows:

Bandwidth = data width (bits) × data transfer rate (1/s) × efficiency

Data rate transfer (1/s) = 2 × frequency of operation (4 × for QDR SRAM interfaces)

The following example shows the bandwidth calculation for a 16-bit interface that has 70% efficiency and runs at 200 MHz frequency:

Bandwidth = 16 bits × 2 clock edges × 200 MHz × 70% = 4.48 Gbps.

DRAM typically has an efficiency of around 70%, but when you use the Altera® memory controller efficiency can vary from 10 to 92%.

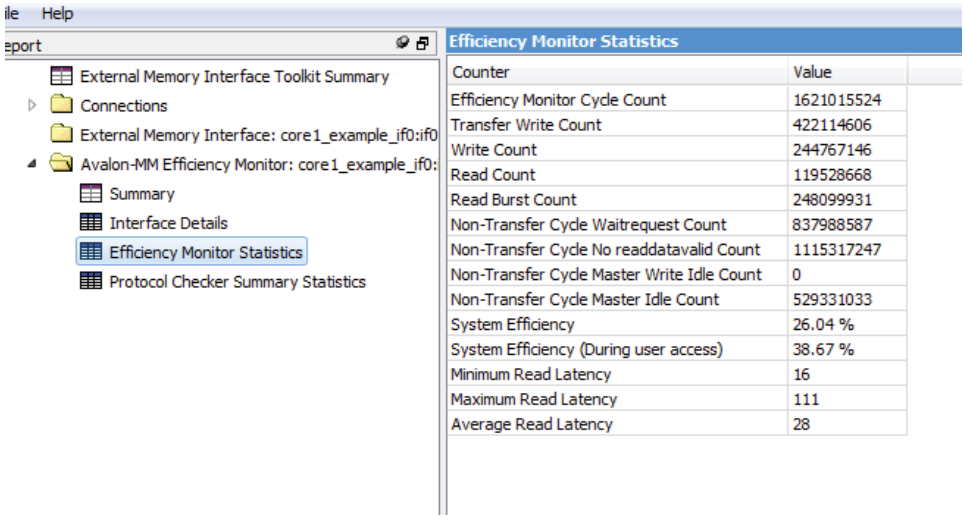
In QDR II+ or QDR II SRAM the IP implements two separate unidirectional write and read data buses, so the data transfer rate is four times the clock rate. The data transfer rate for a 400-MHz interface is 1, 600 Mbps. The efficiency is the percentage of time the data bus is transferring data. It is dependent on the type of memory. For example, in a QDR II+ or QDR II SRAM interface with separate write and read ports, the efficiency is 100% when there is an equal number of read and write operations on these memory interfaces.

For information on best-case and worst-case efficiency scenarios, refer to the white paper, *The Efficiency of the DDR & DDR2 SDRAM Controller Compiler*.

Efficiency Monitor

The Efficiency Monitor and Protocol Checker measures traffic efficiency on the Avalon interface between the traffic generator and the controller, and checks that the Avalon protocol is not violated. The Protocol Checker monitors the controller's Avalon slave interface for any illegal commands presented to it by any master; it does not monitor the legality of the controller's Avalon outputs.

Figure 14–15. Efficiency Monitor and Protocol Checker Statistics



To enable efficiency measurements to be performed on the controller Avalon interface through UniPHY External Memory Interface Toolkit, you need to check on Enable the Efficiency Monitor and Protocol Checker on the Controller Avalon Interface.

The Efficiency Monitor counts the number of cycles of command transfers and wait times for the controller interface and provides an Avalon slave port to allow access to this data. The efficiency monitor has an internal 32-bit counter for accessing transactions; its status can be any of the following:

- Not Running
- Not Running: Waiting for pattern start
- Running
- Not Running: Counter Saturation

For example, once the counter saturates the efficiency monitor stops because it can no longer track transactions. In the summary panel, this appears as `Not Running: Counter Saturation`.

The debug toolkit summarizes efficiency monitor statistics as follows:

- **Efficiency Monitor Cycle Count –** counts cycles from first command/start until 2^{32} or a stop request
- **Transfer Write Count –** counts any data transfer cycle, read or write
- **Write Count –** counts how many writes requested, including those during a burst

- **Read Count –** counts how many reads requested (just commands)
- **Read Burst counter –** counts how many reads requested (total burst requests)
- **Non-Transfer Cycle Waitrequest Count –** counts Non Transfer Cycles (NON-TRANSFER CYCLE) due to slave wait request high.
- **Non-Transfer Cycle No readdatavalid Count –** counts Non Transfer Cycles (NON-TRANSFER CYCLE) due to slave not having read data
- **Non-Transfer Cycle Master Write Idle Count –** counts Non Transfer Cycles (NON-TRANSFER CYCLE) due master not issuing command or pause in write burst
- **Non-Transfer Cycle Master Idle Count –** counts Non Transfer Cycles (NON-TRANSFER CYCLE) due master not issuing command anytime
- **System Efficiency –** The total number of Avalon-MM cycles where the interface is not stalled which takes an idle interface into account.
- **System Efficiency (During user access) –** tracks the efficiency when transactions are occurring, which is basically a reflection on waitrequest, which is defined as:

  System Efficiency (During user access) = Transfer Write Count/(Efficiency Monitor Cycle Count - Non-Transfer Cycle Master Idle Count)
- **Minimum Read Latency –** The lowest of all read latencies, which is measured by time between a read command is being accepted by the Controller till the first beat of read data is presented to the driver.
- **Maximum Read Latency –** The highest of all read latencies, which is measured by time between a read command is being accepted by the Controller till the first beat of read data is presented to the driver.
- **Average Read Latency –** The average of all read latencies, which is measured by time between a read command is being accepted by the Controller till the first beat of read data is presented to the driver.

Document Revision History

Table 14–3 lists the revision history for this document.

Table 14–3. Document Revision History

Date	Version	Changes
November 2012	3.1	Changed chapter number from 13 to 14.
June 2012	3.0	■ Expanded Auto-Precharge Commands section. ■ Expanded Bank Interleaving section. ■ Added Command Queue Look-Ahead Depth section. ■ Added Reordering section. ■ Added Efficiency Monitor section. ■ Added Feedback icon.
November 2011	2.0	Reorganized optimizing the controller information into an individual chapter.
June 2011	1.0	Initial release.

EMI_DG_014-1.2

This chapter describes the design considerations that affect the external memory interface performance and the device resource usage when you use UniPHY IP in your design.

Core Logic and User Interface Data Rate

The clocking operation in the PHY is categorized into the following two domains:

- PHY-memory domain—the PHY interfaces with the external memory device and is always at full-rate.
- PHY-AFI domain—the PHY interfaces with the memory controller and can either be at full, half or quarter rate of the memory clock depending on your choice of controller and PHY.

For the memory controller to operate at full, half and quarter data rate, the UniPHY IP supports full, half and quarter data rate. The data rate defines the ratio between the frequency of the Altera® PHY Interface (AFI) clock and the frequency of the memory device clock.

Table 15–2 compares the clock cycles, data bus width and address/command bus width between the full-, half-, and quarter-rate designs.

Table 15–1. Ratio between Clock Cycles, Data Bus Width, and Address/Command Bus Width

Data Rate	Controller Clock Cycles	Bus Width	
		AFI Data	AFI Address/Command
Full	1	2	1
Half	2	4	2
Quarter	4	8	4

In general, full-rate designs require smaller data and address/command bus width. However, because the core logic runs at a high frequency, full rate designs might have difficulties in closing timing. As such, for high frequency memory interface designs, Altera recommends that you use half-rate or quarter-rate UniPHY IP and controllers.

DDR3 SDRAM interfaces are capable of running at much higher frequencies as compared to the DDR, DDR2 SDRAM, QDRII, QDRII+ SRAM, and RLDRAM II interfaces. For this reason, Altera High-Performance Controller II and UniPHY IPs do not support full rate designs using the DDR3 SDRAM interface. However, DDR3 hard controller in Arria® V devices only support full rate. Quarter rate design support is for DDR3 SDRAM interfaces targeting frequencies higher than 667 MHz.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

While it is easier to close timing for half-rate and quarter-rate designs due to the lower frequency required on the core logic, full-rate interface offer better efficiency for low burst-length designs because of 1T addressing mode where the address and command signals are asserted for one memory clock cycle. Typically half-rate and quarter-rate designs operate in 2T and 4T mode, respectively, in which the address & command signals in 2T and 4T mode must be asserted for two and four memory clock cycles, respectively. To improve efficiency, the controller can operate in Quasi-1T half-rate and Quasi-2T quarter-rate modes. In Quasi-1T half-rate mode, two commands are issued to the memory on two memory clock cycles. In Quasi-2T quarter-rate mode, two commands are issued to the memory on four memory clock cycles. The controller is constrained to issue a row command on the first clock phase and a column command on the second clock phase, or vice versa. Row commands include activate and precharge commands; column commands include read and write commands.

Hard and Soft Memory PHY

The Arria V and Cyclone® V device families support hard and soft memory interfaces. Hard memory interfaces use the hard memory controllers and hard memory PHY blocks in the devices.

Currently the hard memory PHY is instantiated together with the hard memory controller. In addition to the PHY data path that uses the hard IP blocks in the devices (similar to how the soft PHY is implemented for device families supported by UniPHY), the hard memory PHY also uses the dedicated hardware circuitries in the devices for certain component managers in the sequencer, including the read write (RW) and PHY managers.

Standalone hard memory PHY instantiation will be supported in future versions of the Quartus® II software.

In soft memory PHY, the UniPHY sequencer implements the Nios® II processor and all the component managers in the core logic. The hard memory PHY uses dedicated hard IP blocks in the Arria V and Cyclone V devices to implement the RW and PHY managers to save LE resources, and to allow better performance and lower latency.

Each Arria V and Cyclone V device has a fixed number of hard PHYs. Dedicated I/O pins with specific functions for data, strobe, address, command, control, and clock must be used together with each hard PHY.

For the list of hard PHY dedicated pins, refer to the device pin-out files for your target device on the Pin-Out Files for Altera Devices page of the Altera website.

Using the soft memory PHY gives you the flexibility to choose the pins to be used for the memory interface. Soft memory PHY also supports wider interfaces as compared to hard memory PHY.

Sequencer

Starting from Quartus II software version 11.0, the UniPHY IP soft memory PHY supports the following two types of sequencer used for QDRII and QDRII+ SRAM, and RLDRAM II calibration:

- RTL-based sequencer
- Nios II-based sequencer

The RTL-based sequencer performs FIFO calibration that includes adjusting the valid-prediction FIFO (VFIFO) and latency FIFO (LFIFO) length. On top of the FIFO calibration, the Nios II-based sequencer also performs I/O calibration that includes adjusting delay chains and phase settings to center-align the data pins with respect to the strobes that sample them. I/O calibration is required for memory interfaces running at higher frequencies to increase the read and write margin.

Because the RTL-based sequencer performs relatively simpler calibration process, it does not require a Nios II processor. For this reason, the resource utilization like LE and RAM usage is lower as compared to the Nios II-based sequencer.

For more information about the RTL-based sequencer and Nios II-based sequencer, refer to the *Functional Description—UniPHY* chapter in volume 3 of the *External Memory Interface Handbook*.

For more information about the calibration process, refer to the "UniPHY Calibration Stages" section in the *Functional Description—UniPHY* chapter of the *External Memory Interface Handbook*.

PLL, DLL and OCT Resource Sharing

By default, each external memory interface in a device needs one PLL, one DLL and one OCT control block. Due to the fixed number of PLL, DLL and OCT resources available in a device, these resources can be shared by two or more memory interfaces when certain criterias are met. This method allows more memory interfaces to fit into a device and allows the remaining resources to be used for other purposes.

By sharing PLLs, apart from reducing the number of PLLs to be used, the number of clock networks and the clock input pins required are also reduced. To share PLLs, the memory interfaces must meet the following criterias:

- Run the same memory protocol (for example, DDR3 SDRAM)
- Run at the same frequency
- The controllers or PHYs run at the same rate (for example, half rate)
- Use the same phase requirements (for example, additional core-to-periphery clock phase of 90°)
- The memory interfaces are located on the same side of the device, or adjacent sides of the device if the PLL is able to drive both sides.

Altera devices have up to four DLLs available to perform phase shift on the DQS signal for capturing the read data. The DLLs are located at the device corners and some of the DLLs can access two adjacent sides of the device. To share DLLs, the memory interfaces must meet the following criterias:

- Run at the same frequency
- The memory interfaces are located on the same side of the device, or adjacent sides of the device accessible by the DLL.

Memory interface pins with OCT calibration requires the OCT control block to calibrate the OCT resistance value. Depending on the device family, the OCT control block uses either the RUP and RDN, or RZQ pins for OCT calibration. Each OCT control block can only be shared by pins powered by the same VCCIO level. Sharing of the OCT control block by interfaces operating at the same VCCIO level allows other OCT control blocks in the device to support other VCCIO levels. The unused RUP/RDN or RZQ pins can also be used for other purposes. For example, the RUP/RDN pins can be used as DQ or DQS pins. To share OCT control block, the memory interfaces must operate at the same VCCIO level.

For more information about the resources required for memory interfaces in various device families, refer to the *Planning Pin and FPGA Resources* chapter.

For more information about how to share PLL, DLL and OCT control block, refer to the *Functional Description—UniPHY* chapter in volume 3 of the *External Memory Interface Handbook*.

For more information about the DLL, refer to the external memory interface chapters in the respective device handbooks.

For more information about the OCT control block, refer to the I/O features chapters in the respective device handbooks.

Pin Placement Consideration

The Stratix® V, Arria V, and Cyclone V device families use the PHY clock (PHYCLK) networks to clock the external memory interface pins for better performance. Each PHYCLK network is driven by a PLL. In Cyclone V and Stratix V devices, the PHYCLK network spans across two I/O banks on the same side of the device, whereas for Arria V devices, each PHYCLK network spans across one I/O bank. As such, all pins for a memory interface must be placed on the same side of the device.

For more information about pin placement guidelines related to the PHYCLK network, refer to the *External Memory Interfaces in Stratix V Devices* chapter in volume 2 of the *Stratix V Device Handbook*, *External Memory Interfaces in Arria V Devices* chapter in volume 2 of the *Arria V Device Handbook*, or the *External Memory Interfaces in Cyclone V Devices* chapter in volume 2 of the *Cyclone V Device Handbook*.

Wraparound interface, in which data pins from a memory interface are placed on two adjacent sides of a device, and split interface, in which data pins are place on two opposite I/O banks, are supported in certain device families that do not use the PHY clock network to allow more flexibility in pin placement.

The x36 emulated mode is supported in certain device families that do not use the PHY clock network for QDRII and QDRII+ SRAM x36 interfaces. In x36 emulated mode, two x18 DQS groups or four x9 DQS groups can be combined to form a 36-bit wide write data bus, while two x18 DQS groups can be combined to form a 36-bit wide read data bus. This method allows a device to support x36 QDRII and QDRII+ SRAM interfaces even if the device does hot have the required number of x36 DQS groups.

Some device families might support wraparound or x36 emulated mode interfaces at slightly lower frequencies.

For information about the devices that support wraparound and x36 emulated mode interfaces, and the supported frequency for your design, refer to the External Memory Interface Spec Estimator page on the Altera website

For more information about x36 emulated mode support for QDRII and QDRII+ SRAM interfaces, refer to the *Planning Pin and FPGA Resources* chapter.

Document Revision History

Table 15–2 lists the revision history for this document.

Table 15–2. Document Revision History

Date	Version	Changes
November 2012	1.2	Changed chapter number from 14 to 15.
June 2012	1.1	Added Feedback icon.
November 2011	1.0	Initial release.

16. Power Estimation Methods for External Memory Interface Designs

Table 16–1 lists the Altera®-supported power estimation methods for external memory interfaces.

Table 16–1. Power Estimation Methods for External Memory Interfaces

Method	Vector Source	ALTMEMPHY Support	UniPHY Support	Accuracy	Estimation Time (1)
Early power estimator (EPE)	Not applicable	✓	✓	Lowest ↓ Highest	Fastest ↓ Slowest
Vector-less PowerPlay power analysis (PPPA)	Not applicable	✓	✓		
Vector-based PPPA	RTL simulation	✓	✓		
	Zero-delay simulation (2)	✓	✓		
	Timing simulation	(2)	(2)		

Notes to Table 16–1:

(1) To decrease the estimation time, you can skip power estimation during calibration. Power consumption during calibration is typically equivalent to power consumption during user mode.

(2) Power analysis using timing simulation vectors is not supported.

When using Altera IP, you can use the zero-delay simulation method to analyze the power required for the external memory interface. Zero-delay simulation is as accurate as timing simulation for 95% designs (designs with no glitching). For a design with glitching, power may be under estimated.

For more information about zero-delay simulation, refer to the *Power Estimation and Analysis* section in the *Quartus® II Handbook*.

The size of the vector file (**.vcd**) generated by zero-delay simulation of an Altera DDR3 SDRAM High-Performance Controller Example Design is 400 GB. The **.vcd** includes calibration and user mode activities. When vector generation of calibration phase is skipped, the vector size decreases to 1 GB.

To perform vector-based PPPA using zero-delay simulation, follow these steps:

1. Perform design compilation in the Quartus II software to generate your design's Netlist <*project_name*>**.vo**.

   The <*project_name*>**.vo** is generated in the last stage of a compile EDA Netlist Writer.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

2. In <*project_name*>**.vo**, search for the include statement for <*project_name*>**.sdo**, comment the statement out, and save the file.
3. Create a simulation script containing device model files and libraries and design specific files:
   - Netlist file for the design, <*project_name*>**.vo**
   - RTL or netlist file for the memory device
   - Testbench RTL file
4. Compile all the files.
5. Invoke simulator with commands to generate **.vcd** files.
6. Generate **.vcd** files for the parts of the design that contribute the most to power dissipation.
7. Run simulation
8. Use the generated **.vcd** files in PPPA tool as the signal activity input file.
9. Run PPPA

For more information about estimating power, refer to the *Power Estimation and Analysis* section in the *Quartus II Handbook.*

Document Revision History

Table 16–2 lists the revision history for this document.

Table 16–2. Document Revision History

Date	Version	Changes
November 2012	2.2	Changed chapter number from 15 to 16.
June 2012	2.1	Added Feedback icon.
November 2011	2.0	Reorganized power estimation methods section into an individual chapter.
April 2010	1.0	Initial release.

External Memory Interface Handbook

Volume 3: Reference Material

101 Innovation Drive
San Jose, CA 95134
www.altera.com

EMI_RM-3.0

Document last updated for Altera Complete Design Suite version: 12.1
Document publication date: November 2012

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Contents

Chapter Revision Dates xi

Section I. Functional Descriptions

Chapter 1. Functional Description—UniPHY

Block Description 1–1
I/O Pads 1–2
Reset and Clock Generation 1–2
Dedicated Clock Networks 1–3
Address and Command Datapath 1–3
Write Datapath 1–4
Leveling Circuitry 1–5
Read Datapath 1–7
Sequencer 1–8
Nios II-Based Sequencer 1–8
RTL-based Sequencer 1–13
Shadow Registers 1–14
Description 1–14
Operation 1–15
DLL Offset Control Block 1–16
Interfaces 1–16
AFI 1–17
The Memory Interface 1–17
The DLL and PLL Sharing Interface 1–18
About PLL Simulation 1–19
The OCT Sharing Interface 1–20
UniPHY Signals 1–21
PHY-to-Controller Interfaces 1–24
Using a Custom Controller 1–28
Using a Vendor-Specific Memory Model 1–29
AFI 3.0 Specification 1–29
Implementation 1–29
Bus Width and AFI Ratio 1–29
AFI Parameters 1–30
Parameters Affecting Bus Width 1–30
AFI Signals 1–32
Clock and Reset Signals 1–32
Address and Command Signals 1–32
Write Data Signals 1–34
Read Data Signals 1–35
Calibration Status Signals 1–35
Tracking Management Signals 1–36
Register Maps 1–37
UniPHY Register Map 1–37
Controller Register Map 1–39
Ping Pong PHY 1–39
Feature Description 1–39
Architecture 1–40

Ping Pong Gasket . . . 1–41
Calibration . . . 1–42
Operation . . . 1–42
Efficiency Monitor and Protocol Checker . . . 1–42
Efficiency Monitor . . . 1–42
Protocol Checker . . . 1–43
Read Latency Counter . . . 1–43
Using the Efficiency Monitor and Protocol Checker . . . 1–43
Avalon CSR Slave and JTAG Memory Map . . . 1–43
UniPHY Calibration Stages . . . 1–45
Overview . . . 1–45
Calibration Stages . . . 1–46
Assumptions . . . 1–46
Memory Initialization . . . 1–47
Stage 1: Read Calibration Part One—DQS Enable Calibration and DQ/DQS Centering . . . 1–47
Guaranteed Write . . . 1–48
DQS Enable Calibration . . . 1–49
Centering DQ/DQS . . . 1–50
Stage 2: Write Calibration Part One . . . 1–51
Stage 3: Write Calibration Part Two—DQ/DQS Centering . . . 1–53
Stage 4: Read Calibration Part Two—Read Latency Minimization . . . 1–53
Read Latency Tuning . . . 1–53
Calibration Signals . . . 1–53
Calibration Time . . . 1–54
Document Revision History . . . 1–55

Chapter 2. Functional Description—ALTMEMPHY

Block Description . . . 2–2
Calibration . . . 2–3
Address and Command Datapath . . . 2–3
Arria II GX Devices . . . 2–3
Clock and Reset Management . . . 2–5
Clock Management . . . 2–5
Reset Management . . . 2–7
Read Datapath . . . 2–8
Arria II GX Devices . . . 2–8
ALTMEMPHY Signals . . . 2–10
PHY-to-Controller Interfaces . . . 2–16
Using a Custom Controller . . . 2–23
Preliminary Steps . . . 2–23
Design Considerations . . . 2–23
Clocks and Resets . . . 2–23
Calibration Process Requirements . . . 2–24
Other Local Interface Requirements . . . 2–24
Address and Command Interfacing . . . 2–24
Handshake Mechanism Between Read Commands and Read Data . . . 2–24
Handshake Mechanism Between Write Commands and Write Data . . . 2–25
Partial Writes . . . 2–26
Controller Register Map . . . 2–26
ALTMEMPHY Calibration Stages . . . 2–27
Enter Calibration (s_reset) . . . 2–29
Initialize PHY (s_phy_initialize) . . . 2–29
Initialize DRAM . . . 2–29
Initialize DRAM Power Up Sequence (s_int_dram) . . . 2–29

Program Mode Registers for Calibration (s_prog_mr) . . . 2–29
Write Header Information in the internal RAM (s_write_ihi) . . . 2–30
Load Training Patterns . . . 2–30
Write Block Training Pattern (s_write_btp) . . . 2–30
Write More Training Patterns (s_write_mtp) . . . 2–31
Test More Pattern Writes . . . 2–31
Calibrate Read Resynchronization Phase . . . 2–33
Initialize Read Resynchronisation Phase Calibration (s_rrp_reset) . . . 2–34
Calibrate Read Resynchronization Phase (s_rrp_sweep) . . . 2–34
Calculate Read Resynchronization Phase (s_rrp_seek) . . . 2–34
Calculate Read Data Valid Window (s_rdv) . . . 2–34
Advertize Write Latency (s_was) . . . 2–35
Calculate Read Latency (s_adv_rlat) . . . 2–35
Output Write Latency (s_adv_wlat) . . . 2–36
Calibrate Postamble (s_poa) . . . 2–36
Set Up Address and Command Clock Cycle . . . 2–37
Write User Mode Register Settings (s_prep_customer_mr_setup) . . . 2–37
Voltage and Temperature Tracking . . . 2–37
Setup the Mimic Window (s_tracking_setup) . . . 2–38
Perform Tracking (s_tracking) . . . 2–38
Document Revision History . . . 2–38

Chapter 3. Functional Description—Hard Memory Interface

Hard Memory Interface . . . 3–1
High-Level Feature Description . . . 3–1
Multi-Port Front End (MPFE) . . . 3–2
Fabric Interface . . . 3–2
Operation Ordering . . . 3–3
Multi-port Scheduling . . . 3–3
Port Scheduling . . . 3–3
DRAM Burst Scheduling . . . 3–4
DRAM Power Saving Modes . . . 3–5
MPFE Signal Descriptions . . . 3–5
Hard Memory Controller . . . 3–7
Clocking . . . 3–7
DRAM Interface . . . 3–8
ECC . . . 3–8
Controller ECC . . . 3–8
Bonding of Memory Controllers . . . 3–8
Data Return Bonding . . . 3–9
FIFO Ready . . . 3–9
Bonding Latency Impact . . . 3–9
Bonding Controller Usage . . . 3–9
Hard PHY . . . 3–10
Interconnections . . . 3–10
Clock Domains . . . 3–10
Hard Sequencer . . . 3–11
Hard Memory Interface Implementation Guidelines . . . 3–11
MPFE Setup Guidelines . . . 3–11
Soft Memory Interface to Hard Memory Interface Migration Guidelines . . . 3–12
Pin Connections . . . 3–12
Software Interface Preparation . . . 3–12
Latency . . . 3–13
Bonding Interface Guidelines . . . 3–13

Document Revision History 3–14

Chapter 4. Functional Description—HPS Memory Controller

Features of the SDRAM Controller Subsystem 4–1
SDRAM Controller Subsystem Block Diagram and System Integration 4–2
SDRAM Controller 4–2
DDR PHY 4–2
SDRAM Controller Subsystem Interfaces 4–3
MPU Subsystem Interface 4–3
L3 Interconnect Interface 4–3
CSR Interface 4–3
FPGA-to-HPS SDRAM Interface 4–3
Memory Controller Architecture 4–4
MPFE 4–5
Command Block 4–5
Write Data Block 4–6
Read Data Block 4–6
Single-Port Controller 4–6
Command Generator 4–6
Timer Bank Pool 4–6
Arbiter 4–7
Rank Timer 4–7
Write Data Buffer 4–7
ECC Block 4–7
AFI Interface 4–7
CSR Interface 4–7
Functional Description of the SDRAM Controller Subsystem 4–7
MPFE Operational Behavior 4–7
Operation Ordering 4–7
Multiport Scheduling 4–8
SDRAM Burst Scheduling 4–9
Clocking 4–10
Single-Port Controller Operational Behavior 4–10
SDRAM Interface 4–10
ECC 4–11
Interleaving Options 4–12
AXI-Exclusive Support 4–14
Memory Protection 4–14
SDRAM Power Management 4–16
DDR PHY 4–17
Clocks 4–17
Resets 4–18
Initialization 4–18
Protocol Details 4–18
SDRAM Controller Subsystem Programming Model 4–21
Initialization 4–21
Timing Parameters 4–22
SDRAM Controller Address Map and Register Definitions 4–22
Using EMI-Related HPS Features in SoC Devices 4–22
Architecture 4–22
Configuration 4–22
Simulation 4–23
Document Revision History 4–24

Chapter 5. Functional Description—HPC II Controller

Memory Controller Architecture . . . 5–1
Avalon-ST Input Interface . . . 5–2
AXI to Avalon-ST Converter . . . 5–2
Handshaking . . . 5–3
Command Channel Implementation . . . 5–3
Data Ordering . . . 5–3
Burst Types . . . 5–3
Backpressure Support . . . 5–4
Command Generator . . . 5–4
Timing Bank Pool . . . 5–4
Arbiter . . . 5–4
Arbitration Rules . . . 5–4
Rank Timer . . . 5–5
Read Data Buffer . . . 5–5
Write Data Buffer . . . 5–5
ECC Block . . . 5–5
AFI Interface . . . 5–5
CSR Interface . . . 5–5
Controller Features Descriptions . . . 5–5
Data Reordering . . . 5–5
Pre-emptive Bank Management . . . 5–6
Quasi-1T and Quasi-2T . . . 5–6
User Autoprecharge Commands . . . 5–6
Half-Rate Bridge . . . 5–6
Address and Command Decoding Logic . . . 5–7
Low-Power Logic . . . 5–7
User-Controlled Self-Refresh . . . 5–7
Automatic Power-Down with Programmable Time-Out . . . 5–8
ODT Generation Logic . . . 5–8
DDR2 SDRAM . . . 5–8
DDR3 SDRAM . . . 5–9
Burst Merging . . . 5–10
ECC . . . 5–10
Partial Writes . . . 5–11
Partial Bursts . . . 5–12
External Interfaces . . . 5–13
Clock and Reset Interface . . . 5–13
Avalon-ST Data Slave Interface . . . 5–13
AXI Data Slave Interface . . . 5–13
Enabling the AXI Interface . . . 5–13
Controller-PHY Interface . . . 5–20
Memory Side-Band Signals . . . 5–20
Self-Refresh (Low Power) Interface . . . 5–20
User-Controlled Refresh Interface . . . 5–21
Configuration and Status Register (CSR) Interface . . . 5–21
Top-Level Signals Description . . . 5–22
Controller Register Map . . . 5–29
Sequence of Operations . . . 5–33
Write Command . . . 5–33
Read Command . . . 5–33
Read-Modify-Write Command . . . 5–34
Document Revision History . . . 5–34

Chapter 6. Functional Description—QDR II Controller

Block Description . . . 6–1
Avalon-MM Slave Read and Write Interfaces . . . 6–1
Command Issuing FSM . . . 6–2
AFI . . . 6–2
Avalon-MM and Memory Data Width . . . 6–2
Signal Description . . . 6–2
Avalon-MM Slave Read Interface . . . 6–3
Avalon-MM Slave Write Interface . . . 6–3
Document Revision History . . . 6–4

Chapter 7. Functional Description—RLDRAM II Controller

Block Description . . . 7–1
Avalon-MM Slave Interface . . . 7–1
Write Data FIFO Buffer . . . 7–2
Command Issuing FSM . . . 7–2
Refresh Timer . . . 7–2
Timer Module . . . 7–2
AFI . . . 7–2
User-Controlled Features . . . 7–2
Error Detection Parity . . . 7–2
User-Controlled Refresh . . . 7–3
Avalon-MM and Memory Data Width . . . 7–3
Signal Description . . . 7–3
Avalon-MM Slave Interface . . . 7–3
Document Revision History . . . 7–4

Chapter 8. Functional Description—RLDRAM 3 PHY-Only IP

Block Description . . . 8–1
Features . . . 8–1
RLDRAM III AFI Protocol . . . 8–2
Document Revision History . . . 8–3

Chapter 9. Functional Description—Example Designs

Synthesis Example Design . . . 9–1
Simulation Example Design . . . 9–3
Traffic Generator and BIST Engine . . . 9–5
Read and Write Generation . . . 9–6
Individual Read and Write Generation . . . 9–6
Block Read and Write Generation . . . 9–6
Address and Burst Length Generation . . . 9–6
Sequential Addressing . . . 9–6
Random Addressing . . . 9–6
Sequential and Random Interleaved Addressing . . . 9–7
Traffic Generator Signals . . . 9–7
Traffic Generator Add-Ons . . . 9–7
User Refresh Generator . . . 9–7
Traffic Generator Timeout Counter . . . 9–8
Creating and Connecting the UniPHY Memory Interface and the Traffic Generator in Qsys . . . 9–8
Creating the Qsys System . . . 9–8
Notes on Configuring UniPHY IP in Qsys . . . 9–9
Document Revision History . . . 9–10

Section II. UniPHY Reference

Chapter 10. Introduction to UniPHY IP

Release Information . . . 10–2
Device Family Support . . . 10–2
Features . . . 10–3
Unsupported Features . . . 10–5
System Requirements . . . 10–5
MegaCore Verification . . . 10–5
Resource Utilization . . . 10–5
DDR2, DDR3, and LPDDR2 SDRAM Controllers with UniPHY . . . 10–5
QDR II and QDR II+ SRAM Controllers with UniPHY . . . 10–12
RLDRAM II Controller with UniPHY . . . 10–13
Document Revision History . . . 10–14

Chapter 11. Latency for UniPHY IP

DDR2, DDR3, and LPDDR2 . . . 11–2
QDR II and QDR II+ . . . 11–3
RLDRAM II . . . 11–4
RLDRAM 3 . . . 11–4
Variable Controller Latency . . . 11–4
Document Revision History . . . 11–5

Chapter 12. Timing Diagrams for UniPHY IP

DDR2 and DDR3 Timing Diagrams . . . 12–1
QDR II and QDR II+ Timing Diagrams . . . 12–10
RLDRAM II Timing Diagrams . . . 12–15
LPDDR2 Timing Diagrams . . . 12–19
RLDRAM 3 Timing Diagrams . . . 12–23
Document Revision History . . . 12–27

Chapter 13. UniPHY External Memory Interface Debug Toolkit

Introduction . . . 13–1
Architecture . . . 13–1
Communication . . . 13–1
Calibration and Report Generation . . . 13–2
Setup and Use . . . 13–2
General Workflow . . . 13–3
Linking the Project to a Device . . . 13–3
Establishing Communication to Connections . . . 13–4
Reports . . . 13–4
Summary Report . . . 13–4
Calibration Report . . . 13–5
Margin Report . . . 13–5
Operational Considerations . . . 13–5
Specifying a Particular JDI File . . . 13–6
PLL Status . . . 13–6
Margining Reports . . . 13–6
Group Masks . . . 13–6
Troubleshooting . . . 13–6
EMIF On-Chip Debug Toolkit . . . 13–8
Introduction . . . 13–8
Access Protocol . . . 13–8

Command Codes Reference 13–10
Header Files 13–10
Generating UniPHY IP With the Debug Port 13–10
Example C Code for Accessing Debug Data 13–11
Document Revision History 13–14

Chapter 14. Upgrading to UniPHY-based Controllers from ALTMEMPHY-based Controllers
Upgrading from DDR2 or DDR3 SDRAM High-Performance Controller II with ALTMEMPHY Designs 14–1
Generating Equivalent Design 14–1
Replacing the ALTMEMPHY Datapath with UniPHY Datapath 14–2
Resolving Port Name Differences 14–2
Creating OCT Signals 14–4
Running Pin Assignments Script 14–4
Removing Obsolete Files 14–4
Simulating your Design 14–4
Document Revision History 14–6

Section III. ALTMEMPHY Reference

Chapter 15. Introduction to ALTMEMPHY IP
Release Information 15–2
Device Family Support 15–3
Features 15–4
ALTMEMPHY Megafunction 15–4
High-Performance Controller II 15–4
Unsupported Features 15–6
MegaCore Verification 15–6
Resource Utilization 15–6
System Requirements 15–8
Installation and Licensing 15–9
Free Evaluation 15–9
OpenCore Plus Time-Out Behavior 15–10
Document Revision History 15–10

Chapter 16. Latency for ALTMEMPHY IP
Latency Stages 16–2
Document Revision History 16–5

Chapter 17. Timing Diagrams for ALTMEMPHY IP
DDR and DDR2 High-Performance Controllers II 17–1
Half-Rate Read 17–2
Half-Rate Write 17–4
Full-Rate Read 17–6
Full-Rate Write 17–8
DDR3 High-Performance Controller II 17–9
Half-Rate Read (Burst-Aligned Address) 17–10
Half-Rate Write (Burst-Aligned Address) 17–12
Half-Rate Read (Non Burst-Aligned Address) 17–14
Half-Rate Write (Non Burst-Aligned Address) 17–16
Half-Rate Read With Gaps 17–18
Half-Rate Write With Gaps 17–19
Half-Rate Write Operation (Merging Writes) 17–20

Write-Read-Write-Read Operation 17–22
Document Revision History 17–24

Chapter 18. ALTMEMPHY External Memory Interface Debug Toolkit

Debug Toolkit Overview 18–1
Install the Debug Toolkit 18–2
Modify the Example Top-Level File to use the Debug Toolkit 18–2
Verify the Design 18–3
Regenerate the IP 18–4
Instantiate the JTAG Avalon-MM port in to the Example-Top Level Project 18–4
Add Additional Signals 18–5
Add alt_jtagavalon.v to your Quartus II Project Settings Files List 18–7
Recompile your Quartus II Test Design 18–7
Program Hardware with Debug Enabled .sof 18–7
Use the Debug Toolkit 18–8
Interpret the Results 18–9
Calibration Successful 18–9
Calibration Fails 18–13
Save the Calibration Results 18–13
Understand the Checksum and Failure Code 18–15
Document Revision History 18–16

Chapter Revision Dates

The chapters in this document, External Memory Interface Handbook, Volume 3: Reference Material, were revised on the following dates. Where chapters or groups of chapters are available separately, part numbers are listed.

Chapter 1. Functional Description—UniPHY
Revised: *November 2012*
Part Number: *EMI_RM_001-3.1*

Chapter 2. Functional Description—ALTMEMPHY
Revised: *November 2012*
Part Number: *EMI_RM_002-3.3*

Chapter 3. Functional Description—Hard Memory Interface
Revised: *November 2012*
Part Number: *EMI_RM_003-2.1*

Chapter 4. Functional Description—HPS Memory Controller
Revised: *November 2012*
Part Number: *EMI_RM_017-1.0*

Chapter 5. Functional Description—HPC II Controller
Revised: *November 2012*
Part Number: *EMI_RM_004-2.1*

Chapter 6. Functional Description—QDR II Controller
Revised: *November 2012*
Part Number: *EMI_RM_005-3.3*

Chapter 7. Functional Description—RLDRAM II Controller
Revised: *November 2012*
Part Number: *EMI_RM_006-3.3*

Chapter 8. Functional Description—RLDRAM 3 PHY-Only IP
Revised: *November 2012*
Part Number: *EMI_RM_018-1.0*

Chapter 9. Functional Description—Example Designs
Revised: *November 2012*
Part Number: *EMI_RM_007-1.3*

Chapter 10. Introduction to UniPHY IP
Revised: *November 2012*
Part Number: *EMI_RM_008-2.1*

Chapter 11. Latency for UniPHY IP
Revised: *November 2012*
Part Number: *EMI_RM_009-2.1*

Chapter 12. Timing Diagrams for UniPHY IP
Revised: *November 2012*
Part Number: *EMI_RM_010-2.1*

Chapter 13. UniPHY External Memory Interface Debug Toolkit
Revised: *November 2012*
Part Number: *EMI_RM_011-2.2*

Chapter 14. Upgrading to UniPHY-based Controllers from ALTMEMPHY-based Controllers
Revised: *November 2012*
Part Number: *EMI_RM_012-2.3*

Chapter 15. Introduction to ALTMEMPHY IP
Revised: *November 2012*
Part Number: *EMI_RM_013-1.2*

Chapter 16. Latency for ALTMEMPHY IP
Revised: *November 2012*
Part Number: *EMI_RM014-1.2*

Chapter 17. Timing Diagrams for ALTMEMPHY IP
Revised: *November 2012*
Part Number: *EMI_RM_015-1.3*

Chapter 18. ALTMEMPHY External Memory Interface Debug Toolkit
Revised: *November 2012*
Part Number: *EMI_RM_016-1.2*

Section I. Functional Descriptions

This section provides functional descriptions of the major external memory interface components.

This section includes the following chapters:

- Chapter 1, Functional Description—UniPHY
- Chapter 2, Functional Description—ALTMEMPHY
- Chapter 3, Functional Description—Hard Memory Interface
- Chapter 4, Functional Description—HPS Memory Controller
- Chapter 5, Functional Description—HPC II Controller
- Chapter 6, Functional Description—QDR II Controller
- Chapter 7, Functional Description—RLDRAM II Controller
- Chapter 8, Functional Description—RLDRAM 3 PHY-Only IP
- Chapter 9, Functional Description—Example Designs

For information about the revision history for chapters in this section, refer to "Document Revision History" in each individual chapter.

1. Functional Description—UniPHY

EMI_RM_001-3.1

This chapter describes the UniPHY layer of the external memory interface, and includes related information.

The major sections of this chapter are as follows:

- Block Description
- Interfaces
- UniPHY Signals
- PHY-to-Controller Interfaces
- Using a Custom Controller
- Using a Vendor-Specific Memory Model
- AFI 3.0 Specification
- Register Maps
- Ping Pong PHY
- Efficiency Monitor and Protocol Checker
- UniPHY Calibration Stages

Block Description

This section describes the major functional units of the UniPHY layer, which include the following:

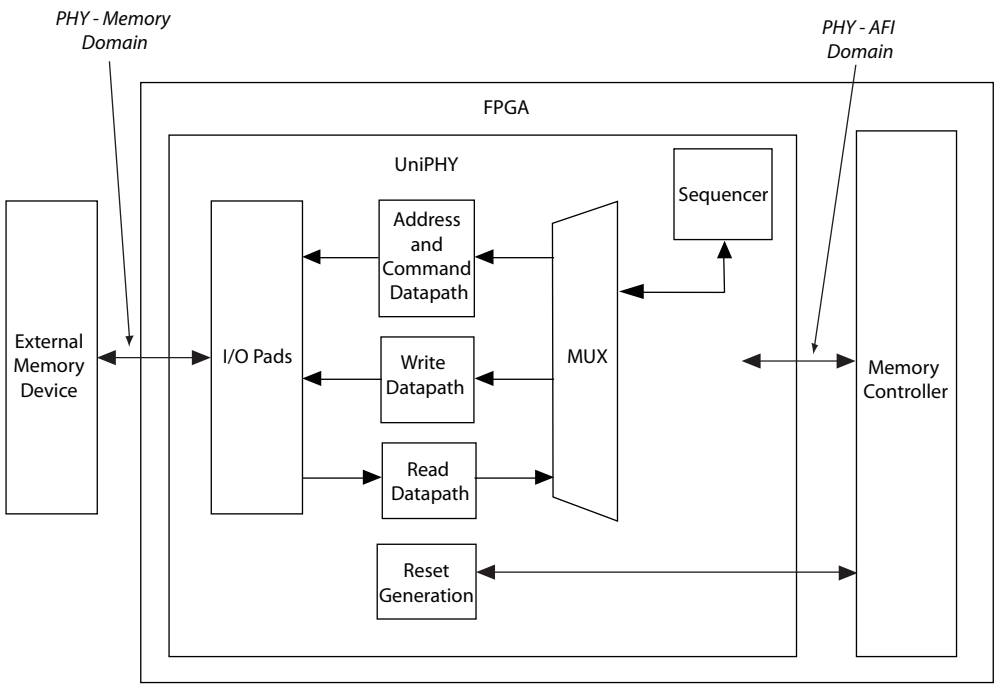
- Reset and Clock Generation
- Address and Command Datapath
- Write Datapath
- Read Datapath
- Sequencer

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure 1–1 shows the PHY block diagram; individual blocks are discussed in the text.

Figure 1–1. PHY Block Diagram



I/O Pads

The I/O pads contain all the I/O instantiations.

Reset and Clock Generation

At a high level, clocks in the PHY can be classified into two domains: the PHY-memory domain and the PHY-AFI domain. The PHY-memory domain interfaces with the external memory device and always operate at full-rate. The PHY-AFI domain interfaces with the memory controller and can be a full-rate, half-rate, or quarter-rate clock, based on the controller in use.

The number of clock domains in a memory interface can vary depending on its configuration; for example:

- At the PHY-memory boundary, separate clocks may exist to generate the memory clock signal, the output strobe, and to output write data, as well as address and command signals. These clocks include `pll_dq_write_clk`, `pll_write_clk`, `pll_mem_clk`, and `pll_addr_cmd_clk`. These clocks are phase-shifted as required to achieve the desired timing relationships between memory clock, address and command signals, output data, and output strobe.
- For quarter-rate interfaces, additional clock domains such as `pll_hr_clock` are required to convert signals between half-rate and quarter-rate.

- For high-performance memory interfaces using Arria V, Cyclone V, or Stratix V devices, additional clocks may be required to handle transfers between the device core and the I/O periphery for timing closure. For core-to-periphery transfers, the latch clock is `pll_c2p_write_clock`; for periphery-to-core transfers, it is `pll_p2c_read_clock`. These clocks are automatically phase-adjusted for timing closure during IP generation, but can be further adjusted in the parameter editor. If the phases of these clocks are zero, the Fitter may remove these clocks during optimization.

  Also, high-performance interfaces using a Nios II-based sequencer require two additional clocks, `pll_avl_clock` for the Nios II processor, and `pll_config_clock` for clocking the I/O scan chains during calibration.

For a complete list of clocks in your memory interface, compile your design and run the **Report Clocks** command in the TimeQuest Timing Analyzer.

Dedicated Clock Networks

The UniPHY layer employs three types of dedicated clock networks:

- Global clock network
- Dual-regional clock network
- PHY clock network (applicable to Arria V, Cyclone V, and Stratix V devices, and later)

The PHY clock network is a dedicated high-speed, low-skew, balanced clock tree designed for high-performance external memory interface. For device families that support the PHY clock network, UniPHY always uses the PHY clock network for all clocks at the PHY-memory boundary.

For families that do not support the PHY clock network, UniPHY uses either dual-regional or global clock networks for clocks at the PHY-memory boundary. During generation, the system selects dual-regional or global clocks automatically, depending on whether a given interface spans more than one quadrant. UniPHY does not mix the usage of dual-regional and global clock networks for clocks at the PHY-memory boundary; this ensures that timing characteristics of the various output paths are as similar as possible.

The <*variation_name*>**_pin_assignments.tcl** script creates the appropriate clock network type assignment. The use of the PHY clock network is specified directly in the RTL code, and does not require an assignment.

The UniPHY uses an active-low, asychronous assert and synchronous de-assert reset scheme. The global reset signal resets the PLL in the PHY and the rest of the system is held in reset until after the PLL is locked.

Address and Command Datapath

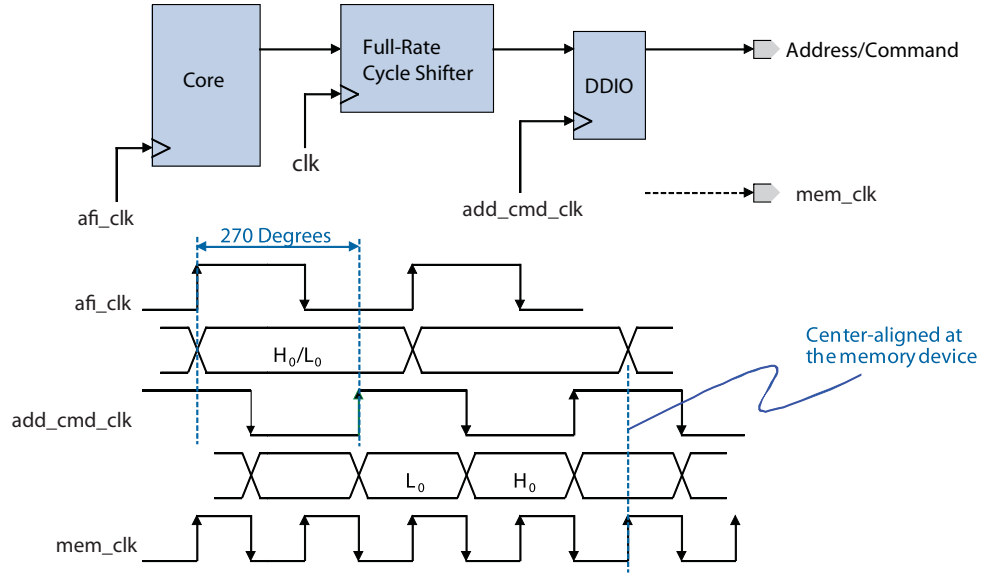
The memory controller controls the read and write addresses and commands to meet the memory specifications. The PHY is indifferent to address or command—that is, it performs no decoding or other operations—and the circuitry is the same for both. In full-rate and half-rate interfaces, address and command is full rate, while in quarter-rate interfaces, address and command is half rate.

Address and command signals are generated in the Altera PHY interface (AFI) clock domain and sent to the memory device in the address and command clock domain. The double-data rate input/output (DDIO) stage converts the half-rate signals into full-rate signals, when the AFI clock runs at half-rate. For quarter-rate interfaces, additional DDIO stages exist to convert the address and command signals in the quarter-rate AFI clock domain to half-rate.

The address and command clock is offset with respect to the memory clock to balance the nominal setup and hold margins at the memory device (center-alignment). In the example of Figure 1–2, this offset is 270 degrees. The Fitter can further optimize margins based on the actual delays and clock skews. In half-rate and quarter-rate designs, the full-rate cycle shifter blocks can perform a shift measured in full-rate cycles to implement the correct write latency; without this logic, the controller would only be able to implement even write latencies as it operates at half the speed. The full-rate cycle shifter is clocked by either the AFI clock or the address and command clock, depending on the PHY configuration, to maximize timing margins on the path from the AFI clock to the address and command clock.

Figure 1–2 illustrates the address and command datapath.

Figure 1–2. Address and Command Datapath (Half-rate example shown)

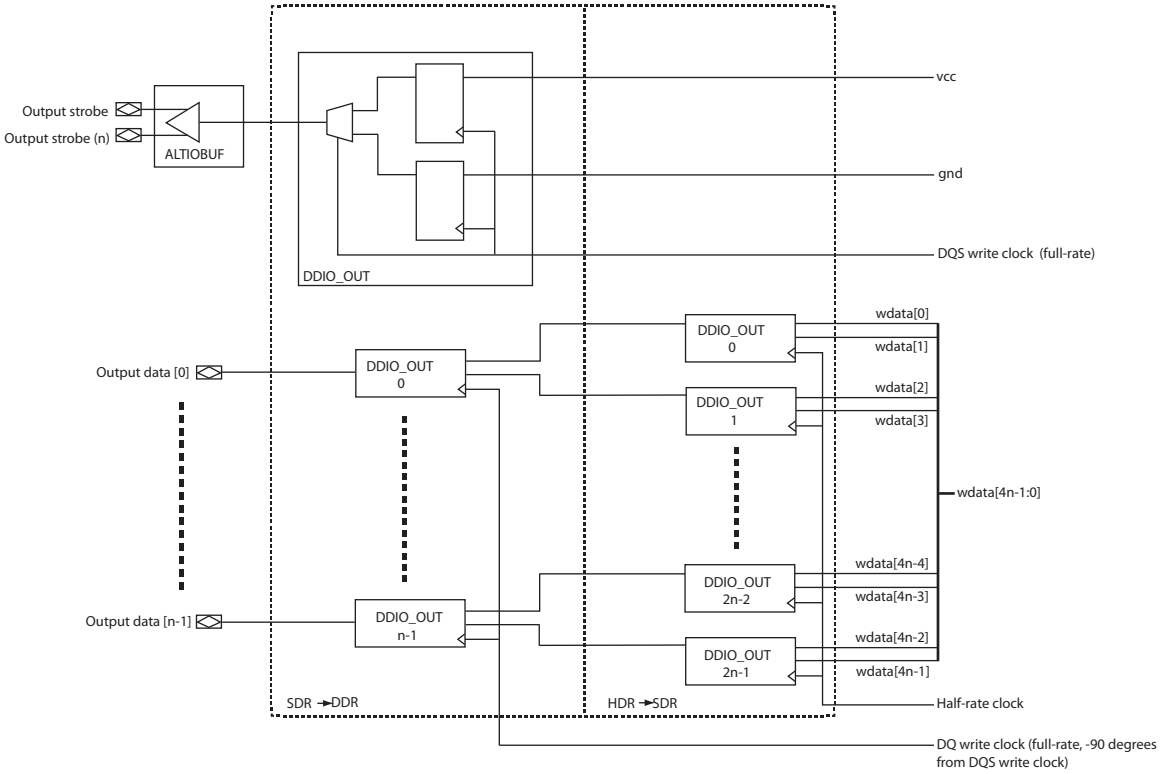


Write Datapath

The write datapath passes write data from the memory controller to the I/O. The write data valid signal from the memory controller generates the output enable signal to control the output buffer. For memory protocols with a bidirectional data bus, it also generates the dynamic termination control signal, which selects between series (output mode) and parallel (input mode) termination.

Figure 1–3 illustrates a simplified write datapath of a typical half-rate interface. The full-rate DQS write clock is sent to a DDIO_OUT cell. The output of DDIO_OUT feeds an output buffer which creates a pair of pseudo differential clocks that connects to the memory. In full-rate mode, only the SDR-DDR portion of the path is used; in half-rate mode, the HDR-SDR circuitry is also required. The use of DDIO_OUT in both the output strobe and output data generation path ensures that their timing characteristics are as similar as possible. The <*variation_name*>**_pin_assignments.tcl** script automatically specifies the logic option that associates all data pins to the output strobe pin. The Fitter treats the pins as a DQS/DQ pin group.

Figure 1–3. Write Datapath



Leveling Circuitry

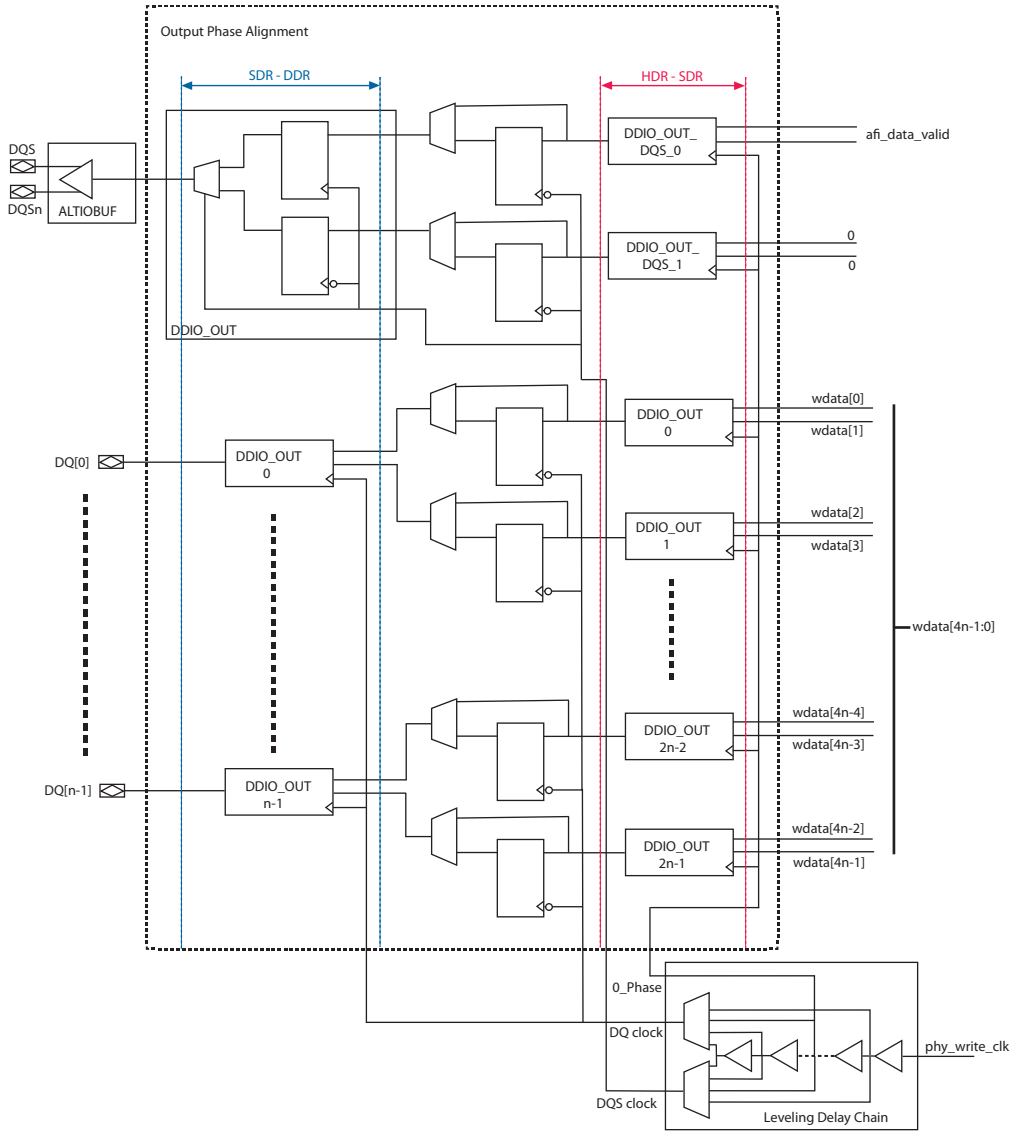
Leveling circuitry is dedicated I/O circuitry to provide calibration support for fly-by address and command networks. For DDR3, leveling is always invoked, whether the interface targets a DIMM or a single component. For DDR3 implementations at higher frequencies, a fly-by topology is recommended for optimal performance. For DDR2, leveling circuitry is invoked automatically for frequencies above 240 MHz; no leveling is used for frequencies below 240 MHz.

For DDR2 at frequencies below 240 MHz, you should use a tree-style layout. For frequencies above 240 MHz, you can choose either a leveled or balanced-T or Y topology, as the leveled PHY calibrates to either implementation. Regardless of protocol, for devices without a levelling block—such as Arria II GZ, Arria V, and Cyclone V—a balanced-T PCB topology for address/command/clock must be used because fly-by topology is not supported.

For details about leveling delay chains, consult the memory interfaces hardware section of the device handbook for your FPGA.

Figure 1–4 shows the write datapath for a leveling interface. The full-rate PLL output clock `phy_write_clk` goes to a leveling delay chain block which generates all other periphery clocks that are needed. The data signals that generate DQ and DQS signals pass to an output phase alignment block. The output phase alignment block feeds an output buffer which creates a pair of pseudo differential clocks that connect to the memory. In full-rate designs, only the SDR-DDR portion of the path is used; in half-rate mode, the HDR-SDR circuitry is also required. The use of DDIO_OUT in both the output strobe and output data generation paths ensures that their timing characteristics are as similar as possible. The *<variation_name>*_**pin_assignments.tcl** script automatically specifies the logic option that associates all data pins to the output strobe pin. The Quartus II Fitter treats the pins as a DQS/DQ pin group.

Figure 1–4. Write Datapath for a Leveling Interface



Read Datapath

Figure 1–5 shows the read datapath. This section describes the blocks and flow in the read datapath.

For all protocols, the DQS logic block delays the strobe by 90 degrees to center-align the rising strobe edge within the data window. For DDR2, DDR3, and LPDDR2 protocols, the logic block also performs strobe gating, holding the DQS enable signal high for the entire period that data is received. One DQS logic block exists for each data group.

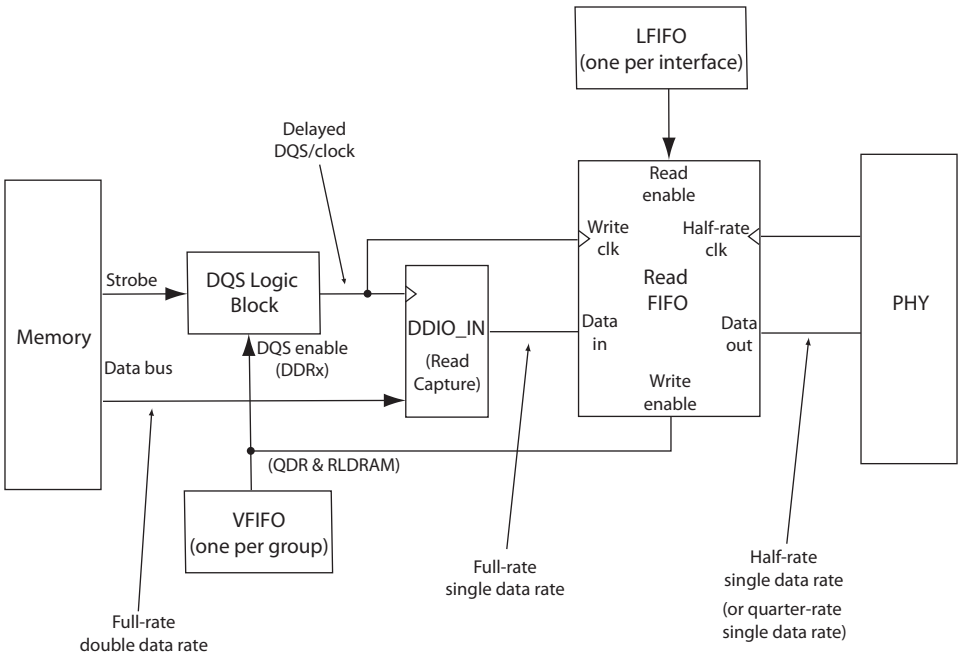
One VFIFO buffer exists for each data group. For DDR2, DDR3, and LPDDR2 protocols, the VFIFO buffer generates the DQS enable signal, which is delayed (by an amount determined during calibration) to align with the incoming DQS signal. For QDR and RLDRAM protocols, the output of the VFIFO buffer serves as the write enable signal for the Read FIFO buffer, signaling when to begin capturing data.

DDIO_IN receives data from memory at double-data rate and passes data on to the Read FIFO buffer at single-data rate.

The Read FIFO buffer temporarily holds data read from memory; one Read FIFO buffer exists for each data group. For half-rate interfaces, the Read FIFO buffer converts the full-rate, single data-rate input to a half-rate, single data-rate output which is then passed to the PHY core logic. In the case of a quarter-rate interface, soft logic in the PHY performs an additional conversion from half-rate single data rate to quarter-rate single data rate.

One LFIFO buffer exists for each memory interface; the LFIFO buffer generates the read enable signal for all Read FIFO blocks in an interface. The read enable signal is asserted when the Read FIFO blocks have buffered sufficient data from the memory to be read. The timing of the read enable signal is determined during calibration.

Figure 1–5. Read Datapath



Sequencer

Depending on the combination of protocol and IP architecture in your external memory interface, you may have either an RTL-based sequencer or a Nios® II-based sequencer. This section discusses the Nios II-based sequencer and the RTL-based sequencer.

Table 1–3 in Volume 1 of this handbook shows the sequencer support for different protocol-architecture combinations.

Be aware that RTL-based sequencer implementations and Nios II-based sequencer implementations can have different pin requirements. You may not be able to migrate from an RTL-based sequencer to a Nios II-based sequencer and maintain the same pinout.

For information on pin planning, refer to Planning Pin and FPGA Resources in volume 2 of the *External Memory Interface Handbook*.

Nios II-Based Sequencer

The DDR2, DDR3, and LPDDR2 controllers with UniPHY employ a Nios II-based sequencer that is parameterizable and is dynamically generated at run time. The Nios II-based sequencer is also available with the QDR II and RLDRAM II controllers.

Function

The sequencer enables high-frequency memory interface operation by calibrating the interface to compensate for variations in setup and hold requirements caused by transmission delays.

The UniPHY converts the double-data rate interface of high-speed memory devices to a full-rate or half-rate interface for use within an FPGA. To compensate for slight variations in data transmission to and from the memory device, double-data rate is usually center-aligned with its strobe signal; nonetheless, at high speeds, slight variations in delay can result in setup or hold time violations. The sequencer implements a calibration algorithm to determine the combination of delay and phase settings necessary to maintain center-alignment of data and clock signals, even in the presence of significant delay variations. Programmable delay chains in the FPGA I/Os then implement the calculated delays to ensure that data remains centered. Calibration also applies settings to the FIFO buffers within the PHY to minimize latency and ensures that the read valid signal is generated at the appropriate time.

When calibration is completed, the sequencer returns control to the memory controller.

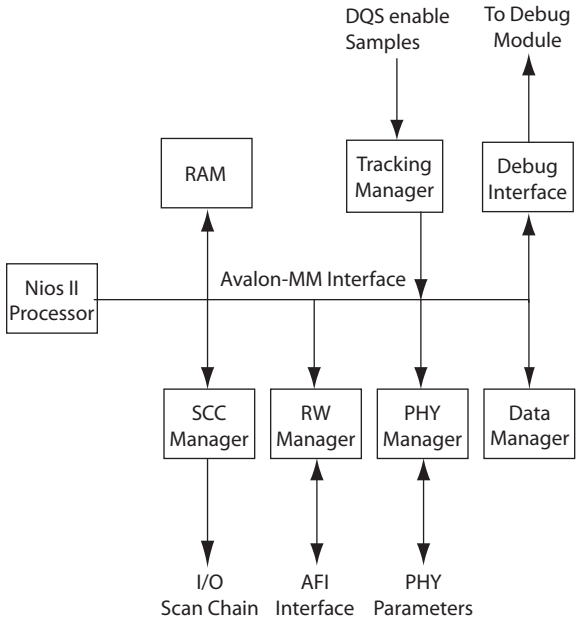
For more information about calibration, refer to UniPHY Calibration Stages.

Architecture

Figure 1–6 shows the sequencer block diagram. The sequencer is composed of a Nios II processor and a series of hardware-based component managers, connected together by an Avalon bus. The Nios II processor performs the high-level algorithmic operations of calibration, while the component managers handle the lower-level timing, memory protocol, and bit-manipulation operations.

The high-level calibration algorithms are specified in C code, which is compiled into Nios II code that resides in the FPGA RAM blocks. The debug interface provides a mechanism for interacting with the various managers and for tracking the progress of the calibration algorithm, and can be useful for debugging problems that arise within the PHY. The various managers are specified in RTL and implement operations that would be slow or inefficient if implemented in software.

Figure 1–6. Sequencer Block Diagram



The C code that defines the calibration routines is available for your reference in the **\<*name*>_s0_software** subdirectory. Altera does not recommend that you modify this C code.

SCC Manager

The scan chain control (SCC) manager allows the sequencer to set various delays and phases on the I/Os that make up the memory interface. The latest Altera device families provide dynamic delay chains on input, output, and output enable paths which can be reconfigured at runtime. The SCC manager provides the calibration routines access to these chains to add delay on incoming and outgoing signals. A master on the Avalon-MM interface may require the maximum allowed delay setting on input and output paths, and may set a particular delay value in this range to apply to the paths.

The SCC manager implements the Avalon-MM interface and the storage mechanism for all input, output, and phase settings. It contains circuitry that configures a DQ- or DQS-configuration block. The Nios II processor may set delay, phases, or register settings; the sequencer scans the settings serially to the appropriate DQ or DQS configuration block.

RW Manager

The read write (RW) manager encapsulates the protocol to read and write to the memory device through the Altera PHY Interface (AFI). It provides a buffer that stores the data to be sent to and read from memory, and provides the following commands:

- Write configuration—configures the memory for use. Sets up burst lengths, read and write latencies, and other device specific parameters.
- Refresh—initiates a refresh operation at the DRAM. The command does not exist on SRAM devices. The sequencer also provides a register that determines whether the RW manager automatically generates refresh signals.
- Enable or disable multi-purpose register (MPR)—for memory devices with a special register that contains calibration specific patterns that you can read, this command enables or disables access to the register.
- Activate row—for memory devices that have both rows and columns, this command activates a specific row. Subsequent reads and writes operate on this specific row.
- Precharge—closes a row before you can access a new row.
- Write or read burst—writes or reads a burst length of data.
- Write guaranteed—writes with a special mode where the memory holds address and data lines constant. Altera guarantees this type of write to work in the presence of skew, but constrains to write the same data across the entire burst length.
- Write and read back-to-back—performs back-to-back writes or reads to adjacent banks. Most memory devices have strict timing constraints on subsequent accesses to the same bank, thus back-to-back writes and reads have to reference different banks.
- Protocol-specific initialization—a protocol-specific command required by the initialization sequence.

PHY Manager

The PHY Manager provides access to the PHY for calibration, and passes relevant calibration results to the PHY. For example, the PHY Manager sets the VFIFO and LFIFO buffer parameters resulting from calibration, signals the PHY when the memory initialization sequence finishes, and reports the pass/fail status of calibration.

Data Manager

The Data Manager stores parameterization-specific data in RAM, for the software to query.

Tracking Manager

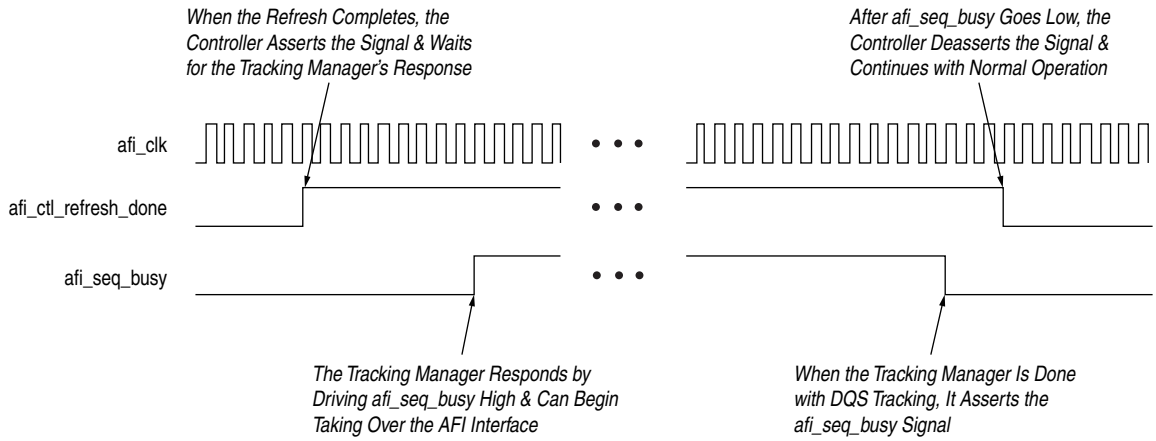
The Tracking Manager detects the effects of voltage and temperature variations that can occur on the memory device over time resulting in reduced margins, and adjusts the DQS enable delay as necessary to maintain adequate operating margins.

The Tracking Manager briefly assumes control of the AFI interface after each memory refresh cycle, issuing a read routine to the RW Manager, and then sampling the DQS tracking. Ideally, the falling edge of the DQS enable signal would align to the last rising edge of the raw DQS signal from the memory device. The Tracking Manager determines whether the DQS enable signal is leading or trailing the raw DQS signal.

Each time a refresh occurs, the Tracking Manager takes a sample of the raw DQS signal; any adjustments of the DQS enable signal occur only after sufficient samples of raw DQS have been taken. When the Tracking Manager determines that the DQS enable signal is either leading or lagging the raw DQS signal, it adjusts the DQS enable appropriately.

Figure 1–7 shows the Tracking manager signals.

Figure 1–7. Tracking Manager Signals



Some notes on Tracking Manager operation:

- The time taken by the Tracking Manager is arbitrary; if the period taken exceeds the refresh period, the Tracking Manager handles memory refresh.
- `afi_seq_busy` should go high fewer than 10 clock cycles after `afi_ctl_refresh_done` or `afi_ctl_long_idle` is asserted.
- `afi_refresh_done` should deassert fewer than 10 clock cycles after `afi_seq_busy` deasserts.
- `afi_ctl_long_idle` causes the Tracking Manager to execute an algorithm different than periodic refresh; use `afi_ctl_long_idle` when a long session has elapsed without a periodic refresh.

- The Tracking Manager is instantiated into the sequencer system when DQS Enable Tracking is turned on. Table 1–1 summarizes configurations supporting DQS tracking.

Table 1–1. Configurations Supporting DQS Tracking

Device Family	Protocol	Frequency
Arria V Arria V GZ Cyclone V Stratix V	LPDDR2	All frequencies.
	DDR3 (single rank)	More than 450 MHz for speed grade 5, or more than 534 MHz.

- If you do not want to use DQS tracking, you can disable it (at your risk), by opening the Verilog file `<variant_name>_if0_c0.v` in an editor, and changing the value of the `USE_DQS_TRACKING` parameter from 1 to 0.

Nios II Processor

The Nios II processor manages the calibration algorithm; the Nios II processor is unavailable after calibration is completed.

The same calibration algorithm supports all device families, with some differences. The following sections describe the calibration algorithm for DDR3 SDRAM on Stratix III devices. Calibration algorithms for other protocols and families are a subset and significant differences are pointed out when necessary. As the algorithm is fully contained in the software of the sequencer (in the C code) enabling and disabling specific steps involves turning flags on and off.

Calibration consists of the following stages:

- Initialize memory.
- Calibrate read datapath.
- Calibrate write datapath.
- Run diagnostics.

Initialize Memory

Calibration must initialize all memory devices before they can operate properly. The sequencer performs this memory initialization stage when it takes control of the PHY at startup.

Calibrate Read Datapath

Calibrating the read datapath comprises the following steps:

- Calibrate DQS enable cycle and phase.
- Perform read per-bit deskew to center the strobe signal within data valid window.
- Reduce LFIFO latency.

Calibrate Write Datapath

Calibrating the write datapath involves the following steps:

- Center align DQS with respect to DQ.
- Align DQS with `mem_clk`.

Test Diagnostics

The sequencer estimates the read and write margins under noisy conditions, by sweeping input and output DQ and DQS delays to determine the size of the data valid windows on the input and output sides. The sequencer stores this information in the local memory and you can access it through the debugging interface.

When the diagnostic test finishes, control of the PHY interface passes back to the controller and the sequencer issues a pass or fail signal.

RTL-based Sequencer

The RTL-based sequencer is available for QDR II and RLDRAM II interfaces; it is a state machine that processes the calibration algorithm. The sequencer assumes control of the interface at reset (whether at initial startup or when the IP is reset) and maintains control throughout the calibration process. The sequencer relinquishes control to the memory controller only after successful calibration. Table 1–2 lists the major states in the RTL-based sequencer.

Table 1–2. Sequencer States (Part 1 of 2)

State	Description
RESET	Remain in this state until reset is released.
LOAD_INIT	Load any initialization values for simulation purposes.
STABLE	Wait until the memory device is stable.
WRITE_ZERO	Issue write command to address 0.
WAIT_WRITE_ZERO	Write all 0xAs to address 0.
WRITE_ONE	Issue write command to address 1.
WAIT_WRITE_ONE	Write all 0x5s to address 1.
Valid Calibration States	
V_READ_ZERO	Issue read command to address 0 (expected data is all 0xAs).
V_READ_NOP	This state represents the minimum number of cycles required between 2 back-to-back read commands. The number of NOP states depends on the burst length.
V_READ_ONE	Issue read command to address 1 (expected data is all 0x5s).
V_WAIT_READ	Wait for read valid signal.
V_COMPARE_READ_ZERO_READ_ONE	Parameterizable number of cycles to wait before making the read data comparisons.
V_CHECK_READ_FAIL	When a read fails, the write pointer (in the AFI clock domain) of the valid FIFO buffer is incremented. The read pointer of the valid FIFO buffer is in the DQS clock domain. The gap between the read and write pointers is effectively the latency between the time when the PHY receives the read command and the time valid data is returned to the PHY.
V_ADD_FULL_RATE	Advance the read valid FIFO buffer write pointer by an extra full rate cycle.
V_ADD_HALF_RATE	Advance the read valid FIFO buffer write pointer by an extra half rate cycle. In full-rate designs, equivalent to V_ADD_FULL_RATE.
V_READ_FIFO_RESET	Reset the read and write pointers of the read data synchronization FIFO buffer.
V_CALIB_DONE	Valid calibration is successful.

Table 1–2. Sequencer States (Part 2 of 2)

State	Description
Latency Calibration States	
L_READ_ONE	Issue read command to address 1 (expected data is all 0x5s).
L_WAIT_READ	Wait for read valid signal from read datapath. Initial read latency is set to a predefined maximum value.
L_COMPARE_READ_ONE	Check returned read data against expected data. If data is correct, go to L_REDUCE_LATENCY; otherwise go to L_ADD_MARGIN.
L_REDUCE_LATENCY	Reduce the latency counter by 1.
L_READ_FLUSH	Read from address 0, to flush the contents of the read data resynchronization FIFO buffer.
L_WAIT_READ_FLUSH	Wait until the whole FIFO buffer is flushed, then go back to L_READ and try again.
L_ADD_MARGIN	Increment latency counter by 3 (1 cycle to get the correct data, 2 more cycles of margin for run time variations). If latency counter value is smaller than predefined ideal condition minimum, then go to CALIB_FAIL.
CALIB_DONE	Calibration is successful.
CALIB_FAIL	Calibration is not successful.

Shadow Registers

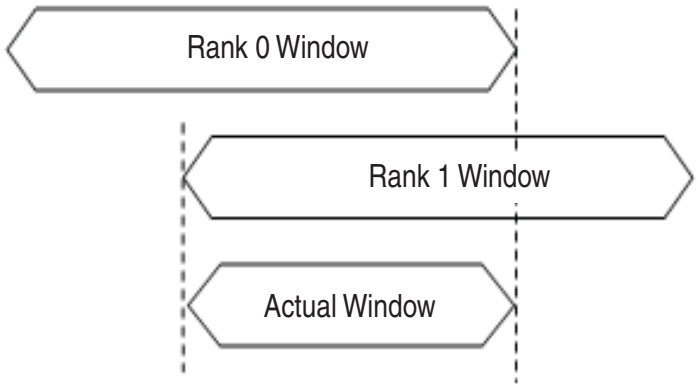
Shadow registers are a hardware feature of Arria V GZ and Stratix V devices that enables high-speed multi-rank calibration for DDR3 quarter-rate and half-rate memory interfaces, up to 667MHz for dual-rank interfaces and 533MHz for quad-rank interfaces.

Description

Prior to the introduction of shadow registers, the data valid window of a multi-rank interface was calibrated to the overlapping portion of the data valid windows of the individual ranks. The resulting data valid window for the interface would be smaller than the individual data valid windows, limiting overall performance.

Figure 1–8 illustrates the calibration of overlapping data valid windows, without shadow registers.

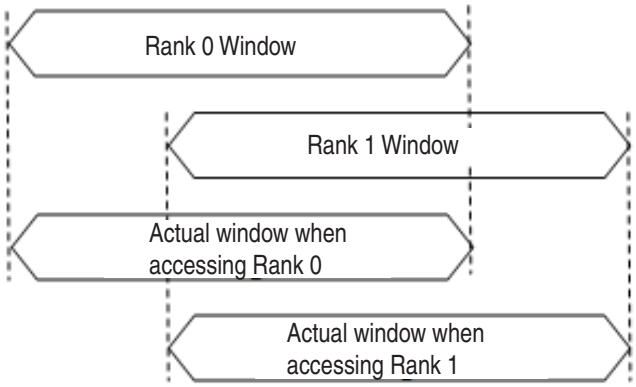
Figure 1–8. Calibration of Overlapping Data Valid Windows, without Shadow Registers



Shadow registers allow the sequencer to calibrate each rank separately and fully, and then to save the calibrated settings for each rank in its own set of shadow registers, which are part of the IP scan chains. During a rank-to-rank switch, the rank-specific set of calibration settings are restored just-in-time to optimize the data valid window for each rank.

Figure 1–9 illustrates how the use of rank-specific calibration settings results in a data valid window appropriate for the current rank.

Figure 1–9. Rank-Specific Calibration Settings, with Shadow Registers



The shadow registers and their associated rank-switching circuitry are part of the device I/O periphery hardware.

Operation

The sequencer calibrates each rank individually and stores the resulting configuration in shadow registers, which are part of the IP scan chains. UniPHY then selects the appropriate configuration for the rank in use, switching between configurations as necessary. Calibration results for deskew delay chains are stored in the shadow registers. For DQS enable/disable, delay chain configurations come directly from the FPGA core.

Arria V GZ and Stratix V devices provide two sets of shadow registers, allowing for full support of a dual-rank interface, and support of additional ranks at reduced frequency.

- For a 2-rank interface, each rank is fully calibrated separately.
- For a 4-rank interface, the first set of shadow registers stores calibration data for the first and second rank, and the second set of shadow registers stores calibration data for the third and fourth ranks.

DLL Offset Control Block

For designs generated with HardCopy compatibility enabled, the UniPHY layer includes DLL offset control blocks, which allow adjustment of the DLL control word for modifying the DQS delay chains to compensate for variations in silicon.

A DLL offset control block can feed only one side of the chip, therefore the IP instantiates two DLL offset control blocks for each DLL, to permit control of resources on two sides of the chip, as in the case of wraparound designs. In non-wraparound designs, where the second DLL offset control block is not needed, the second control block remains unused and disappears during synthesis.

One complication of the dual DLL offset control block process, is that at generation time it is impossible to know where resources are going to be placed, so the system automatically connects the output of the first DLL offset control block to all DQS groups. In the case of a wraparound design, you must modify the RTL code after pin placement, to connect the second DLL offset control block.

To connect the second DLL offset control block, follow these steps:

1. Open the file **<*variation_name*>_new_io_pads.v** in an editor.
2. Find a line of a form similar to the following:

   ```
   .dll_offsetdelay_in((i < 0) ?
   hc_dll_config_dll_offset_ctrl_offsetctrlout :
   hc_dll_config_dll_offset_ctrl_offsetctrlout),
   ```

   This line connects the output of the first DLL offset control block (`hc_dll_config_dll_offset_ctrl_offsetctrlout`) to the offset control input of the DQS delay chain, for every DQS delay chain where `i < 0`.
3. Modify the above line to connect the second DLL offset control block. The following example shows the correct syntax to connect groups 0 to 3 to the output of the first DLL offset control block, and groups 4 and above to the output of the second DLL offset control block:

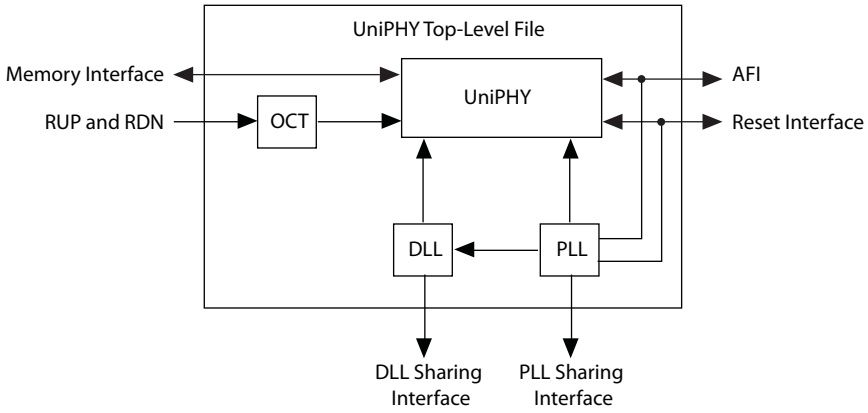
   ```
   .dll_offsetdelay_in((i < 4) ?
   hc_dll_config_dll_offset_ctrl_offsetctrlout :
   hc_dll_config_dll_offset_ctrl_b_offsetctrlout),
   ```

Interfaces

Figure 1–10 shows the major blocks of the UniPHY and how it interfaces with the external memory device and the controller.

Instantiating the delay-locked loop (DLL) and the phase-locked loop (PLL) on the same level as the UniPHY eases DLL and PLL sharing.

Figure 1–10. UniPHY Interfaces with the Controller and the External Memory



The following interfaces are on the UniPHY top-level file:

- AFI
- Memory interface
- DLL sharing interface
- PLL sharing interface
- OCT interface

AFI

The UniPHY datapath uses the Altera PHY interface (AFI). The AFI is in a simple connection between the PHY and controller. The AFI is based on the DDR PHY interface (DFI) specification, with some calibration-related signals not used and some additional Altera-specific sideband signals added.

For more information about the AFI, refer to AFI 3.0 Specification.

The Memory Interface

For more information on the memory interface, refer to "UniPHY Signals" on page 1–21.

The DLL and PLL Sharing Interface

You can generate the UniPHY memory interface and configure it to share its PLL, DLL, or both interfaces. By default, a UniPHY memory interface variant contains a PLL and DLL; the PLL produces a variety of required clock signals derived from the reference clock, and the DLL produces a delay codeword. In this case the PLL sharing mode is "No sharing". A UniPHY variant can be configured as a PLL Master and/or DLL Master, in which case the corresponding interfaces are exported to the UniPHY top-level and can be connected to an identically configured UniPHY variant PLL Slave and/or DLL Slave. The UniPHY slave variant is instantiated without a PLL and/or DLL, which saves device resources.

☞ For Arria II GX, Arria II GZ, Stratix III, and Stratix IV devices, the PLL and DLL must both be shared at the same time—their sharing modes must match. This restriction does not apply to Arria V, Arria V GZ, Cyclone V, or Stratix V devices.

☞ For devices with hard memory interface components onboard, you cannot share PLL or DLL resources between soft and hard interfaces.

To share PLLs or DLLs, follow these steps:

1. To create a PLL or DLL master, create a UniPHY memory interface IP core. To make the PLL and/or DLL interface appear at the top-level in the core, on the **PHY Settings** tab in the parameter editor, set the **PLL Sharing Mode** and/or **DLL Sharing Mode** to `Master`.
2. To create a PLL or DLL slave, create a second UniPHY memory interface IP core. To make the PLL and/or DLL interface appear at the top-level in the core, on the **PHY Settings** tab set the **PLL Sharing Mode** and/or **DLL Sharing Mode** to `Slave`.
3. Connect the PLL and/or DLL sharing interfaces by following the appropriate step, below:
   - **For cores generated with Megawizard Plug-in Manager**: connect the PLL and/or DLL interface ports between the master and slave cores in your wrapper RTL. When using PLL sharing, connect the `afi_clk`, `afi_half_clk`, and `afi_reset_n` outputs from the UniPHY PLL master to the `afi_clk`, `afi_half_clk`, and `afi_reset_in` inputs on the UniPHY PLL slave
   - **For cores generated with Qsys**, connect the PLL and/or DLL interface in the Qsys GUI. When using PLL sharing, connect the `afi_clk`, `afi_half_clk`, and afi_reset interfaces from the UniPHY PLL master to the `afi_clk_in`, `afi_half_clk_in`, and `afi_reset_in` interfaces on the UniPHY PLL slave.

     Qsys supports only one-to-one conduit connections in the patch panel. To share a PLL from a Uniphy PLL master with multiple slaves, you should replicate the number of PLL sharing conduit interfaces in the Qsys patch panel by choosing **Number of PLL sharing interfaces** in the parameter editor.

You may connect a slave UniPHY instance to the clocks from a user-defined PLL instead of from a UniPHY master. The general procedure for doing so is as follows:

a. Make a template, by generating your IP with **PLL Sharing Mode** set to `No Sharing`, and then compiling the example project to determine the frequency and phases of the clock outputs from the PLL.

b. Generate an external PLL using the MegaWizard flow, with the equivalent output clocks.

c. Generate your IP with **PLL Sharing Mode** set to `Slave`, and connect the external PLL to the PLL sharing interface.

You must be very careful when connecting clock signals to the slave. Connecting to clocks with frequency or phase different than what the core expects may result in hardware failure.

The signal `dll_pll_locked` is an internal signal from the PLL to the DLL which ensures that the DLL remains in reset mode until the PLL becomes locked. This signal is not available for use by customer logic.

The PLL and DLL sharing interfaces are available when using SOPC Builder, however the PLL and/or DLL interfaces cannot be connected using the SOPC Builder GUI interface. To complete master-slave connections when using SOPC Builder, you must edit the generated RTL code manually.

About PLL Simulation

PLL frequencies may differ between the synthesis and simulation file sets. In either case the achieved PLL frequencies and phases are calculated and reported in real time in the parameter editor.

For the simulation file set, clocks are specified in the RTL, not in units of frequency but by the period in picoseconds, thus avoiding clock drift due to picosecond rounding error.

For the synthesis file set, there are two mechanisms by which clock frequencies are specified in the RTL, based on the target device family:

- For Arria V, Arria V GZ, Cyclone V, and Stratix V, clock frequencies are specified in megahertz.
- For Arria II GX, Arria II GZ, Stratix III, and Stratix IV, clock frequencies are specified by integer multipliers and divisors. For these families, the real simulation model—as opposed to the default abstract simulation model—also uses clock frequencies specified by integer ratios.

The OCT Sharing Interface

By default, the UniPHY IP generates the required OCT control block at the top-level RTL file for the PHY. If you want, you can instantiate this block elsewhere in your code and feed the required termination control signals into the IP core by turning off **Master for OCT Control Block** on the **PHY Settings** tab. If you turn off **Master for OCT Control Block**, you must instantiate the OCT control block or use another UniPHY instance as a master, and ensure that the parallel and series termination control bus signals are connected to the PHY.

Figure 1–11 and Figure 1–12, respectively, show the PHY architecture with and without Master for OCT Control Block.

Figure 1–11. PHY Architecture with Master for OCT Control Block

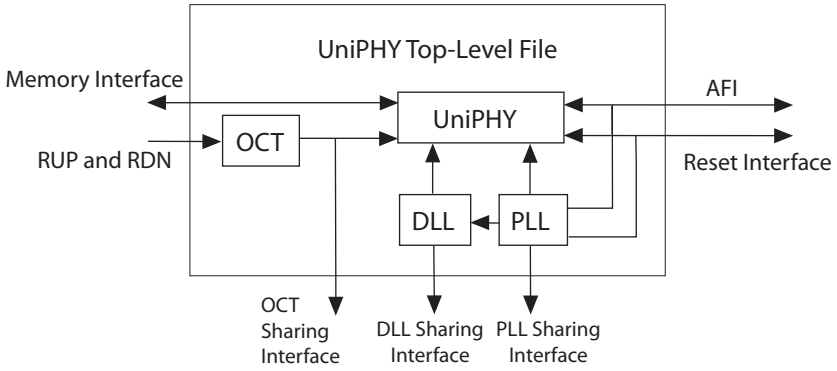
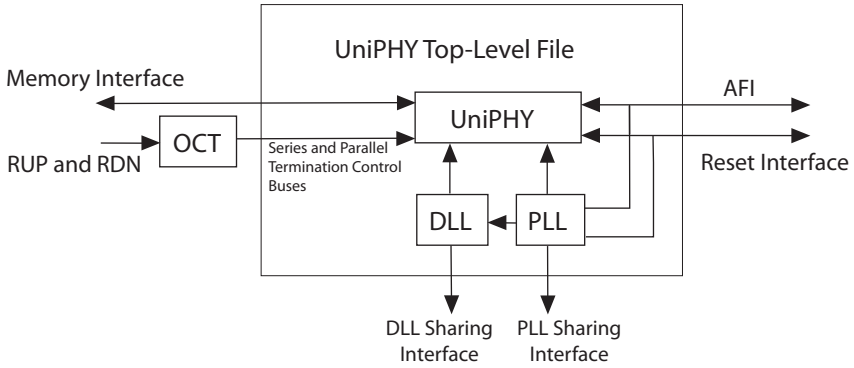


Figure 1–12. PHY Architecture without Master for OCT Control Block



The OCT sharing interface is available when using SOPC Builder, however the OCT sharing interface cannot be connected using the SOPC Builder parameter editor. To complete master-slave connections when using SOPC Builder, you must edit the generated RTL code manually.

If you generate a QDR II or RLDRAM II slave IP core, you must modify the pin assignment script to allow the fitter to correctly resolve the OCT termination block name in the OCT master core.

To modify the pin assignment script for QDR II or RLDRAM II slaves, follow these steps:

1. In a text editor, open your system's Tcl pin assignments script file, as follows:
   - For systems generated with the MegaWizard Plug-In Manager: Open the <*IP core name*>/<*slave core name*>**_p0_pin_assignments.tcl** file.
   - For systems generated with Qsys or SOPC Builder: Open the <*HDL Path*>/<*submodules*>/<*slave core name*>**_p0_pin_assignments.tcl** file.
2. Search for the following line:
   - `set ::master_corename "_MASTER_CORE_"`
3. Replace `_MASTER_CORE_` with the instance name of the UniPHY master to which the slave is connected. The instance name is determined from the pin assignments file name, as follows:
   - For systems generated with Qsys or SOPC Builder, the instance name is the <*master core name*> component of the pins assignments file name: <*HDL path*>/<*submodules*>/<*master core name*>**_p0_pin_assignments.tcl**
   - For systems generated with the MegaWizard Plug-in Manager, the instance name is the <*master core name*> component of the pins assignments file name: <*IP core name*>/<*master core name*>**_p0_pin_assignments.tcl**

UniPHY Signals

This section describes the UniPHY signals.

Table 1–3 lists the clock and reset signals.

Table 1–3. Clock and Reset Signals

Name	Direction	Width	Description
`pll_ref_clk`	Input	1	PLL reference clock input.
`global_reset_n`	Input	1	Active low global reset for PLL and all logic in the PHY, which causes a complete reset of the whole system. Minimum pulse width is 40ns.
`soft_reset_n`	Input	1	Holding `soft_reset_n` low holds the PHY in a reset state. However it does not reset the PLL, which keeps running. It also holds the `afi_reset_n` output low. Mainly for use by SOPC Builder.

Table 1–4 lists the DDR2 and DDR3 SDRAM interface signals.

Table 1–4. DDR2 and DDR3 SDRAM Interface Signals (Part 1 of 2)

Name	Direction	Width	Description
`mem_ck, mem_ck_n`	Output	`MEM_CK_WIDTH`	Memory clock.
`mem_cke`	Output	`MEM_CLK_EN_WIDTH`	Clock enable.
`mem_cs_n`	Output	`MEM_CHIP_SELECT_WIDTH`	Chip select..
`mem_cas_n`	Output	`MEM_CONTROL_WIDTH`	Column address strobe.

Table 1–4. DDR2 and DDR3 SDRAM Interface Signals (Part 2 of 2)

Name	Direction	Width	Description
mem_ras_n	Output	MEM_CONTROL_WIDTH	Row address strobe.
mem_we_n	Output	MEM_CONTROL_WIDTH	Write enable.
mem_a	Output	MEM_ADDRESS_WIDTH	Address.
mem_ba	Output	MEM_BANK_ADDRESS_WIDTH	Bank address.
mem_dqs, mem_dqs_n	Bidirectional	MEM_DQS_WIDTH	Data strobe.
mem_dq	Bidirectional	MEM_DQ_WIDTH	Data.
mem_dm	Output	MEM_DM_WIDTH	Data mask.
mem_odt	Output	MEM_ODT_WIDTH	On-die termination.
mem_reset_n (DDR3 only)	Output	1	Reset
mem_ac_parity (DDR3 only, RDIMM/LRDIMM only)	Output	MEM_CONTROL_WIDTH	Address/command parity bit. (Even parity, per the RDIMM spec, JESD82-29A.)
mem_err_out_n (DDR3 only, RDIMM/LRDIMM only)	Input	MEM_CONTROL_WIDTH	Address/command parity error.

For information about the AFI signals, refer to AFI 3.0 Specification.

For information about top-level HardCopy migration signals, refer to HardCopy Design Migration Guidelines in volume 2 of the *External Memory Interface Handbook*.

Table 1–5 lists parameters relating to UniPHY signals.

Table 1–5. UniPHY Parameters (Part 1 of 2)

Parameter Name	Description
AFI_RATIO	AFI_RATIO is 1 in full-rate designs. AFI_RATIO is 2 for half-rate designs. AFI_RATIO is 4 for quarter-rate designs.
MEM_IF_DQS_WIDTH	The number of DQS pins in the interface.
MEM_ADDRESS_WIDTH	The address width of the specified memory device.
MEM_BANK_WIDTH	The bank width of the specified memory device.
MEM_CHIP_SELECT_WIDTH	The chip select width of the specified memory device.
MEM_CONTROL_WIDTH	The control width of the specified memory device.
MEM_DM_WIDTH	The DM width of the specified memory device.
MEM_DQ_WIDTH	The DQ width of the specified memory device.
MEM_READ_DQS_WIDTH	The READ DQS width of the specified memory device.
MEM_WRITE_DQS_WIDTH	The WRITE DQS width of the specified memory device.
OCT_SERIES_TERM_CONTROL_WIDTH	—
OCT_PARALLEL_TERM_CONTROL_WIDTH	—
AFI_ADDRESS_WIDTH	The AFI address width, derived from the corresponding memory interface width.
AFI_BANK_WIDTH	The AFI bank width, derived from the corresponding memory interface width.

Table 1–5. UniPHY Parameters (Part 2 of 2)

Parameter Name	Description
AFI_CHIP_SELECT_WIDTH	The AFI chip select width, derived from the corresponding memory interface width.
AFI_DATA_MASK_WIDTH	The AFI data mask width.
AFI_CONTROL_WIDTH	The AFI control width, derived from the corresponding memory interface width.
AFI_DATA_WIDTH	The AFI data width.
AFI_DQS_WIDTH	The AFI DQS width.
DLL_DELAY_CTRL_WIDTH	The DLL delay output control width.
NUM_SUBGROUP_PER_READ_DQS	A read datapath parameter for timing purposes.
QVLD_EXTRA_FLOP_STAGES	A read datapath parameter for timing purposes.
READ_VALID_TIMEOUT_ WIDTH	A read datapath parameter; calibration fails when the timeout counter expires.
READ_VALID_FIFO_WRITE_ADDR _WIDTH	A read datapath parameter; the write address width for half-rate clocks.
READ_VALID_FIFO_READ_ ADDR_WIDTH	A read datapath parameter; the read address width for full-rate clocks.
MAX_LATENCY_COUNT_ WIDTH	A latency calibration parameter; the maximum latency count width.
MAX_READ_LATENCY	A latency calibration parameter; the maximum read latency.
READ_FIFO_READ_ADDR_ WIDTH	—
READ_FIFO_WRITE_ADDR_ WIDTH	—
MAX_WRITE_LATENCY_ COUNT_WIDTH	A write datapath parameter; the maximum write latency count width.
INIT_COUNT_WIDTH	An initailization sequence.
MRSC_COUNT_WIDTH	A memory-specific initialization parameter.
INIT_NOP_COUNT_WIDTH	A memory-specific initialization parameter.
MRS_CONFIGURATION	A memory-specific initialization parameter.
MRS_BURST_LENGTH	A memory-specific initialization parameter.
MRS_ADDRESS_MODE	A memory-specific initialization parameter.
MRS_DLL_RESET	A memory-specific initialization parameter.
MRS_IMP_MATCHING	A memory-specific initialization parameter.
MRS_ODT_EN	A memory-specific initialization parameter.
MRS_BURST_LENGTH	A memory-specific initialization parameter.
MEM_T_WL	A memory-specific initialization parameter.
MEM_T_RL	A memory-specific initialization parameter.
SEQ_BURST_COUNT_WIDTH	The burst count width for the sequencer.
VCALIB_COUNT_WIDTH	The width of a counter that the sequencer uses.
DOUBLE_MEM_DQ_WIDTH	—
HALF_AFI_DATA_WIDTH	—
CALIB_REG_WIDTH	The width of the calibration status register.
NUM_AFI_RESET	The number of AFI resets to generate.

PHY-to-Controller Interfaces

This section describes the typical modules that are connected to the UniPHY and the port name prefixes each module uses. Also this section describes using a custom controller and describes the AFI.

The AFI standardizes and simplifies the interface between controller and PHY for all Altera memory designs, thus allowing you to easily interchange your own controller code with Altera's high-performance controllers. The AFI PHY interface includes an administration block that configures the memory for calibration and performs necessary accesses to mode registers that configure the memory as required.

For half-rate designs, the address and command signals in the UniPHY are asserted for one `mem_clk` cycle (1T addressing), such that there are two input bits per address and command pin in half-rate designs. If you require a more conservative 2T addressing (where signals are asserted for two `mem_clk cycles`), drive both input bits (of the address and command signal) identically in half-rate designs.

Figure 1–13 shows the half-rate write operation.

Figure 1–13. Half-Rate Write with Word-Aligned Data

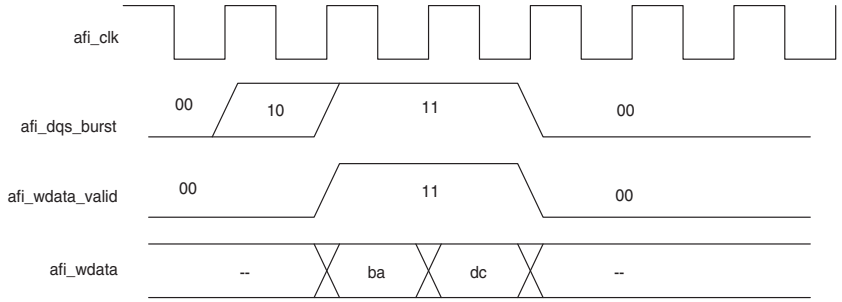
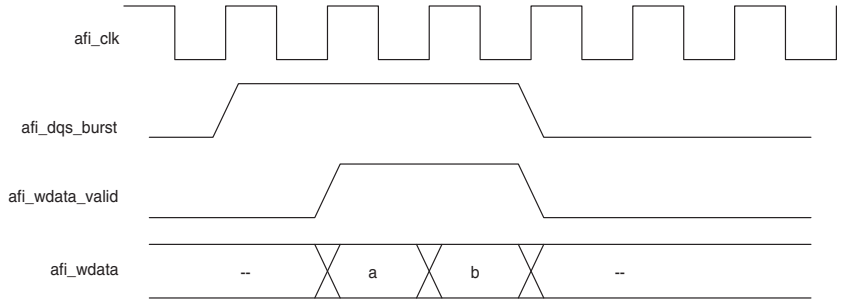


Figure 1–14 shows a full-rate write.

Figure 1–14. Full-Rate Write



After calibration is completed, the sequencer sends the write latency in number of clock cycles to the controller.

Figure 1–15 and Figure 1–16 show writes and reads, where the IP core writes data to and reads from the same address. In each example, `afi_rdata` and `afi_wdata` are aligned with controller clock (`afi_clk`) cycles. All the data in the bit vector is valid at once.

The AFI has the following conventions:

- With the AFI, high and low signals are combined in one signal, so for a single chip select (`afi_cs_n`) interface, `afi_cs_n[1:0]`, location 0 appears on the memory bus on one `mem_clk` cycle and location 1 on the next `mem_clk` cycle.

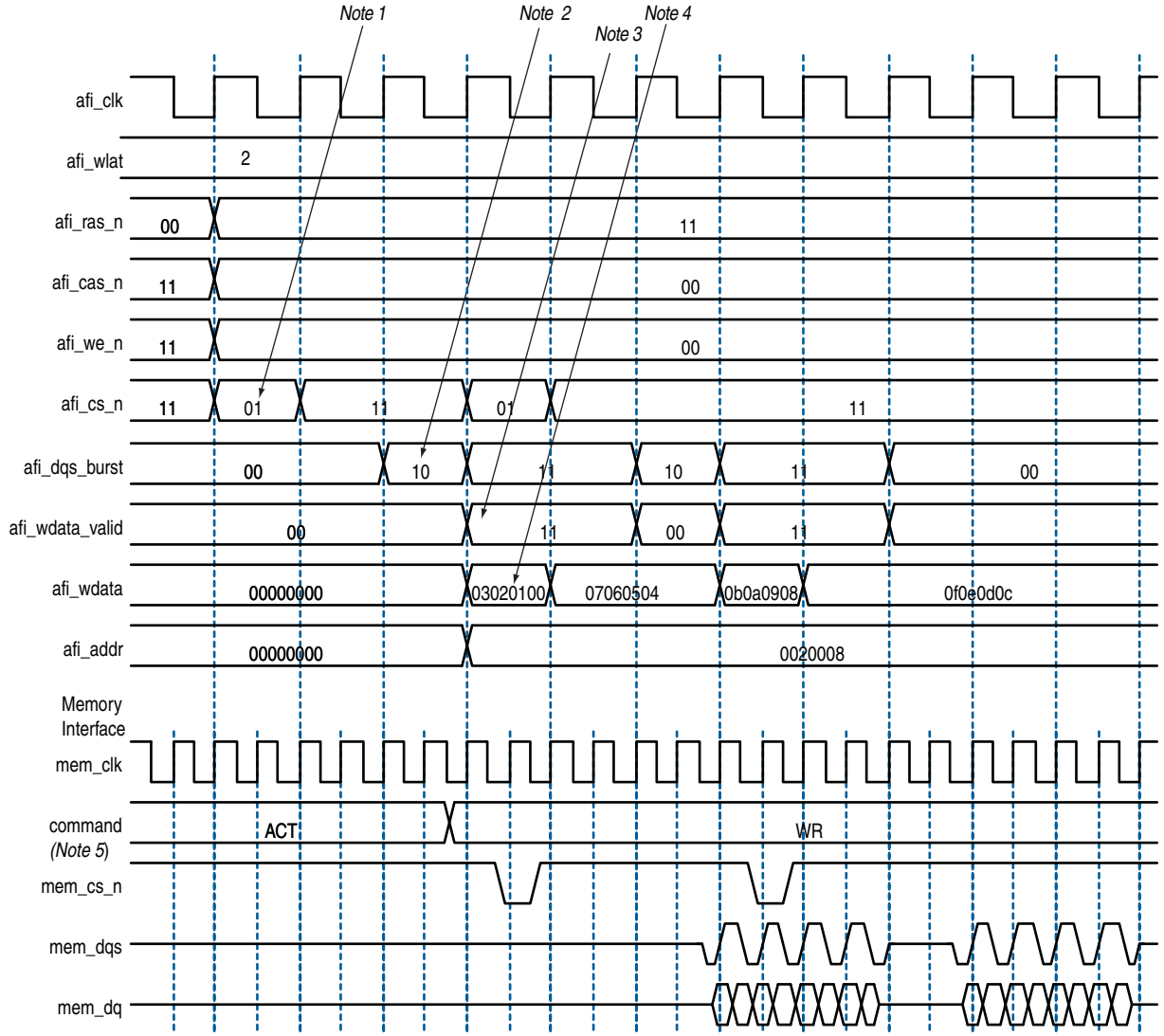
  This convention is maintained for all signals so for an 8 bit memory interface, the write data (`afi_wdata`) signal is `afi_wdata[31:0]`, where the first data on the DQ pins is `afi_wdata[7:0]`, then `afi_wdata[15:8]`, then `afi_wdata[23:16]`, then `afi_wdata[31:24]`.

- Spaced reads and writes have the following definitions:
  - Spaced writes—write commands separated by a gap of one controller clock (`afi_clk`) cycle.
  - Spaced reads—read commands separated by a gap of one controller clock (`afi_clk`) cycle.

Figure 1–15 and Figure 1–16 assume the following general points:

- The burst length is four.
- An 8-bit interface with one chip select.
- The data for one controller clock (`afi_clk`) cycle represents data for two memory clock (`mem_clk`) cycles (half-rate interface).

Figure 1–15. Word-Aligned Writes



Notes to Figure 1–15:

(1) To show the even alignment of `afi_cs_n`, expand the signal (this convention applies for all other signals).

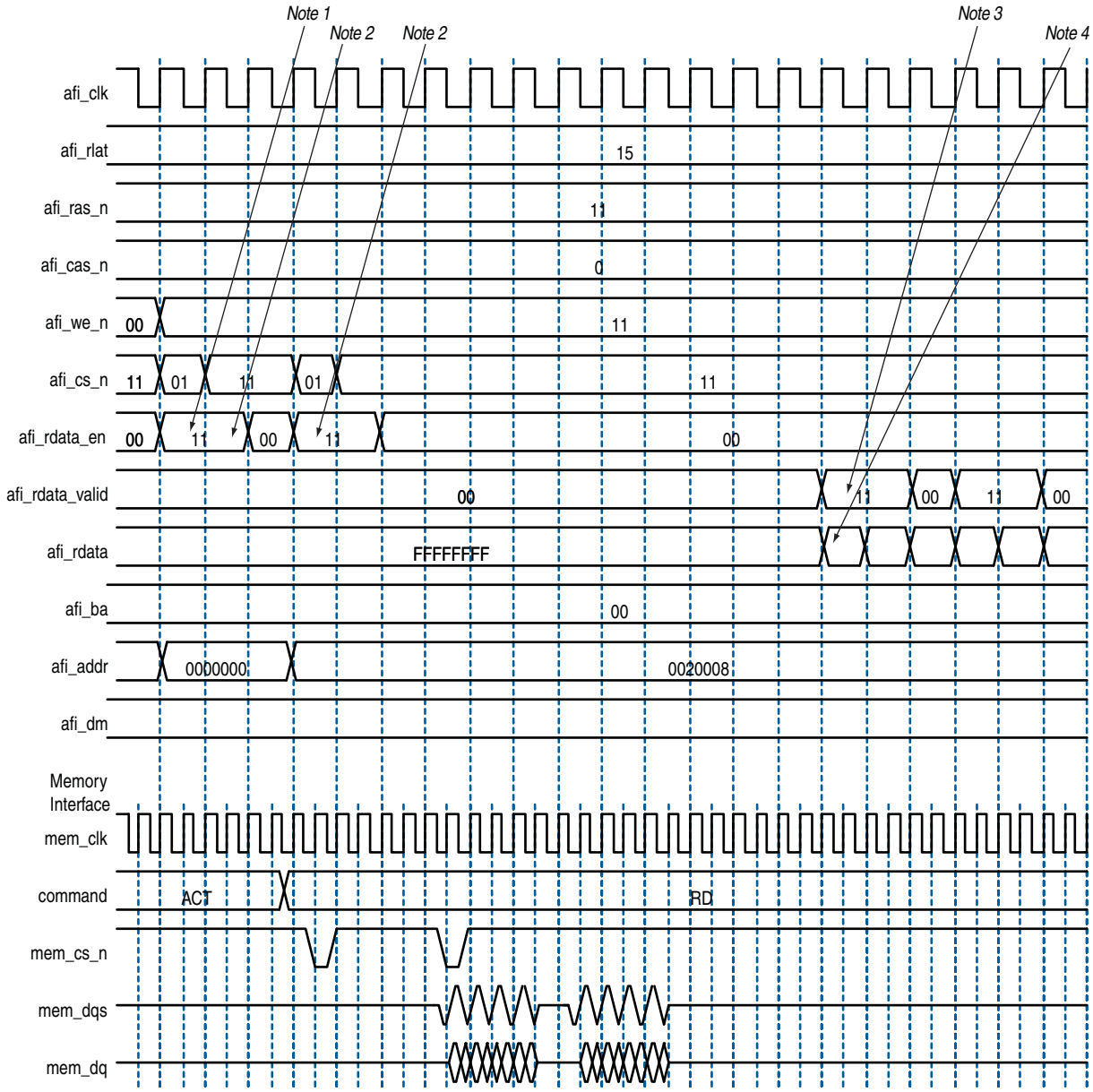
(2) The `afi_dqs_burst` must go high one memory clock cycle before `afi_wdata_valid`. Compare with the word-unaligned case.

(3) The `afi_wdata_valid` is asserted two `afi_wlat` controller clock (`afi_clk`) cycles after chip select (`afi_cs_n`) is asserted. The `afi_wlat` indicates the required write latency in the system. The value is determined during calibration and is dependant upon the relative delays in the address and command path and the write datapath in both the PHY and the external DDR SDRAM subsystem. The controller must drive `afi_cs_n` and then wait `afi_wlat` (two in this example) `afi_clks` before driving `afi_wdata_valid`.

(4) Observe the ordering of write data (`afi_wdata`). Compare this to data on the `mem_dq` signal.

(5) In all waveforms a command record is added that combines the memory pins `ras_n`, `cas_n` and `we_n` into the current command that is issued. This command is registered by the memory when chip select (`mem_cs_n`) is low. The important commands in the presented waveforms are WR = write, ACT = activate.

Figure 1–16. Word-Aligned Reads



Notes to Figure 1–16:

(1) For AFI, `afi_rdata_en` is required to be asserted one memory clock cycle before chip select (`afi_cs_n`) is asserted. In the half-rate `afi_clk` domain, this requirement manifests as the controller driving `11` (as opposed to the `01`) on `afi_rdata_en`.

(2) AFI requires that `afi_rdata_en` is driven for the duration of the read. In this example, it is driven to `11` for two half-rate `afi_clks`, which equates to driving to `1`, for the four memory clock cycles of this four-beat burst.

(3) The `afi_rdata_valid` returns 15 (`afi_rlat`) controller clock (`afi_clk`) cycles after `afi_rdata_en` is asserted. Returned is when the `afi_rdata_valid` signal is observed at the output of a register within the controller. A controller can use the `afi_rlat` value to determine when to register to returned data, but this is unnecessary as the `afi_rdata_valid` is provided for the controller to use as an enable when registering read data.

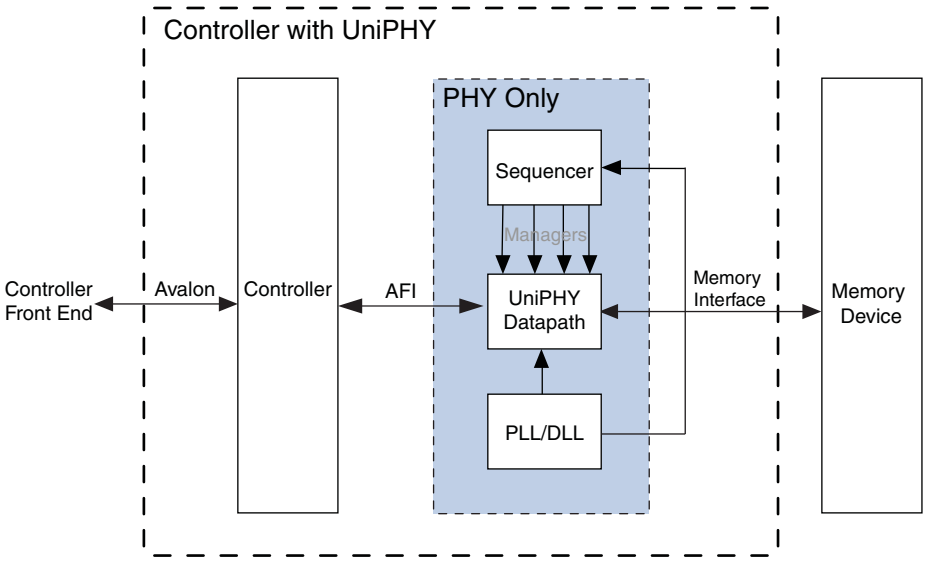
(4) Observe the alignment of returned read data with respect to data on the bus.

Using a Custom Controller

By default, the UniPHY-based external memory interface IP cores are delivered with both the PHY and the memory controller integrated, as depicted in Figure 1–17.

If you want to use your own custom controller with the UniPHY PHY, check the **Generate PHY only** box on the **PHY Settings** tab of the parameter editor and generate the IP. The resulting top-level IP consists of only the sequencer, UniPHY datapath, and PLL/DLL — the shaded area in Figure 1–17.

Figure 1–17. Memory Controller with UniPHY



The AFI interface is exposed at the top-level of the generated IP core; you can connect the AFI interface to your custom controller.

When you enable **Generate PHY only**, the generated example designs include the memory controller appropriately instantiated to mediate read/write commands from the traffic generator to the PHY-only IP.

For information on the AFI protocol, refer to the AFI 3.0 Specification.

For information on the example designs, refer to Traffic Generator and BIST Engine in chapter 7 of this section.

Using a Vendor-Specific Memory Model

You can replace the Altera-supplied memory model with a vendor-specific memory model. In general, you may find vendor-specific models to be standardized, thorough, and well supported, but sometimes more complex to setup and use.

If you do want to replace the Altera-supplied memory model with a vendor-supplied memory model, observe the following guidelines:

- Ensure that the vendor-supplied memory model that you have is correct for your memory device.
- Disconnect all signals from the default memory model and reconnect them to the vendor-supplied memory model.
- If you intend to run simulation from the Quartus II software, ensure that the **.qip** file points to the vendor-supplied memory model.

For related information, refer to Simulating Memory IP in volume 2 of the *External Memory Interface Handbook*.

AFI 3.0 Specification

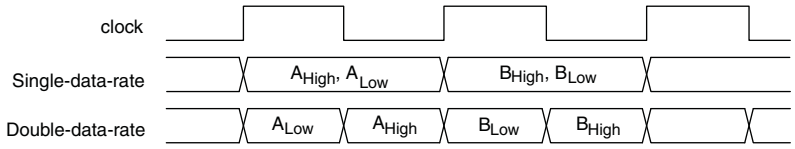
The Altera AFI interface defines communication between the controller and physical layer (PHY) in the external memory interface. The following sections describe the AFI implementation and the AFI signals.

Implementation

The AFI interface is a single-data-rate interface, meaning that data is transferred on the rising edge of each clock cycle. Most memory interfaces, however, operate at double-data-rate, transferring data on both the rising and falling edges of the clock signal. If the AFI interface is to directly control a double-data-rate signal, two single-data-rate bits must be transmitted on each clock cycle; the PHY then sends out one bit on the rising edge of the clock and one bit on the falling edge.

The AFI convention is to send the low part of the data first and the high part second, as shown in Figure 1–18.

Figure 1–18. Single versus Double Data Rate Transfer



Bus Width and AFI Ratio

In cases where the AFI clock frequency is one-half or one-quarter of the memory clock frequency, the AFI data must be twice or four times as wide, respectively, as the corresponding memory data. The ratio between AFI clock and memory clock frequencies is referred to as the AFI ratio. (A half-rate AFI interface has an AFI ratio of 2, while a quarter-rate interface has an AFI ratio of 4.)

In general, the width of the AFI signal depends on the following three factors:

- The size of the equivalent signal on the memory interface. For example, if `a[15:0]` is a DDR3 address input and the AFI clock runs at the same speed as the memory interface, the equivalent `afi_addr` bus will be 16-bits wide.
- The data rate of the equivalent signal on the memory interface. For example, if `d[7:0]` is a double-data-rate QDR II input data bus and the AFI clock runs at the same speed as the memory interface, the equivalent `afi_write_data` bus will be 16-bits wide.
- The AFI ratio. For example, if `cs_n` is a single-bit DDR3 chip select input and the AFI clock runs at half the speed of the memory interface, the equivalent `afi_cs_n` bus will be 2-bits wide.

The following formula summarizes the three factors described above:

```
AFI_width = memory_width * signal_rate * AFI_RATE_RATIO
```

The above formula is a general rule, but not all signals obey it. For definite signal-size information, refer to the specific table.

AFI Parameters

Table 1–6 through Table 1–14 list the AFI parameters grouped according to their functions.

Not all parameters are used for all protocols.

Parameters Affecting Bus Width

The following parameters affect the width of AFI signal buses. Parameters prefixed by `MEM_IF_` refer to the signal size at the interface between the PHY and memory device.

Table 1–6. Ratio Parameters

Parameter Name	Description
AFI_RATE_RATIO	The ratio between the AFI clock frequency and the memory clock frequency. For full-rate interfaces this value is 1, for half-rate interfaces the value is 2, and for quarter-rate interfaces the value is 4.
DATA_RATE_RATIO	The number of data bits transmitted per clock cycle. For single-date rate protocols this value is 1, and for double-data rate protocols this value is 2.
ADDR_RATE_RATIO	The number of address bits transmitted per clock cycle. For single-date rate address protocols this value is 1, and for double-data rate address protocols this value is 2.

Table 1–7. Memory Interface Parameters

Parameter Name	Description
MEM_IF_ADDR_WIDTH	The width of the address bus on the memory device(s).
MEM_IF_BANKADDR_WIDTH	The width of the bank address bus on the interface to the memory device(s). Typically, the $\log_2$ of the number of banks.
MEM_IF_CS_WIDTH	The number of chip selects on the interface to the memory device(s).
MEM_IF_WRITE_DQS_WIDTH	The number of DQS (or write clock) signals on the write interface. For example, the number of DQS groups.
MEM_IF_CLK_PAIR_COUNT	The number of CK/CK# pairs.
MEM_IF_DQ_WIDTH	The number of DQ signals on the interface to the memory device(s). For single-ended interfaces such as QDR II, this value is the number of D or Q signals.
MEM_IF_DM_WIDTH	The number of data mask pins on the interface to the memory device(s).
MEM_IF_READ_DQS_WIDTH	The number of DQS signals on the read interface. For example, the number of DQS groups.

Table 1–8. Derived AFI Parameters

Parameter Name	Derivation Equation
AFI_ADDR_WIDTH	MEM_IF_ADDR_WIDTH * AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_BANKADDR_WIDTH	MEM_IF_BANKADDR_WIDTH * AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_CONTROL_WIDTH	AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_CS_WIDTH	MEM_IF_CS_WIDTH * AFI_RATE_RATIO
AFI_DM_WIDTH	MEM_IF_DM_WIDTH * AFI_RATE_RATIO * DATA_RATE_RATIO
AFI_DQ_WIDTH	MEM_IF_DQ_WIDTH * AFI_RATE_RATIO * DATA_RATE_RATIO
AFI_WRITE_DQS_WIDTH	MEM_IF_WRITE_DQS_WIDTH * AFI_RATE_RATIO
AFI_LAT_WIDTH	6
AFI_RLAT_WIDTH	AFI_LAT_WIDTH
AFI_WLAT_WIDTH	AFI_LAT_WIDTH * MEM_IF_WRITE_DQS_WIDTH
AFI_CLK_PAIR_COUNT	MEM_IF_CLK_PAIR_COUNT
AFI_WRANK_WIDTH	Number of ranks * MEM_IF_WRITE_DQS_WIDTH * AFI_RATE_RATIO
AFI_RRANK_WIDTH	Number of ranks * MEM_IF_READ_DQS_WIDTH * AFI_RATE_RATIO

AFI Signals

The following tables list the AFI signals grouped according to their functions.

In each table, the **direction** column denotes the direction of the signal relative to the PHY. For example, a signal defined as an output passes out of the PHY to the controller. The AFI specification does not include any bidirectional signals.

Not all signals are used for all protocols.

Clock and Reset Signals

The AFI interface provides up to two clock signals and an asynchronous reset signal.

Table 1–9. Clock and Reset Signals

Signal Name	Direction	Width	Description
afi_clk	Output	1	Clock with which all data exchanged on the AFI bus is synchronized. In general, this clock is referred to as full-rate, half-rate, or quarter-rate, depending on the ratio between the frequency of this clock and the frequency of the memory device clock.
afi_half_clk	Output	1	Clock signal that runs at half the speed of the afi_clk. The controller uses this signal when the half-rate bridge feature is in use. This signal is optional.
afi_reset_n	Output	1	Asynchronous reset output signal. You must synchronize this signal to the clock domain in which you use it.

Address and Command Signals

The address and command signals encode read/write/configuration commands to send to the memory device. The address and command signals are single-data rate signals.

Table 1–10. Address and Command Signals (Part 1 of 2)

Signal Name	Direction	Width	Description
afi_ba	Input	AFI_BANKADDR_WIDTH	Bank address.
afi_cke	Input	AFI_CLK_EN_WIDTH	Clock enable.
afi_cs_n	Input	AFI_CS_WIDTH	Chip select signal. (The number of chip selects may not match the number of ranks; for example, RDIMMs and LRDIMMs require a minimum of 2 chip select signals for both single-rank and dual-rank configurations. Consult your memory device datasheet for information about chip select signal width.)
afi_ras_n	Input	AFI_CONTROL_WIDTH	RAS# (for DDR2 and DDR3 memory devices.)

Table 1–10. Address and Command Signals (Part 2 of 2)

Signal Name	Direction	Width	Description
afi_we_n	Input	AFI_CONTROL_WIDTH	WE# (for DDR2, DDR3, and RLDRAM II memory devices.)
afi_cas_n	Input	AFI_CONTROL_WIDTH	CAS# (for DDR2 and DDR3 memory devices.)
afi_ref_n	Input	AFI_CONTROL_WIDTH	REF# (for RLDRAM II memory devices.)
afi_rst_n	Input	AFI_CONTROL_WIDTH	RESET# (for DDR3 memory devices.)
afi_odt	Input	AFI_CLK_EN_WIDTH	On-die termination signal for DDR2 and DDR3 memory devices. (Do not confuse this memory device signal with the FPGA's internal on-chip termination signal.)
afi_mem_clk_disable	Input	AFI_CLK_PAIR_COUNT	When this signal is asserted, mem_clk and mem_clk_n are disabled. This signal is used in low-power mode.
afi_wps_n	Output	AFI_CS_WIDTH	WPS (for QDR II/II+ memory devices.)
afi_rps_n	Output	AFI_CS_WIDTH	RPS (for QDR II/II+ memory devices.)

Write Data Signals

The signals described in this section control the data, data mask, and strobe signals passed to the memory device during write operations.

Table 1–11. Write Data Signals

Signal Name	Direction	Width	Description
afi_dqs_burst	Input	AFI_WRITE_DQS_WIDTH	Controls the enable on the strobe (DQS) pins for DDR2, DDR3, and LPDDR2 memory devices. When this signal is asserted, mem_dqs and mem_dqsn are driven. This signal must be asserted before afi_wdata_valid to implement the write preamble, and must be driven for the correct duration to generate a correctly timed mem_dqs signal.
afi_wdata_valid	Input	AFI_WRITE_DQS_WIDTH	Write data valid signal. This signal controls the output enable on the data and data mask pins.
afi_wdata	Input	AFI_DQ_WIDTH	Write data signal to send to the memory device at double-data rate. This signal controls the PHY's mem_dq output.
afi_dm	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_dm signal for DDR2, DDR3, LPDDR2, and RLDRAM II memory devices.)
afi_bws_n	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_bws_n signal for QDR II/II+ memory devices.
afi_wrank	Input	AFI_WRANK_WIDTH	Shadow register signal. Signal indicating the rank to which the controller is writing, so that the PHY can switch to the appropriate setting. Signal timing is identical to afi_dqs_burst; that is, afi_wrank must be asserted at the same time as afi_dqs_burst, and must be of the same duration.

Read Data Signals

The signals described in this section control the data sent from the memory device during read operations.

Table 1–12. Read Data Signals

Signal Name	Direction	Width	Description
afi_rdata_en	Input	AFI_RATE_RATIO	Read data enable. Indicates that the memory controller is currently performing a read operation. This signal is held high only for cycles of relevant data (read data masking). If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rdata_en_full	Input	AFI_RATE_RATIO	Read data enable full. Indicates that the memory controller is currently performing a read operation. This signal is held high for the entire read burst. If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rdata	Output	AFI_DQ_WIDTH	Read data from the memory device. This data is considered valid only when afi_rdata_valid is asserted by the PHY.
afi_rdata_valid	Output	AFI_RATE_RATIO	Read data valid. When asserted, this signal indicates that the afi_rdata bus is valid. If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rrank	Input	AFI_RRANK_WIDTH	Shadow register signal. Signal indicating the rank from which the controller is reading, so that the PHY can switch to the appropriate setting. Must be asserted at the same time as afi_rdata_en when issuing a read command, and once asserted, must remain unchanged until the controller issues a new read command to another rank.

Calibration Status Signals

The PHY instantiates a sequencer which calibrates the memory interface with the memory device and some internal components such as read FIFOs and valid FIFOs. The sequencer reports the results of the calibration process to the controller through the AFI interface. This section describes the calibration status signals.

Table 1–13. Calibration Status Signals

Signal Name	Direction	Width	Description
afi_cal_success	Output	1	Asserted to indicate that calibration has completed successfully.
afi_cal_fail	Output	1	Asserted to indicate that calibration has failed.
afi_cal_req	Input	1	Effectively a synchronous reset for the sequencer. When this signal is asserted, the sequencer returns to the reset state; when this signal is released, a new calibration sequence begins.
afi_wlat	Output	AFI_WLAT_WIDTH	The required write latency in afi_clk cycles, between address/command and write data being issued at the PHY/controller interface. The afi_wlat value can be different for different groups; each group's write latency can range from 0 to 63. If write latency is the same for all groups, only the lowest 6 bits are required.
afi_rlat	Output	AFI_RLAT_WIDTH	The required read latency in afi_clk cycles between address/command and read data being returned to the PHY/controller interface. Values can range from 0 to 63.

Tracking Management Signals

When tracking management is enabled, the sequencer can take control over the AFI interface at given intervals, and issue commands to the memory device to track the internal DQS Enable signal alignment to the DQS signal returning from the memory device. The tracking management portion of the AFI interface provides a means for the sequencer and the controller to exchange handshake signals.

Table 1–14. Tracking Management Signals

Signal Name	Direction	Width‘	Description
afi_ctl_refresh_done	Input	MEM_IF_CS_WIDTH	Handshaking signal from controller to tracking manager, indicating that a refresh has occurred and waiting for a response.
afi_seq_busy	Output	MEM_IF_CS_WIDTH	Handshaking signal from sequencer to controller, indicating when DQS tracking is in progress.
afi_ctl_long_idle	Input	MEM_IF_CS_WIDTH	Handshaking signal from controller to tracking manager, indicating that it has exited low power state without a periodic refresh, and waiting for response.

Register Maps

Table 1–15 lists the overall register mapping for the DDR2, DDR3, and LPDDR2 SDRAM Controllers with UniPHY.

Table 1–15. Register Map

Address	Description
UniPHY Register Map	
0x001	Reserved.
0x004	UniPHY status register 0.
0x005	UniPHY status register 1.
0x006	UniPHY status register 2.
0x007	UniPHY memory initialization parameters register 0.
Controller Register Map	
0x100	Reserved.
0x110	Controller status and configuration register.
0x120	Memory address size register 0.
0x121	Memory address size register 1.
0x122	Memory address size register 2.
0x123	Memory timing parameters register 0.
0x124	Memory timing parameters register 1.
0x125	Memory timing parameters register 2.
0x126	Memory timing parameters register 3.
0x130	ECC control register.
0x131	ECC status register.
0x132	ECC error address register.

UniPHY Register Map

The UniPHY register map allows you to control the memory components' mode register settings. Table 1–16 lists the register map for UniPHY.

Table 1–16. UniPHY Register Map (Part 1 of 3)

Address	Bit	Name	Default	Access	Description
0x001	15:0	Reserved.	0	—	Reserved for future use.
	31:16	Reserved.	0	—	Reserved for future use.
0x002	15:0	Reserved.	0	—	Reserved for future use.
	31:16	Reserved.	0	—	Reserved for future use.

Table 1–16. UniPHY Register Map (Part 2 of 3)

Address	Bit	Name	Default	Access	Description
0x004	0	SOFT_RESET	—	Write only	Initiate a soft reset of the interface. This bit is automatically deasserted after reset.
	23:1	Reserved.	0	—	Reserved for future use.
	24	AFI_CAL_SUCCESS	—	Read only	Reports the value of the UniPHY `afi_cal_success`. Writing to this bit has no effect.
	25	AFI_CAL_FAIL	—	Read only	Reports the value of the UniPHY `afi_cal_fail`. Writing to this bit has no effect.
	26	Reserved.	0	—	Reserved for future use.
	31:27	Reserved.	0	—	Reserved for future use.
0x005	7:0	Reserved.	0	—	Reserved for future use.
	15:8	Reserved.	0	—	Reserved for future use.
	23:16	Reserved.	0	—	Reserved for future use.
	31:24	Reserved.	0	—	Reserved for future use.
0x006	7:0	INIT_FAILING_STAGE	—	Read only	Initial failing error stage of calibration. Only applicable if AFI_CAL_FAIL=1.
	15:8	INIT_FAILING_SUBSTAGE	—	Read only	Initial failing error substage of calibration. Only applicable if AFI_CAL_FAIL=1.
	23:16	INIT_FAILING_GROUP	—	Read only	Initial failing error group of calibration. Only applicable if AFI_CAL_FAIL=1. Returns failing DQ pin instead of failing group, if: INIT_FAILING_STAGE=1 and INIT_FAILING_SUBSTAGE=3. Or INIT_FAILING_STAGE=4 and INIT_FAILING_SUBSTAGE=1.
	31:24	Reserved.	0	—	Reserved for future use.
0x007	31:0	DQS_DETECT	—	Read only	Identifies if DQS edges have been identified for each of the groups. Each bit corresponds to one DQS group.
0x008 (DDR2)	1:0	RTT_NOM	—	Read only	Rtt (nominal) setting of the DDR2 Extended Mode Register used during memory initialization.
	31:2	Reserved.	0	—	Reserved for future use.
0x008 (DDR3)	2:0	RTT_NOM	—		Rtt (nominal) setting of the DDR3 MR1 mode register used during memory initialization.
	4:3	Reserved.	0		Reserved for future use.
	6:5	ODS	—		Output driver impedence control setting of the DDR3 MR1 mode register used during memory initialization.
	8:7	Reserved.	0		Reserved for future use.
	10:9	RTT_WR	—		Rtt (writes) setting of the DDR3 MR2 mode register used during memory initialization.
	31:11	Reserved.	0		Reserved for future use.

Table 1–16. UniPHY Register Map (Part 3 of 3)

Address	Bit	Name	Default	Access	Description
0x008 (LPDDR 2)	3:0	DS			Driver impedence control for MR3 during initialization.
	31:4	Reserved.			Reserved for future use.

Controller Register Map

The controller register map allows you to control the memory controller settings.

For information on the controller register map, refer to "Controller Register Map" on page 5–29 of this volume.

Ping Pong PHY

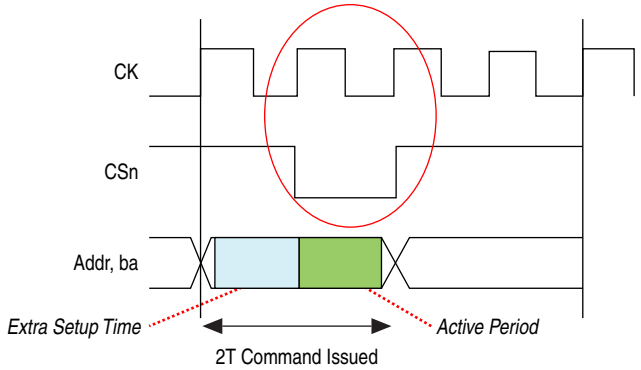
Ping Pong PHY is an implementation of UniPHY that allows two memory interfaces to share address and command buses through time multiplexing. Compared to having two independent interfaces, Ping Pong PHY uses fewer pins and less logic, while maintaining equivalent throughput.

The Ping Pong PHY supports only quarter-rate configurations of the DDR3 protocol on Arria V GZ and Stratix V devices.

Feature Description

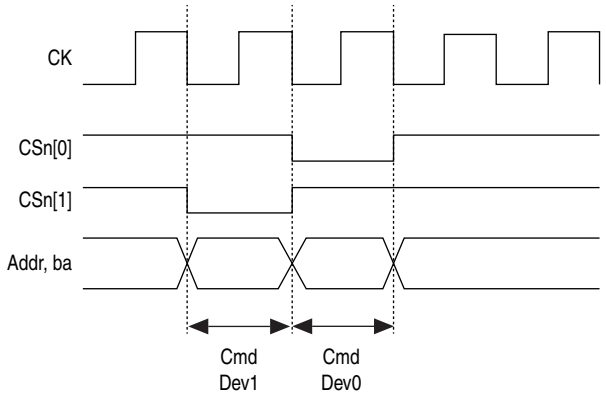
In conventional UniPHY, the address and command buses of a DDR3 quarter-rate interface use 2T time—meaning that they are issued for two full-rate clock cycles, as illustrated in Figure 1–19.

Figure 1–19. 2T Command Timing



With the Ping Pong PHY, address and command signals from two independent controllers are multiplexed onto shared buses by delaying one of the controller outputs by one full-rate clock cycle. The result is 1T timing, with a new command being issued on each full-rate clock cycle. Figure 1–20 shows address and command timing for the Ping Pong PHY.

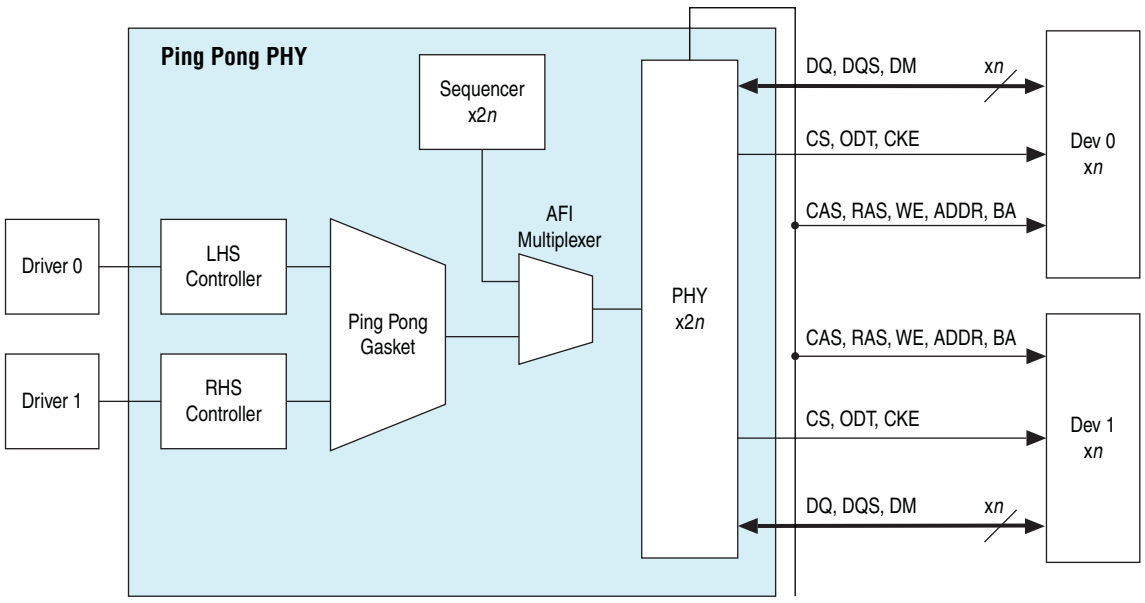
Figure 1–20. 1T Command Timing Use by Ping Pong PHY



Architecture

Figure 1–21 shows a top-level block diagram of the Ping Pong PHY. Functionally, the IP looks like two independent memory interfaces. The two controller blocks are referred to as right-hand side (RHS) and left-hand side (LHS), respectively. A gasket block located between the controllers and the PHY merges the AFI signals. The PHY is double data width and supports both memory devices. The sequencer is the same as with regular UniPHY, and calibrates the entire double-width PHY.

Figure 1–21. Ping Pong PHY Architecture



Ping Pong Gasket

Figure 1–22 shows the gasket architecture. The gasket delays and remaps quarter-rate signals so that they are correctly time-multiplexed at the full-rate PHY output. The gasket also merges address and command buses ahead of the PHY.

AFI interfaces at the input and output of the gasket provide compatibility with the PHY and with memory controllers.

Figure 1–22. Ping Pong PHY Gasket Architecture

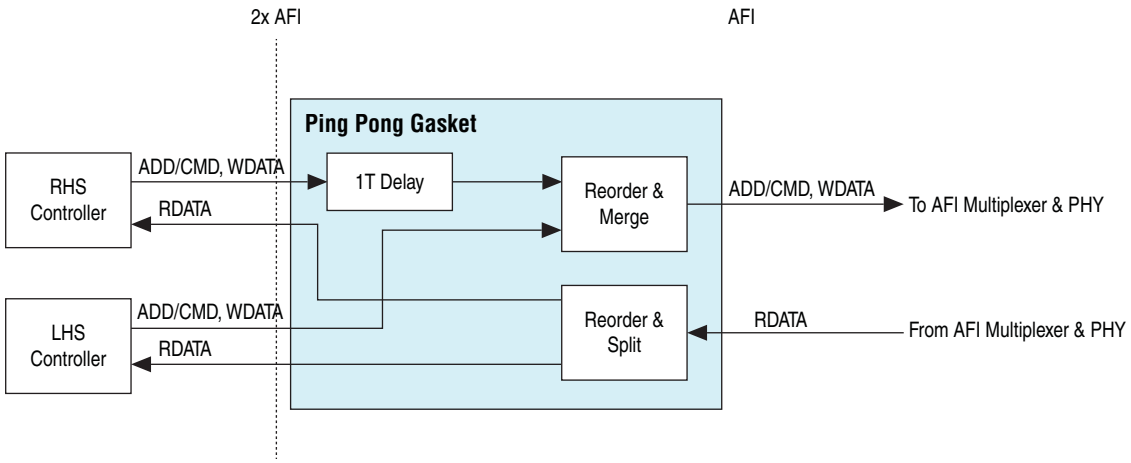


Table 1–17 shows how the gasket processes key AFI signals.

Table 1–17. Key AFI Signals Processed by Ping Pong PHY Gasket (Part 1 of 2)

Signal	Direction (Width multiplier)	Description	Gasket Conversions
cas, ras, we, addr, ba	Controller (1x) to PHY (1x)	Address and command buses shared between devices.	Delay RHS by 1T; merge.
cs, odt, cke	Controller (1x) to PHY (2x)	Chip select, on-die termination, and clock enable, one per device.	Delay RHS by 1T; reorder, merge.
wdata, wdata_valid, dqs_burst, dm	Controller (1x) to PHY (2x)	Write datapath signals, one per device.	Delay RHS by 1T; reorder, merge.
rdata_en_rd, rdata_en_rd_full	Controller (1x) to PHY (2x)	Read datapath enable signals indicating controller performing a read operation, one per device.	Delay RHS by 1T.
rdata_rdata_valid	PHY (2x) to Controller (1x)	Read data, one per device.	Reorder; split.
cal_fail, cal_success, seq_busy, wlat, rlat	PHY (1x) to Controller (1x)	Calibration result, one per device.	Pass through.

Table 1–17. Key AFI Signals Processed by Ping Pong PHY Gasket (Part 2 of 2)

Signal	Direction (Width multiplier)	Description	Gasket Conversions
rst_n, mem_clk_disable, ctl_refresh_done, ctl_long_idle	Controller (1x) to PHY (1x)	Reset and DQS tracking signals, one per PHY.	AND (&)
cal_req, init_req	Controller (1x) to PHY (1x)	Controller to sequencer requests.	OR (|)
wrank, rrank	Controller (1x) to PHY (2x)	Shadow register support.	Delay RHS by 1T; reorder; merge.

Calibration

The sequencer treats the Ping Pong PHY as a regular interface of double the width. For example, in the case of two x16 devices, the sequencer calibrates both devices together as a x32 interface. The sequencer chip select signal fans out to both devices so that they are treated as a single interface. The VFIFO calibration process is unchanged. For LFIFO calibration, the LFIFO buffer is duplicated for each interface and the worst-case read datapath delay of both interfaces is used.

Operation

To use the Ping Pong PHY, you configure a single memory interface according to your requirements, and then select the **Enable Ping Pong PHY** option in the **Advanced PHY Options** section of the **PHY Settings** tab in the DDR3 parameter editor.

The Quartus II software then replicates the interface, resulting in two memory controllers and a shared PHY, with the gasket block inserted between the controllers and PHY. The system makes the necessary modifications to top-level component connections, as well as the PHY read and write datapaths, and the AFI mux, without further input from you.

Efficiency Monitor and Protocol Checker

The Efficiency Monitor and Protocol Checker is a feature available with the DDR2, DDR3, and LPDDR2 SDRAM controllers with UniPHY and the RLDRAM II Controller with UniPHY. The Efficiency Monitor and Protocol Checker allows measurement of traffic efficiency on the Avalon-MM bus between the controller and user logic, measures read latencies, and checks the legality of Avalon commands passed from the master. The following sections describe the parts of this feature.

Efficiency Monitor

The Efficiency Monitor reports read and write throughput on the controller input, by counting command transfers and wait times, and making that information available to the External Memory Interface Toolkit via an Avalon slave port. This information may be useful to you when experimenting with advanced controller settings, such as command look ahead depth and burst merging.

Protocol Checker

The Protocol Checker checks the legality of commands on the controller's input interface against Altera's Avalon interface specification, and sets a flag in a register on an Avalon slave port if an illegal command is detected.

Read Latency Counter

The Read Latency Counter measures the minimum and maximum wait times for read commands to be serviced on the Avalon bus. Each read command is time-stamped and placed into a FIFO buffer upon arrival, and latency is determined by comparing that timestamp to the current time when the first beat of the returned read data is provided back to the master.

Using the Efficiency Monitor and Protocol Checker

To include the Efficiency Monitor and Protocol Checker when you generate your IP core, on the **Diagnostics** tab in the parameter editor, turn on **Enable the Efficiency Monitor and Protocol Checker on the Controller Avalon Interface**.

To see the results of the data compiled by the Efficiency Monitor and Protocol Checker, use the External Memory Interface Toolkit.

For information on the External Memory Interface Toolkit, refer to UniPHY External Memory Interface Debug Toolkit, in section 2 of this volume. For information about the Avalon interface, refer to *Avalon Interface Specifications*.

Avalon CSR Slave and JTAG Memory Map

Table 1–18 lists the memory map of registers inside the Efficiency Monitor and Protocol Checker; this information is only of interest if you want to communicate directly with the Efficiency Monitor and Protocol Checker without using the External Memory Interface Toolkit. This CSR map is not part of the UniPHY CSR map.

Table 1–18. Avalon CSR Slave and JTAG Memory Map (Part 1 of 3)

Address	Bit	Name	Default	Access	Description
0x01	31:0	Reserved	0	—	Used internally by EMIF Toolkit to identify Efficiency Monitor type.
0x02	31:0	Reserved	0	—	Used internally by EMIF Toolkit to identify Efficiency Monitor version.

Table 1–18. Avalon CSR Slave and JTAG Memory Map (Part 2 of 3)

Address	Bit	Name	Default	Access	Description
0x08	0	Efficiency Monitor reset	—	Write only	Write a 0 to reset.
	7:1	Reserved	—	—	Reserved for future use.
	8	Protocol Checker reset	—	Write only	Write a 0 to reset.
	15:9	Reserved	—	—	Reserved for future use.
	16	Start/stop Efficiency Monitor	—	Read/Write	Starting and stopping stastics gathering.
	23:17	Reserved	—	—	Reserved for future use.
	31:24	Efficiency Monitor status	—	Read Only	bit 0: Efficiency Monitor stopped bit 1: Waiting for start of pattern bit 2: Running bit 3: Counter saturation
0x10	15:0	Efficiency Monitor address width	—	Read Only	Address width of the Efficiency Monitor.
	31:16	Efficiency Monitor data width	—	Read Only	Data Width of the Efficiency Monitor.
0x11	15:0	Efficiency Monitor byte enable	—	Read Only	Byte enable width of the Efficiency Monitor.
	31:16	Efficiency Monitor burst count width	—	Read Only	Burst count width of the Efficiency Monitor.
0x14	31:0	Cycle counter	—	Read Only	Clock cycle counter for the Efficiency Monitor. Lists the number of clock cycles elapsed before the Efficiency Monitor stopped.
0x18	31:0	Transfer counter	—	Read Only	Counts any read or write data transfer cycle.
0x1C	31:0	Write counter	—	Read Only	Counts write requests, including those during bursts.
0x20	31:0	Read counter	—	Read Only	Counts read requests.
0x24	31:0	Readtotal counter	—	Read Only	Counts read requests (total burst requests).
0x28	31:0	NTC waitrequest counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to slave wait request high.
0x2C	31:0	NTC noreaddatavalid counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to slave not having read data.
0x30	31:0	NTC master write idle counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to master not issuing command, or pause in write burst.
0x34	31:0	NTC master idle counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to master not issuing command anytime.
0x40	31:0	Read latency min	—	Read Only	The lowest read latency value.
0x44	31:0	Read latency max	—	Read Only	The highest read latency value.
0x48	31:0	Read latency total [31:0]	—	Read Only	The lower 32 bits of the total read latency.

Table 1–18. Avalon CSR Slave and JTAG Memory Map (Part 3 of 3)

Address	Bit	Name	Default	Access	Description
0x49	31:0	Read latency total [63:32]	—	Read Only	The upper 32 bits of the total read latency.
0x50	7:0	Illegal command	—	Read Only	Bits used to indicate which illegal command has occurred. Each bit represents a unique error.
	31:8	Reserved	—		Reserved for future use.

UniPHY Calibration Stages

This section describes the calibration stages performed by the DDR2, DDR3, and LPDDR2 SDRAM, QDR II and QDR II+ SRAM, and RLDRAM II Controllers with UniPHY, and the RLDRAM 3 PHY-only IP. This information is useful in debugging calibration failures. The section includes an overview of calibration, explanation of the calibration stages, and a list of generated calibration signals. The information in this section applies only to the Nios II-based sequencer used in the DDR2, DDR3, and LPDDR2 SDRAM Controllers with UniPHY versions 10.0 and later, and, optionally, in the QDR II and QDR II+ SRAM and RLDRAM II Controllers with UniPHY version 11.0 and later, and the RLDRAM 3 PHY-only IP. The information in this section applies to the Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V device families.

For QDR II and QDR II+ SRAM and RLDRAM II Controllers with UniPHY version 11.0 and later, you have the option to select either the RTL-based sequencer or the Nios II-based sequencer. Generally, choose the RTL-based sequencer when area is the major consideration, and choose the Nios II-based sequencer when performance is the major consideration.

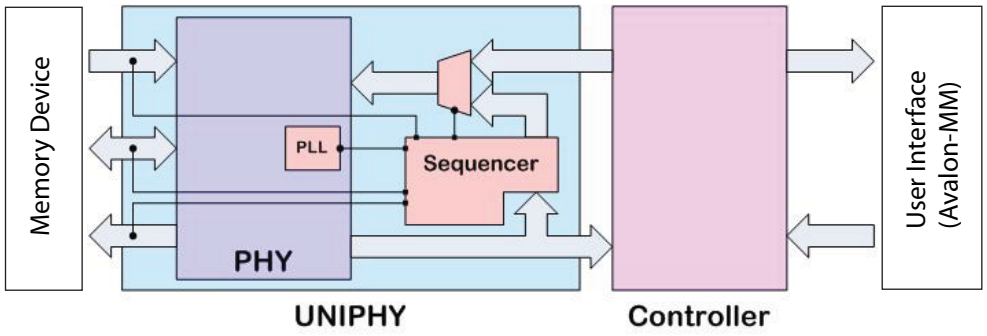
For RLDRAM 3, write leveling is not performed. The sequencer does not attempt to optimize margin for the `tCKDK` timing requirement.

Overview

Calibration configures the memory interface (PHY and I/Os) so that data can pass reliably to and from memory. The sequencer illustrated in Figure 1–23 calibrates the PHY and the I/Os. To correctly transmit data between a memory device and the FPGA at high speed, the data must be center-aligned with the data clock.

Calibration also determines the delay settings needed to center-align the various data signals with respect to their clocks. I/O delay chains implement the required delays in accordance with the computed alignments. The Nios II-based sequencer performs two major tasks: FIFO buffer calibration and I/O calibration. FIFO buffer calibration adjusts FIFO lengths and I/O calibration adjusts any delay chain and phase settings to center-align data signals with respect to clock signals for both reads and writes. When the calibration process completes, the sequencer shuts off and passes control to the memory controller.

Figure 1–23. Sequencer in Memory Interface Logic



Calibration Stages

The calibration process begins when the PHY reset signal deasserts and the PLL and DLL lock. The following stages of calibration take place:

1. Read calibration part one—DQS enable calibration (only for DDR2 and DDR3 SDRAM Controllers with UniPHY) and DQ/DQS centering
2. Write calibration part one—Leveling
3. Write calibration part two—DQ/DQS centering
4. Read calibration part two—Read latency minimization

For multirank calibration, the sequencer transmits every read and write command to each rank in sequence. Each read and write test is successful only if all ranks pass the test. The sequencer calibrates to the intersection of all ranks.

Assumptions

The calibration process assumes the following conditions; if either of these conditions is not true, calibration likely fails in its early stages:

- The address and command paths must be functional; calibration does not tune the address and command paths. (The Quartus II software fully analyzes the timing for the address and command paths, and the slack report is accurate, assuming the correct board timing parameters.)
- At least one bit per group must work before running per-bit-deskew calibration. (This assumption requires that DQ-to-DQS skews be within the recommended 20 ps.)

Memory Initialization

The memory is powered up according to protocol initialization specifications. All ranks power up simultaneously. Once powered, the device is ready to receive mode register load commands. This part of initialization occurs separately for each rank. The sequencer issues mode register set commands on a per-chip-select basis and initializes the memory to the user-specified settings.

Stage 1: Read Calibration Part One—DQS Enable Calibration and DQ/DQS Centering

Read calibration occurs in two parts. Part one is DQS enable calibration with DQ/DQS centering, which happens during stage 1 of the overall calibration process; part two is read latency minimization, which happens during stage 4 of the overall calibration process.

The objectives of DQS enable calibration and DQ/DQS centering are as follows:

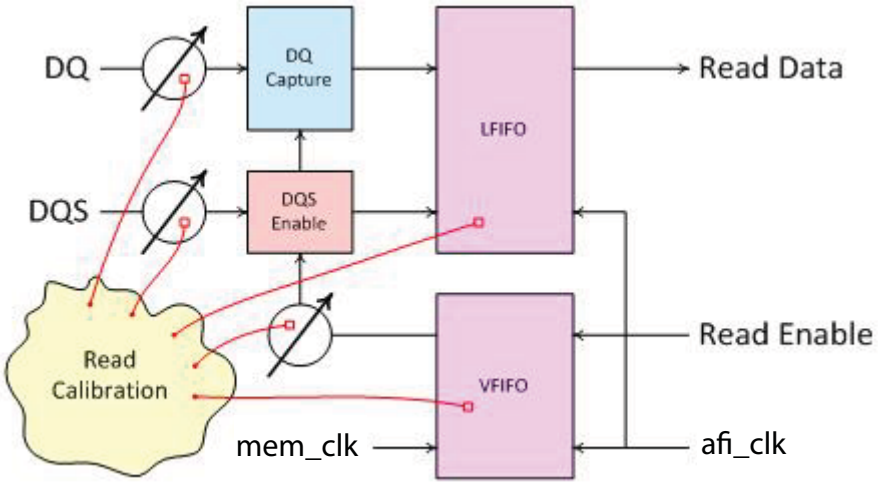
- To calculate when the read data is received after a read command is issued to setup the Data Valid Prediction FIFO (VFIFO) cycle
- To align the input data (DQ) with respect to the clock (DQS) to maximize the read margins (DDR2 and DDR3 only)

DQS enable calibration and DQ/DQS centering consists of the following actions:

- Guaranteed Write
- DQS Enable Calibration
- DQ/DQS Centering

Figure 1–24 illustrates the components in the read data path that the sequencer calibrates in this stage. (The round knobs in the figure represent configurable hardware over which the sequencer has control.)

Figure 1–24. Read Data Path Calibration Model

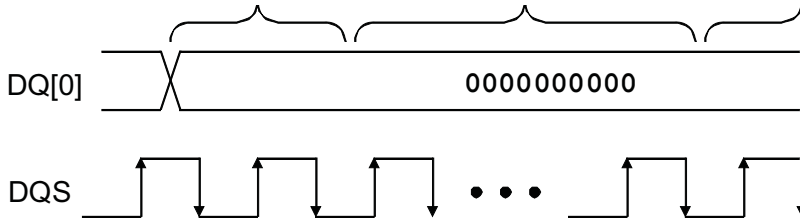


Guaranteed Write

Since initially no communication can be reliably performed with the memory device, the sequencer uses a guaranteed write mechanism to write data into the memory device. (For the QDR II protocol, guaranteed write is not necessary, a simple write mechanism is sufficient.)

The guaranteed write is a write command issued with all data pins, all address and bank pins, and all command pins (except chip select) held constant. The sequencer begins toggling DQS well before the expected latch time at memory and continues to toggle DQS well after the expected latch time at memory. DQ-to-DQS relationship is not a factor at this stage because DQ is held constant. Figure 1–25 illustrates a guaranteed write of zeros.

Figure 1–25. Guaranteed Write of Zeros



The guaranteed write consists of a series of back-to-back writes to alternating columns and banks. For example, for DQ[0] for the DDR3 protocol, the guaranteed write performs the following operations:

a. Writes a full burst of zeros to bank 0, column 0
b. Writes a full burst of zeros to bank 0, column 1
c. Writes a full burst of ones to bank 3, column 0
d. Writes a full burst of ones to bank 3, column 1

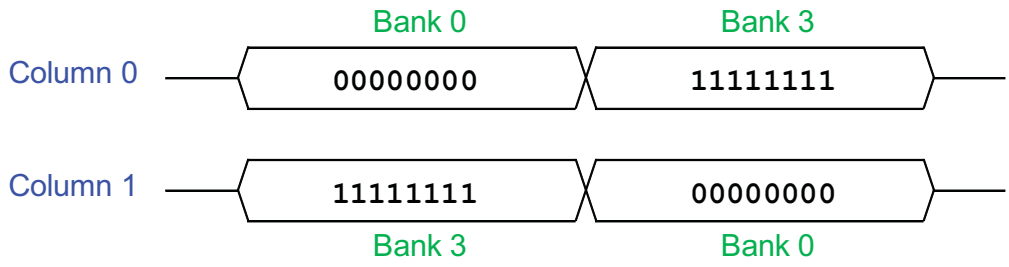
(Different protocols may use different combinations of banks and columns.)

The guaranteed write is followed by back-to-back read operations at alternating banks, effectively producing a stream of zeros followed by a stream of ones, or vice versa. The sequencer uses the zero-to-one and one-to-zero transitions in between the two bursts to identify a correct read operation, as shown in Figure 1–26.

Although the approach described above for pin DQ[0] would work by writing the same pattern to all DQ pins, it is more effective and robust to write (and read) alternating ones and zeros to alternating DQ bits. The value of the DQ bit is still constant across the burst, and the back-to-back read mechanism works exactly as described above, except that odd DQ bits have ones instead of zeros, or vice versa.

The guaranteed write does not ensure a correct DQS-to-memory clock alignment at the memory device—DQS-to-memory clock alignment is performed later, in stage 2 of the calibration process. However, the process of guaranteed write followed by read calibration is repeated several times for different DQS-to-memory clock alignments, to ensure at least one correct alignment is found.

Figure 1–26. Back to Back Reads on pin DQ[0]



DQS Enable Calibration

The full DQS enable calibration is applicable only for DDR2 and DDR3 protocols; QDR II and RLDRAM protocols use only the VFIFO-based cycle-level calibration, described below.

Delay and phase values used in this section are examples, for illustrative purposes. Your exact values may vary depending on device and configuration.

DQS enable calibration ensures reliable capture of the DQ signal without glitches on the DQS line. At this point LFIFO is set to its maximum value to guarantee a reliable read from read capture registers to the core. Read latency is minimized later.

DQS enable calibration controls the timing of the enable signal using 3 independent controls: a cycle-based control (the VFIFO), a phase control, and a delay control. The VFIFO selects the cycle by shifting the controller-generated read data enable signal, `rdata_en`, by a number of full-rate clock cycles. The phase is controlled using the DLL, while the delays are adjusted using a sequence of individual delay taps. The resolution of the phase and delay controls varies with family and configuration, but is approximately 45° for the phase, and between 10 and 50 picoseconds for the delays.

The sequencer finds the two edges of the DQS enable window by searching the space of cycles, phases, and delays (an exhaustive search can usually be avoided by initially assuming the window is at least one phase wide). During the search, to test the current settings, the sequencer issues back-to-back reads from column 0 of bank 0 and bank 3, and column 1 of bank 0 and bank 3, as shown in Figure 1–26. Two full bursts are read and compared with the reference data for each phase and delay setting.

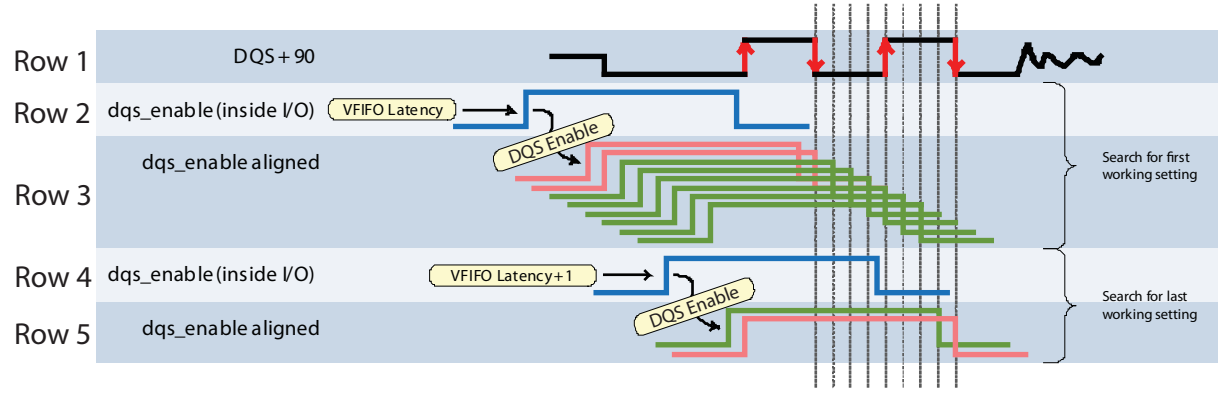
Once the sequencer identifies the two edges of the window, it center-aligns the falling edge of the DQS enable signal within the window. At this point, per-bit deskew has not yet been performed, therefore not all bits are expected to pass the read test; however, for read calibration to succeed, at least one bit per group must pass the read test.

Figure 1–27 shows the DQS and DQS enable signal relationship. The goal of DQS enable calibration is to find settings that satisfy the following conditions:

- The DQS enable signal rises before the first rising edge of DQS.
- The DQS enable signal is at one after the second-last falling edge of DQS.
- The DQS enable signal falls before the last falling edge of DQS.

The ideal position for the falling edge of the DQS enable signal is centered between the second-last and last falling edges of DQS.

Figure 1–27. DQS and DQS Enable Signal Relationships



The following points describe each row of Figure 1–27:

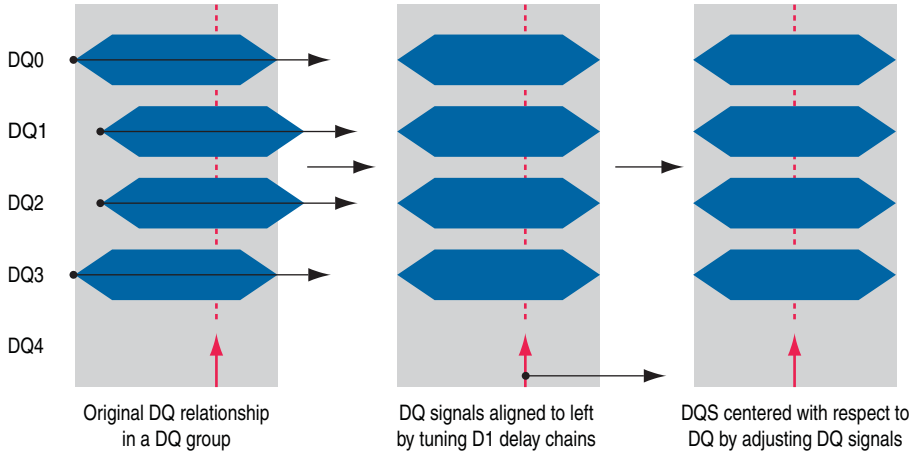
- Row 1 shows the DQS signal shifted by 90° to center-align it to the DQ data.
- Row 2 shows the raw DQS enable signal from the VFIFO.
- Row 3 shows the effect of sweeping DQS enable phases. The first two settings (shown in red) fail to properly gate the DQS signal because the enable signal turns off before the second-last falling edge of DQS. The next six settings (shown in green) gate the DQS signal successfully, with the DQS signal covering DQS from the first rising edge to the second-last falling edge.
- Row 4 shows the raw DQS enable signal from the VFIFO, increased by one clock cycle relative to Row 2.
- Row 5 shows the effect of sweeping DQS enable, beginning from the initial DQS enable of Row 4. The first setting (shown in green) successfully gates DQS, with the signal covering DQS from the first rising edge to the second-last falling edge. The second signal (shown in red), does not gate DQS successfully because the enable signal extends past the last falling edge of DQS. Any further adjustment would show the same failure.

Centering DQ/DQS

The centering DQ/DQS stage attempts to align DQ and DQS signals on reads within a group. Each DQ signal within a DQS group might be skewed and consequently arrive at the FPGA at a different time. At this point, the sequencer sweeps each DQ signal in a DQ group to align them, by adjusting DQ input delay chains (D1).

Figure 1–28 illustrates a four DQ/DQS group per-bit-deskew and centering.

Figure 1–28. Per-bit Deskew



To align and center DQ and DQS, the sequencer finds the right edge of DQ signals with respect to DQS by sweeping DQ signals within a DQ group to the right until a failure occurs. In Figure 1–28, DQ0 and DQ3 fail after six taps to the right; DQ1 and DQ2 fail after 5 taps to the right. To align the DQ signals, DQ0 and DQ3 are shifted to the right by 1 tap.

To find the center of DVW, the DQS signal is shifted to the right until a failure occurs. In Figure 1–28, a failure occurs after 3 taps, meaning that there are 5 taps to the right edge and 3 taps to the left edge. To center-align DQ and DQS, the sequencer shifts the aligned DQ signal by 1 more tap to the right.

The sequencer does not adjust DQS directly; instead, the sequencer center-aligns DQS with respect to DQ by delaying the DQ signals.

Stage 2: Write Calibration Part One

The objectives of the write calibration stage are to align DQS to the memory clock at each memory device, and to compensate for address, command, and memory clock skew at each memory device. This stage is important because the address, command, and clock signals for each memory component arrive at different times.

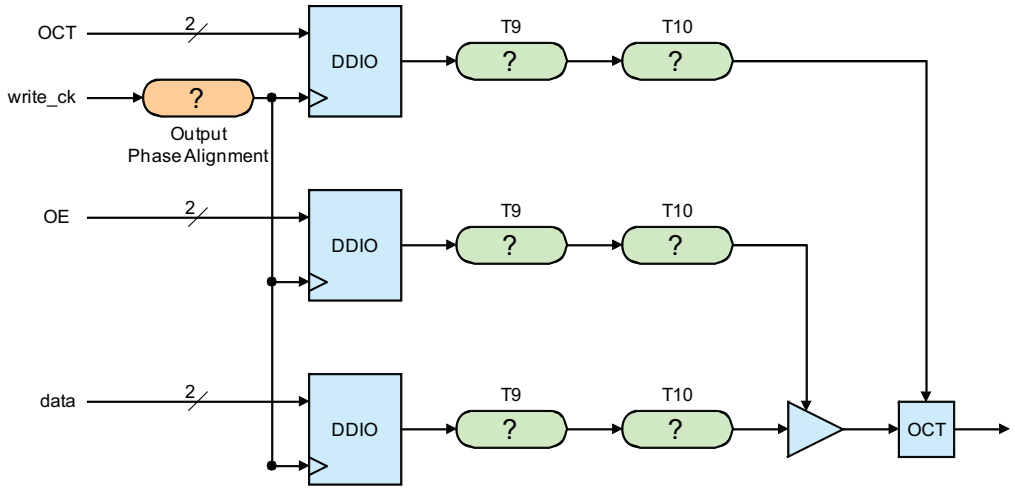
This stage applies only to DDR2, DDR3, LPDDR2, and RLDRAM II protocols; it does not apply to the QDR II and QDR II+ protocols.

Memory clock signals and DQ/DM and DQS signals have specific relationships mandated by the memory device. The PHY must ensure that these relationships are met by skewing DQ/DM and DQS signals. The relationships between DQ/DM and DQS and memory clock signals must meet the tDQSS, tDSS, and tDSH timing constraints.

The sequencer calibrates the write data path using a variety of random burst patterns to compensate for the jitter on the output data path. Simple write patterns are insufficient to ensure a reliable write operation because they might cause imprecise DQS-to-CK alignments, depending on the actual capture circuitry on a memory device. The write patterns in the write leveling stage have a burst length of 8, and are generated by a linear feedback shift register in the form of a pseudo-random binary sequence.

The write data path architecture is the same for DQ, DM, and DQS pins. Figure 1–29 illustrates the write data path for a DQ signal. The phase coming out of the Output Phase Alignment block can be set to different values to center-align DQS with respect to DQ, and it is the same for data, OE, and OCT of a given output.

Figure 1–29. Write Data Path



In write leveling, the sequencer performs write operations with different delay and phase settings, followed by a read. The sequencer can implement any phase shift between 0° and 720° (depending on device and configuration). The sequencer uses the Output Phase Alignment for coarse delays and D5 and D6 for fine delays; D5 has 15 taps of 50 ps each, and D6 has 7 taps of 50 ps each. The DQS signal phase is held at +90° with respect to DQ signal phase (Stratix IV example).

Coarse delays are called *phases*, and fine delays are called *delays*; phases are process, voltage, and temperature (PVT) compensated, delays are not (depending on family).

Delay and phase values used in this section are examples, for illustrative purposes. Your exact values may vary depending on device and configuration.

The sequencer writes and reads back several burst-length-8 patterns. Because the sequencer has not performed per-bit deskew on the write data path, not all bits are expected to pass the write test. However, for write calibration to succeed, at least one bit per group must pass the write test. The test begins by shifting the DQ/DQS phase until the first write operation completes successfully. The DQ/DQS signals are then delayed to the left by D5 and D6 to find the left edge for that working phase. Then DQ/DQS phase continues the shift to find the last working phase. For the last working phase, DQ/DQS is delayed in 50 ps steps to find the right edge of the last working phase.

The sequencer sweeps through all possible phase and delay settings for each DQ group where the data read back is correct, to define a window within which the PHY can reliably perform write operations. The sequencer picks the closest value to the center of that window as the phase/delay setting for the write data path.

Stage 3: Write Calibration Part Two—DQ/DQS Centering

The process of DQ/DQS centering in write calibration is similar to that performed in read calibration, except that write calibration is performed on the output path, using D5 and D6 delay chains.

Stage 4: Read Calibration Part Two—Read Latency Minimization

At this stage of calibration the sequencer adjusts LFIFO latency to determine the minimum read latency that guarantees correct reads.

Read Latency Tuning

In general, DQ signals from different DQ groups may arrive at the FPGA in a staggered fashion. In a DIMM or multiple memory device system, the DQ/DQS signals from the first memory device arrive sooner, while the DQ/DQS signals from the last memory device arrive the latest at the FPGA.

LFIFO transfers data from the capture registers in IOE to the core and aligns read data to the AFI clock. Up to this point in the calibration process, the read latency has been a maximum value set initially by LFIFO; now, the sequencer progressively lowers the read latency until the data can no longer be transferred reliably. The sequencer then increases the latency by one cycle to return to a working value and adds an additional cycle of margin to assure reliable reads.

Calibration Signals

Table 1–19 lists signals produced by the calibration process.

Table 1–19. Calibration Signals

Signal	Description
afi_cal_fail	Asserts high if calibration fails.
afi_cal_success	Asserts high if calibration is successful.

Calibration Time

The time needed for calibration varies, depending on many factors including the interface width, the number of ranks, frequency, board layout, and difficulty of calibration. In general, designs using the Nios II-based sequencer will take longer to calibrate than designs using the RTL-based sequencer.

Table 1–20 lists approximate typical and maximum calibration times for various protocols.

Table 1–20. Approximate Calibration Times

Protocol	Typical	Maximum
DDR2, DDR3, LPDDR2, RLDRAM 3	50-250 ms	Can take several minutes if the interface is difficult to calibrate, or if calibration initially fails and exhausts multiple retries.
QDR II/II+, RLDRAM II (with Nios II-based sequencer)	50-100 ms	Can take several minutes if the interface is difficult to calibrate, or if calibration initially fails and exhausts multiple retries.
QDR II/II+, RLDRAM II (with RTL-based sequencer)	<5 ms	<5 ms

Document Revision History

Table 1–21 lists the revision history for this document.

Table 1–21. Document Revision History

Date	Version	Changes
November 2012	3.1	■ Moved Controller Register Map to Functional Description—HPC II Controller chapter. ■ Updated Sequencer States information in Table 1–2. ■ Enhanced Using a Custom Controller information. ■ Enhanced Tracking Manager information. ■ Added Ping Pong PHY information. ■ Added RLDRAM 3 support. ■ Added LRDIMM support. ■ Added Arria V GZ support.
June 2012	3.0	■ Added Shadow Registers section. ■ Added LPDDR2 support. ■ Added new AFI signals. ■ Added Calibration Time section. ■ Added Feedback icon.
November 2011	2.1	■ Consolidated UniPHY information from 11.0 **DDR2 and DDR3 SDRAM Controller with UniPHY User Guide**, **QDR II and QDR II+ SRAM Controller with UniPHY User Guide**, and **RLDRAM II Controller with UniPHY IP User Guide**. ■ Revised Reset and Clock Generation and Dedicated Clock Networks sections. ■ Revised Figure 1–3 and Figure 1–5. ■ Added Tracking Manager to Sequencer section. ■ Revised Interfaces section for DLL, PLL, and OCT sharing interfaces. ■ Revised Using a Custom Controller section. ■ Added UniPHY Calibration Stages section; reordered stages 3 and 4, removed stage 5.

2. Functional Description—ALTMEMPHY

EMI_RM_002-3.3

The ALTMEMPHY megafunction creates the datapath between the memory device and the memory controller, and user logic in various Altera devices. The ALTMEMPHY megafunction GUI helps you configure multiple variations of a memory interface. You can then connect the ALTMEMPHY megafunction variation with either a user-designed controller or with the Altera high-performance controller. In addition, the ALTMEMPHY megafunction and the Altera high-performance controller are available for half-rate DDR3 SDRAM interfaces.

If the ALTMEMPHY megafunction does not meet your requirements, you can also create your own memory interface datapath using the ALTDLL and ALTDQ_DQS megafunctions, available in the Quartus II software. However, you are then responsible for every aspect of the interface, including timing analysis and debugging.

This chapter describes the DDR3 SDRAM ALTMEMPHY megafunction, which uses AFI as the interface between the PHY and the controller.

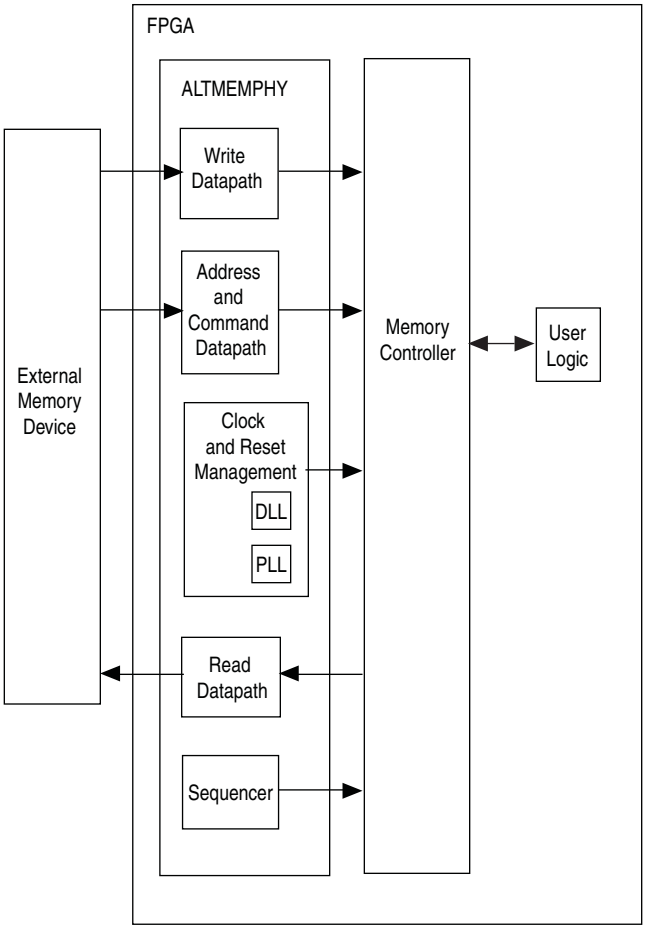
© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Block Description

Figure 2–1 shows the major blocks of the ALTMEMPHY megafunction and how it interfaces with the external memory device and the controller. The ALTPLL megafunction is instantiated inside the ALTMEMPHY megafunction, so that you do not need to generate the clock to any of the ALTMEMPHY blocks.

Figure 2–1. ALTMEMPHY Megafunction Interfacing with the Controller and the External Memory



The ALTMEMPHY megafunction comprises the following blocks:

- Write datapath
- Address and command datapath
- Clock and reset management, including DLL and PLL
- Sequencer for calibration
- Read datapath

The major advantage of the ALTMEMPHY megafunction is that it supports an initial calibration sequence to remove process variations in both the Altera device and the memory device. In Arria series devices, the DDR3 SDRAM ALTMEMPHY calibration process centers the resynchronization clock phase into the middle of the captured data valid window to maximize the resynchronization setup and hold margin. During the user operation, the VT tracking mechanism eliminates the effects of VT variations on resynchronization timing margin.

Calibration

The sequencer performs calibration to find the optimal clock phase for the memory interface.

For information about calibration, refer to ALTMEMPHY Calibration Stages.

Address and Command Datapath

This topic discusses the address and command datapath.

Arria II GX Devices

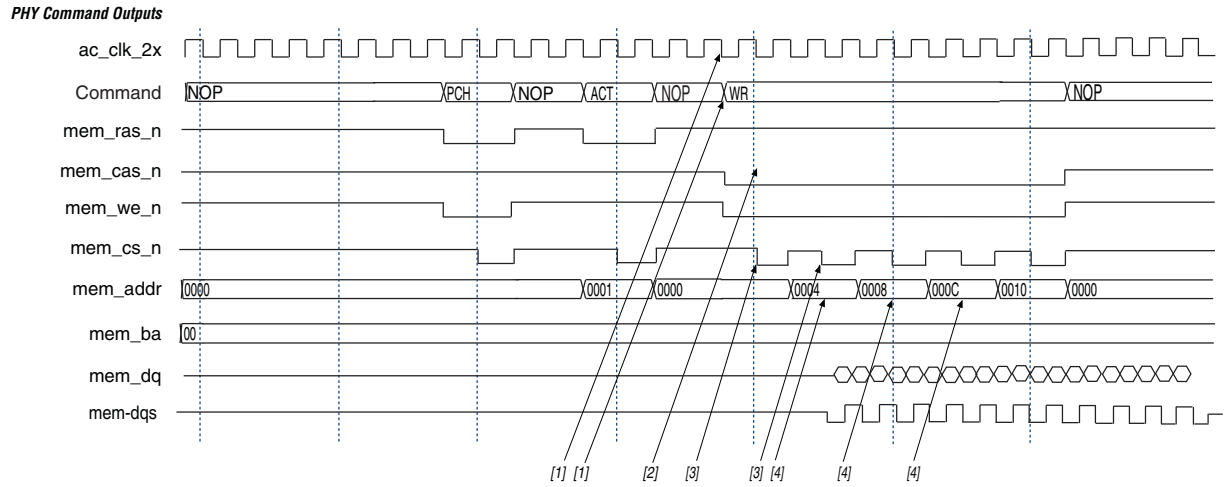
The address and command datapath is responsible for taking the address and command outputs from the controller and converting them from half-rate clock to full-rate clock. Two types of addressing are possible:

- 1T (full rate)—the duration of the address and command is a single memory clock cycle (`mem_clk_2x`, Figure 2–2). This applies to all address and command signals in full-rate designs or `mem_cs_n`, `mem_cke`, and `mem_odt` signals in half-rate designs.
- 2T (half rate)—the duration of the address and command is two memory clock cycles. For half-rate designs, the ALTMEMPHY megafunction supports only a burst size of four, which means the burst size on the local interface is always set to 1. The size of the data is 4n-bits wide on the local side and is *n*-bits wide on the memory side. To transfer all the 4*n*-bits at the double data rate, two memory-clock cycles are required. The new address and command can be issued to memory every two clock cycles. This scheme applies to all address and command signals, except for `mem_cs_n`, `mem_cke`, and `mem_odt` signals in half-rate mode.

Refer to Table 2–1 on page 2–6 to see the frequency relationship of `mem_clk_2x` with the rest of the clocks.

Figure 2–2 shows a 1T chip select signal (mem_cs_n), which is active low, and disables the command in the memory device. All commands are masked when the chip-select signal is inactive. The mem_cs_n signal is considered part of the command code.

Figure 2–2. Arria II GX Address and Command Datapath



The command interface is made up of the signals mem_ras_n, mem_cas_n, mem_we_n, mem_cs_n, mem_cke, and mem_odt.

The waveform in Figure 2–2 shows a NOP command followed by five back-to-back write commands. The following sequence corresponds with the numbered items in Figure 2–2.

1. The commands are asserted either on the rising edge of ac_clk_2x. The ac_clk_2x is derived from either mem_clk_2x (0°), write_clk_2x (270°), or the inverted variations of those two clocks (for 180° and 90° phase shifts). This depends on the setting of the address and command clock in the ALTMEMPHY parameter editor. Refer to "Address and Command Datapath" on page 2–3 for illustrations of this clock in relation to the mem_clk_2x or write_clk_2x signals.
2. All address and command signals (except for mem_cs_ns, mem_cke, and mem_odt signals) remain asserted on the bus for two clock cycles, allowing sufficient time for the signals to settle.
3. The mem_cs_n, mem_cke, and mem_odt signals are asserted during the second cycle of the address/command phase. By asserting the chip-select signal in alternative cycles, back-to-back read or write commands can be issued.
4. The address is incremented every other ac_clk_2x cycle.

The ac_clk_2x clock is derived from either mem_clk_2x (when you choose 0° or 180° phase shift) or write_clk_2x (when you choose 90° or 270° phase shift).

The address and command clock can be 0, 90, 180, or 270° from the system clock.

Clock and Reset Management

The clocking and reset block is responsible for clock generation, reset management, and phase shifting of clocks. It also has control of clock network types that route the clocks, which is handled in the <*variation_name*>`_alt_mem_phy_clk_reset` module in the <*variation_name*>_alt_mem_phy.v/.vhd file.

Clock Management

The clock management feature allows the ALTMEMPHY megafunction to work out the optimum phase during calibration, and to track voltage and temperature variation relies on phase shifting the clocks relative to each other.

Certain clocks require phase shifting during the ALTMEMPHY megafunction operation.

You can implement clock management circuitry using PLLs and DLLs.

The ALTMEMPHY MegaWizard Plug-In Manager automatically generates an ALTPLL megafunction instance. The ALTPLL megafunction generates the different clock frequencies and relevant phases used within the ALTMEMPHY megafunction.

The available device families have different PLL capabilities. The minimum PHY requirement is to have 16 phases of the highest frequency clock. The PLL uses **With No Compensation** option to minimize jitter. Changing the PLL compensation to a different operation mode may result in inaccurate timing results.

The input clock to the PLL does not have any other fan-out to the PHY, so you do not have to use a global clock resource for the path between the clock input pin to the PLL. You must use the PLL located in the same device quadrant or side as the memory interface and the corresponding clock input pin for that PLL, to ensure optimal performance and accurate timing results from the Quartus II software.

You must choose a PLL and PLL input clock pin that are located on the same side of the device as the memory interface to ensure minimal jitter. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset (by driving the `global_reset_n` signal low) and relock the PLL to ensure that the phase relationship between all PLL outputs is properly set.

If the design cascades PLLs, the source (upstream) PLL should have a low-bandwidth setting, and the destination (downstream) PLL should have a high-bandwidth setting. Adjacent PLLs cascading is recommended to reduce clock jitter.

For more information about the VCO frequency range and the available phase shifts, refer to the *Clock Networks and PLLs* chapter in the respective device family handbook.

Table 2–1 shows the clock outputs that Arria II GX devices use.

Table 2–1. DDR3 SDRAM Clocking in Arria II GX Devices (Part 1 of 2)

Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type		Notes
				All Quadrants	Any 3 Quadrants (2)	
phy_clk_1x and aux_half_rate_clk	C0	0°	Half-Rate	Global	Global	The only clocks parameterizable for the ALTMEMPHY megafunction. These clocks also feed into a divider circuit to provide the PLL scan_clk signal (for reconfiguration) that must be lower than 100 MHz.
mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	Regional (3) Global (4)	This clock is for clocking DQS and as a reference clock for the memory devices.
mem_clk_1x	C2	0°	Half-Rate	Global	Regional	This clock is for clocking DQS and as a reference clock for the memory devices.
write_clk_2x	C3	–90°	Full-Rate	Global	Regional	This clock is for clocking the data out of the DDR I/O (DDIO) pins in advance of the DQS strobe (or equivalent). As a result, its phase leads that of the mem_clk_2x by 90°.
ac_clk_2x	C3	–90°	Full-Rate	Global	Regional	Address and command clock. The ac_clk_2x clock is derived from either mem_clk_2x (when you choose 0° or 180° phase shift) or write_clk_2x (when you choose 90° or 270° phase shift). Refer to "Address and Command Datapath" on page 2–3 for illustrations of the address and command clock relationship with the mem_clk_2x or write_clk_2x signals.
cs_n_clk_2x	C3	–90°	Full-Rate	Global	Global	Memory chip-select clock. The cs_n_clk_2x clock is derived from ac_clk_2x.
resync_clk_2x	C4	Calibrated	Full-Rate	Global	Regional	Clocks the resynchronization registers after the capture registers. Its phase is adjusted to the center of the data valid window across all the DQS-clocked DDIO groups.

Table 2–1. DDR3 SDRAM Clocking in Arria II GX Devices (Part 2 of 2)

Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type		Notes
				All Quadrants	Any 3 Quadrants (2)	
measure_clk_2x	C5	Calibrated	Full-Rate	Global	Regional	This clock is for VT tracking. This free-running clock measures relative phase shifts between the internal clock(s) and those being fed back through a mimic path. As a result, the ALTMEMPHY megafunction can track VT effects on the FPGA and compensate for the effects.

Note to Table 2–1:

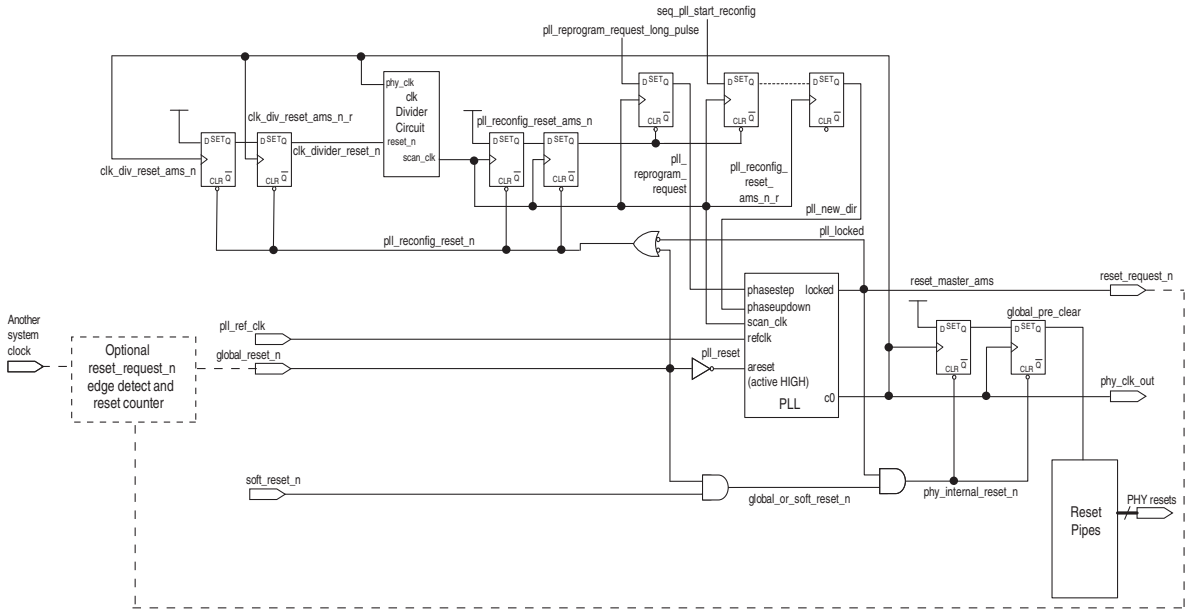
(1) The _1x clock represents a frequency that is half of the memory clock frequency; the _2x clock represents the memory clock frequency.
(2) The default clock network type is Global, however you can specify a regional clock network to improve clock jitter if your design uses any three quadrants.
(3) For mem_clk2x.
(4) For aux_full_rate_clk.

Reset Management

Figure 2–3 shows the main features of the reset management block for the DDR3 SDRAM PHY. You can use the pll_ref_clk input to feed the optional reset_request_n edge detect and reset counter module. However, this requires the pll_ref_clk signal to use a global clock network resource.

There is a unique reset metastability protection circuit for the clock divider circuit because the phy_clk domain reset metastability protection registers have fan-in from the soft_reset_n input so these registers cannot be used.

Figure 2–3. ALTMEMPHY Reset Management Block for Arria II GX Devices



Read Datapath

This topic discusses the read datapath.

Arria II GX Devices

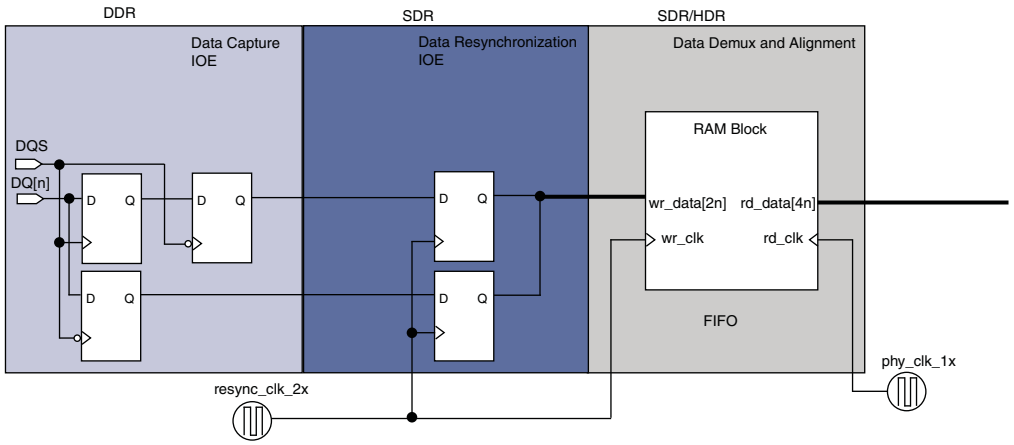
The read datapath logic captures data sent by the memory device and subsequently aligns the data back to the system clock domain. The read datapath for DDR3 SDRAM consists of the following three main blocks:

- Data capture
- Data resynchronization
- Data demultiplexing and alignment

As the DQS/DQSn signal is not continuous, the PHY also has postamble protection logic to ensure that any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches.

Figure 2–4 shows the order of the functions performed by the read datapath and the frequency at which the read data is handled.

Figure 2–4. DDR3 SDRAM Read Datapath in Arria II GX Devices



Data Capture and Resynchronization

The data capture and resynchronization registers for Arria II GX devices are implemented in the I/O element (IOE) to achieve maximum performance. Data capture and resynchronization is the process of capturing the read data (DQ) with the DQS/DQSn strobes and resynchronizing the captured data to an internal free-running full-rate clock supplied by the enhanced PLL. The resynchronization clock is an intermediate clock whose phase shift is determined during the calibration stage. The captured data (`rdata_p_captured` and `rdata_n_captured`) is synchronized to the resynchronization clock (`resync_clk_2x`), refer to Figure 2–4. For Arria II GX devices, the ALTMEMPHY instances an ALTDQ_DQS megafunction that instantiates the required IOEs for all the DQ and DQS pins.

Data Demultiplexing

Data demultiplexing is the process of changing the SDR data into HDR data. Data demultiplexing is required to bring the frequency of the resynchronized data down to the frequency of the system clock, so that data from the external memory device can ultimately be brought into the FPGA controller clock domain. Before data capture, the data is DDR and *n*-bit wide. After data capture, the data is SDR and 2*n*-bit wide. After data demuxing, the data is HDR of width 4*n*-bits wide. The system clock frequency is half the frequency of the memory clock. Demultiplexing is achieved using a dual-port memory with a 2*n*-bit wide write-port operating on the resynchronization clock (SDR) and a 4*n*-bit wide read-port operating on the PHY clock (HDR). The basic principle of operation is that data is written to the memory at the SDR rate and read from the memory at the HDR rate while incrementing the read- and write-address pointers. As the SDR and HDR clocks are generated, the read and write pointers are continuously incremented by the same PLL, and the 4*n*-bit wide read data follows the 2*n*-bit wide write data with a constant latency

Read Data Alignment

Data alignment is the process controlled by the sequencer to ensure the correct captured read data is present in the same half-rate clock cycle at the output of the read data DPRAM. Data alignment is implemented using memory blocks in the core of devices.

Postamble Protection

A dedicated postamble register controls the gating of the shifted DQS signal that clocks the DQ input registers at the end of a read operation. Any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches. The postamble path is also calibrated to determine the correct clock cycle, clock phase shift, and delay chain settings.

ALTMEMPHY Signals

This section describes the ALMEMPHY megafunction signals for DDR3 SDRAM variants.

Table 2–2 through Table 2–4 show the signals.

Signals with the prefix `mem_` connect the PHY with the memory device; ports with the prefix `ctl_` connect the PHY with the controller.

The signal lists include the following signal groups:

- I/O interface to the SDRAM devices
- Clocks and resets
- External DLL signals
- User-mode calibration OCT control
- Write data interface
- Read data interface
- Address and command interface
- Calibration control and status interface
- Debug interface

Table 2–2. Interface to the DDR3 SDRAM Devices (Part 1 of 2) *(1)*

Signal Name	Type	Width *(2)*	Description
mem_addr	Output	MEM_IF_ROWADDR_WIDTH	The memory row and column address bus.
mem_ba	Output	MEM_IF_BANKADDR_WIDTH	The memory bank address bus.
mem_cas_n	Output	1	The memory column address strobe.
mem_cke	Output	MEM_IF_CS_WIDTH	The memory clock enable.
mem_clk	Bidirectional	MEM_IF_CLK_PAIR_COUNT	The memory clock, positive edge clock. *(3)*
mem_clk_n	Bidirectional	MEM_IF_CLK_PAIR_COUNT	The memory clock, negative edge clock.
mem_cs_n	Output	MEM_IF_CS_WIDTH	The memory chip select signal.
mem_dm	Output	MEM_IF_DM_WIDTH	The optional memory DM bus.
mem_dq	Bidirectional	MEM_IF_DWIDTH	The memory bidirectional data bus.
mem_dqs	Bidirectional	MEM_IF_DWIDTH/ MEM_IF_DQ_PER_DQS	The memory bidirectional data strobe bus.
mem_dqs_n	Bidirectional	MEM_IF_DWIDTH/ MEM_IF_DQ_PER_DQS	The memory bidirectional data strobe bus.
mem_odt	Output	MEM_IF_CS_WIDTH	The memory on-die termination control signal.

Table 2–2. Interface to the DDR3 SDRAM Devices (Part 2 of 2) *(1)*

Signal Name	Type	Width *(2)*	Description
mem_ras_n	Output	1	The memory row address strobe.
mem_reset_n	Output	1	The memory reset signal. This signal is derived from the PHY's internal reset signal, which is generated by gating the global reset, soft reset, and the PLL locked signal.
mem_we_n	Output	1	The memory write enable signal.
mem_ac_parity *(4)*	Output	1	The address or command parity signal generated by the PHY and sent to the DIMM.
parity_error_n *(4)*	Output	1	The active-low signal that is asserted when a parity error occurs and stays asserted until the PHY is reset.
mem_err_out_n *(4)*	Input	1	The signal sent from the DIMM to the PHY to indicate that a parity error has occured for a particular cycle.

Notes to Table 2–2:

(1) Connected to I/O pads.
(2) Refer to Table 2–5 for parameter description.
(3) Output is for memory device, and input path is fed back to ALTMEMPHY megafunction for VT tracking.
(4) This signal is for Registered DIMMs only.

Table 2–3. AFI Signals (Part 1 of 4)

Signal Name	Type	Width *(1)*	Description
Clocks and Resets			
pll_ref_clk	Input	1	The reference clock input to the PHY PLL.
global_reset_n	Input	1	Active-low global reset for PLL and all logic in the PHY. A level set reset signal, which causes a complete reset of the whole system. The PLL may maintain some state information.
soft_reset_n	Input	1	Edge detect reset input intended for SOPC Builder use or to be controlled by other system reset logic. Causes a complete reset of PHY, but not the PLL used in the PHY.
reset_request_n	Output	1	Directly connected to the locked output of the PLL and is intended for optional use either by automated tools such as SOPC Builder or could be manually ANDed with any other system-level signals and combined with any edge detect logic as required and then fed back to the global_reset_n input. Reset request output that indicates when the PLL outputs are not locked. Use this as a reset request input to any system-level reset controller you may have. This signal is always low while the PLL is locking (but not locked), and so any reset logic using it is advised to detect a reset request on a falling-edge rather than by level detection.

Table 2–3. AFI Signals (Part 2 of 4)

Signal Name	Type	Width (1)	Description
ctl_clk	Output	1	Half-rate clock supplied to controller and system logic. The same signal as the non-AFI phy_clk.
ctl_reset_n	Output	1	Reset output on ctl_clk clock domain.
Other Signals			
aux_half_rate_clk	Output	1	In half-rate designs, a copy of the phy_clk_1x signal that you can use in other parts of your design, same as phy_clk port.
aux_full_rate_clk	Output	1	In full-rate designs, a copy of the mem_clk_2x signal that you can use in other parts of your design.
aux_scan_clk	Output	1	Low frequency scan clock supplied primarily to clock any user logic that interfaces to the PLL and DLL reconfiguration interfaces.
aux_scan_clk_reset_n	Output	1	This reset output asynchronously asserts (drives low) when global_reset_n is asserted and de-assert (drives high) synchronous to aux_scan_clk when global_reset_n is de-asserted. It allows you to reset any external circuitry clocked by aux_scan_clk.
Write Data Interface			
ctl_dqs_burst	Input	MEM_IF_DQS_WIDTH × DWIDTH_RATIO / 2	When asserted, mem_dqs is driven. The ctl_dqs_burst signal must be asserted before the ctl_wdata_valid signal and must be driven for the correct duration to generate a correctly timed mem_dqs signal.
ctl_wdata_valid	Input	MEM_IF_DQS_WIDTH × DWIDTH_RATIO / 2	Write data valid. Generates ctl_wdata and ctl_dm output enables.
ctl_wdata	Input	MEM_IF_DWIDTH × DWIDTH_RATIO	Write data input from the controller to the PHY to generate mem_dq.
ctl_dm	Input	MEM_IF_DM_WIDTH × DWIDTH_RATIO	DM input from the controller to the PHY.
ctl_wlat	Output	5	Required write latency between address/command and write data that is issued to ALTMEMPHY controller local interface. This signal is only valid when the ALTMEMPHY sequencer successfully completes calibration, and does not change at any point during normal operation. The legal range of values for this signal is 0 to 31; and the typical values are between 0 and ten, 0 mostly for low CAS latency DDR memory types.

Table 2–3. AFI Signals (Part 3 of 4)

Signal Name	Type	Width (1)	Description
Read Data Interface			
ctl_doing_rd	Input	MEM_IF_DQS_WIDTH × DWIDTH_RATIO / 2	Doing read input. Indicates that the DDR3 SDRAM controller is currently performing a read operation. The controller generates ctl_doing_rd to the ALTMEMPHY megafunction. The ctl_doing_rd signal is asserted for one phy_clk cycle for every read command it issues. If there are two read commands, ctl_doing_rd is asserted for two phy_clk cycles. The ctl_doing_rd signal also enables the capture registers and generates the ctl_mem_rdata_valid signal. The ctl_doing_rd signal should be issued at the same time the read command is sent to the ALTMEMPHY megafunction.
ctl_rdata	Output	DWIDTH_RATIO × MEM_IF_DWIDTH	Read data from the PHY to the controller.
ctl_rdata_valid	Output	DWIDTH_RATIO/2	Read data valid indicating valid read data on ctl_rdata. This signal is two-bits wide (as only half-rate or DWIDTH_RATIO = 4 is supported) to allow controllers to issue reads and writes that are aligned to either the half-cycle of the half-rate clock.
ctl_rlat	Output	READ_LAT_WIDTH	Contains the number of clock cycles between the assertion of ctl_doing_rd and the return of valid read data (ctl_rdata). This signal is unused by the Altera high-performance controller.
Address and Command Interface			
ctl_addr	Input	MEM_IF_ROWADDR_WIDTH × DWIDTH_RATIO / 2	Row address from the controller.
ctl_ba	Input	MEM_IF_BANKADDR_WIDTH × DWIDTH_RATIO / 2	Bank address from the controller.
ctl_cke	Input	MEM_IF_CS_WIDTH × DWIDTH_RATIO / 2	Clock enable from the controller.
ctl_cs_n	Input	MEM_IF_CS_WIDTH ×DWIDTH_RATIO / 2	Chip select from the controller.
ctl_odt	Input	MEM_IF_CS_WIDTH × DWIDTH_RATIO / 2	On-die-termination control from the controller.
ctl_ras_n	Input	DWIDTH_RATIO / 2	Row address strobe signal from the controller.
ctl_we_n	Input	DWIDTH_RATIO / 2	Write enable.
ctl_cas_n	Input	DWIDTH_RATIO / 2	Column address strobe signal from the controller.
ctl_rst_n	Input	DWIDTH_RATIO / 2	Reset from the controller.
Calibration Control and Status Interface			
ctl_mem_clk_disable	Input	MEM_IF_CLK_PAIR_COUNT	When asserted, mem_clk and mem_clk_n are disabled.
ctl_cal_success	Output	1	A 1 indicates that calibration was successful.
ctl_cal_fail	Output	1	A 1 indicates that calibration has failed.

Table 2–3. AFI Signals (Part 4 of 4)

Signal Name	Type	Width (1)	Description
ctl_cal_req	Input	1	When asserted, a new calibration sequence is started. Currently not supported.
ctl_cal_byte_lane_sel_n	Input	MEM_IF_DQS_WIDTH × MEM_CS_WIDTH	Indicates which DQS groups should be calibrated. Not supported.

Note to Table 2–2:

(1) Refer to Table 2–5 for parameter descriptions.

Table 2–4. Other Interface Signals (Part 1 of 2)

Signal Name	Type	Width	Description
External DLL Signals			
dqs_delay_ctrl_export	Output	DQS_DELAY_CTL_WIDTH	Allows sharing DLL in this ALTMEMPHY instance with another ALTMEMPHY instance. Connect the dqs_delay_ctrl_export port on the ALTMEMPHY instance with a DLL to the dqs_delay_ctrl_import port on the other ALTMEMPHY instance.
dqs_delay_ctrl_import	Input	DQS_DELAY_CTL_WIDTH	Allows the use of DLL in another ALTMEMPHY instance in this ALTMEMPHY instance. Connect the dqs_delay_ctrl_export port on the ALTMEMPHY instance with a DLL to the dqs_delay_ctrl_import port on the other ALTMEMPHY instance.
dqs_offset_delay_ctrl_ width	Input	DQS_DELAY_CTL_WIDTH	Connects to the DQS delay logic when dll_import_export is set to IMPORT. Only connect if you are using a DLL offset, which can otherwise be tied to zero. If you are using a DLL offset, connect this input to the offset_ctrl_out output of the dll_offset_ctrl block.
dll_reference_ clk	Output	1	Reference clock to feed to an externally instantiated DLL. This clock is typically from one of the PHY PLL outputs.
User-Mode Calibration OCT Control Signals			
oct_ctl_rs_value	Input	14	OCT RS value port for use with ALT_OCT megafunction if you want to use OCT with user-mode calibration.
oct_ctl_rt_value	Input	14	OCT RT value port for use with ALT_OCT megafunction if you want to use OCT with user-mode calibration.
Debug Interface Signals (1), (2)			
dbg_clk	Input	1	Debug interface clock.
dbg_reset_n	Input	1	Debug interface reset.
dbg_addr	Input	DBG_A_WIDTH	Address input.
dgb_wr	Input	1	Write request.
dbg_rd	Input	1	Read request.
dbg_cs	Input	1	Chip select.
dbg_wr_data	Input	32	Debug interface write data.
dbg_rd_data	Output	32	Debug interface read data.
dbg_waitrequest	Output	1	Wait signal.

Table 2–4. Other Interface Signals (Part 2 of 2)

Signal Name	Type	Width	Description
Calibration Interface Signals—without leveling only			
rsu_codvw_phase	Output	—	The sequencer sweeps the phase of a resynchronization clock across 360° or 720° of a memory clock cycle. Data reads from the DIMM are performed for each phase position, and a data valid window is located, which is the set of resynchronization clock phase positions where data is successfully read. The final resynchronization clock phase is set at the center of this range: the center of the data valid window or CODVW. This output is set to the current calculated value for the CODVW, and represents how many phase steps were performed by the PLL to offset the resynchronization clock from the memory clock.
rsu_codvw_size	Output	—	The final centre of data valid window size (rsu_codvw_size) is the number of phases where data was successfully read in the calculation of the resynchronization clock centre of data valid window phase (rsu_codvw_phase).
rsu_read_latency	Output	—	The rsu_read_latency output is then set to the read latency (in phy_clk cycles) using the rsu_codvw_phase resynchronization clock phase. If calibration is unsuccessful then this signal is undefined.
rsu_no_dvw_err	Output	—	If the sequencer sweeps the resynchronization clock across every phase and does not see any valid data at any phase position, then calibration fails and this output is set to 1.
rsu_grt_one_dvw_err	Output	—	If the sequencer sweeps the resynchronization clock across every phase and sees multiple data valid windows, this is indicative of unexpected read data (random bit errors) or an incorrectly configured PLL that must be resolved. Calibration has failed and this output is set to 1.

Notes to Table 2–4:

(1) The debug interface uses the simple Avalon-MM interface protocol.

(2) These ports exist in the Quartus II software, even though the debug interface is for Altera's use only.

Table 2–5 shows the parameters to which Table 2–2 through Table 2–4 refer.

Table 2–5. Parameters (Part 1 of 2)

Parameter Name	Description
DWIDTH_RATIO	The data width ratio from the local interface to the memory interface. DWIDTH_RATIO of 2 means full rate, while DWIDTH_RATIO of 4 means half rate.
LOCAL_IF_DWIDTH	The width of the local data bus must be quadrupled for half-rate and doubled for full-rate.
MEM_IF_DWIDTH	The data width at the memory interface. MEM_IF_DWIDTH can have values that are multiples of MEM_IF_DQ_PER_DQS.
MEM_IF_DQS_WIDTH	The number of DQS pins in the interface.
MEM_IF_ROWADDR_WIDTH	The row address width of the memory device.
MEM_IF_BANKADDR_WIDTH	The bank address with the memory device.
MEM_IF_CS_WIDTH	The number of chip select pins in the interface. The sequencer only calibrates one chip select pin.
MEM_IF_DM_WIDTH	The number of mem_dm pins on the memory interface.

Table 2–5. Parameters (Part 2 of 2)

Parameter Name	Description
MEM_IF_DQ_PER_DQS	The number of mem_dq[] pins per mem_dqs pin.
MEM_IF_CLK_PAIR_COUNT	The number of mem_clk/mem_clk_n pairs in the interface.

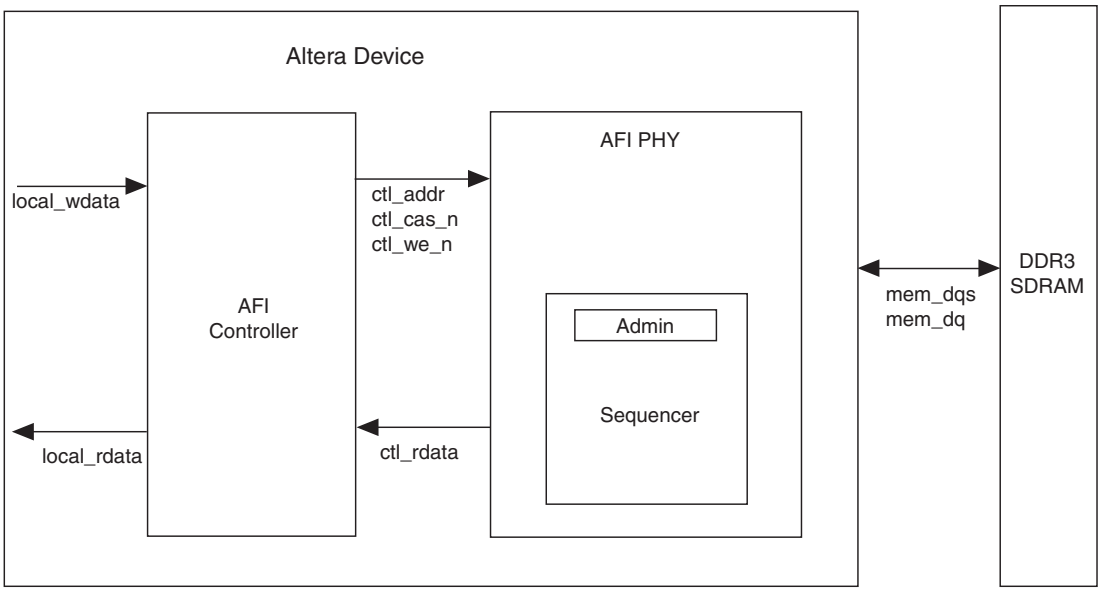
PHY-to-Controller Interfaces

The following section describes the typical modules that are connected to the ALTMEMPHY variation and the port name prefixes each module uses. This section also describes using a custom controller. This section describes the AFI.

The AFI standardizes and simplifies the interface between controller and PHY for all Altera memory designs, thus allowing you to easily interchange your own controller code with Altera's high-performance controller. The AFI PHY includes an administration block that configures the memory for calibration and performs necessary mode registers accesses to configure the memory as required (these calibration processes are different). Figure 2–5 shows an overview of the connections between the PHY, the controller, and the memory device.

Altera recommends that you use the AFI for new designs.

Figure 2–5. AFI PHY Connections



For half-rate designs, the address and command signals in the ALTMEMPHY megafunction are asserted for one mem_clk cycle (1T addressing), such that there are two input bits per address and command pin in half-rate designs. If you require a more conservative 2T addressing, drive both input bits (of the address and command signal) identically in half-rate designs.

For DDR3 SDRAM with the AFI, the read and write control signals are on a per-DQS group basis. The controller can calibrate and use a subset of the available DDR3 SDRAM devices. For example, the controller can calibrate and use two devices out of a 64- or 72-bit DIMM for better debugging mechanism.

For half-rate designs, the AFI allows the controller to issue reads and writes that are aligned to either half-cycle of the half-rate `phy_clk`, which means that the datapaths can support multiple data alignments—word-unaligned and word-aligned writes and reads. Figure 2–6 and Figure 2–7 display the half-rate write operation.

Figure 2–6. Half-Rate Write with Word-Unaligned Data

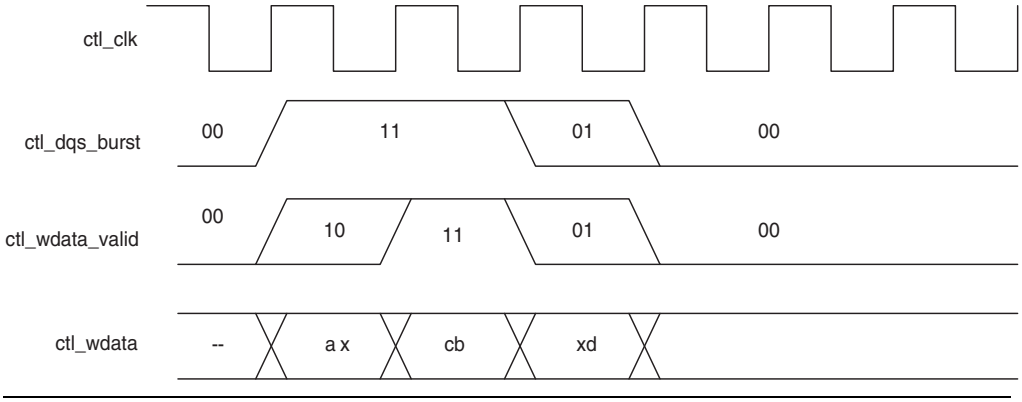
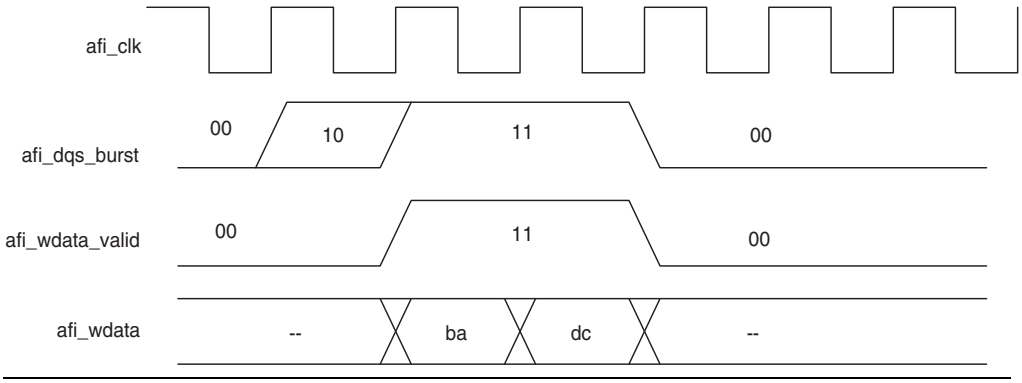


Figure 2–7. Half-Rate Write with Word-Aligned Data



After calibration process is complete, the sequencer sends the write latency in number of clock cycles to the controller.

Figure 2–8 and Figure 2–9 show word-aligned writes and reads. In the following read and write examples the data is written to and read from the same address. In each example, `ctl_rdata` and `ctl_wdata` are aligned with controller clock (`ctl_clk`) cycles. All the data in the bit vector is valid at once. For comparison, refer Figure 2–10 and Figure 2–11 that show the word-unaligned writes and reads.

The ctl_doing_rd is represented as a half-rate signal when passed into the PHY. Therefore, the lower half of this bit vector represents one memory clock cycle and the upper half the next memory clock cycle. Figure 2–11 on page 2–22 shows separated word-unaligned reads as an example of two ctl_doing_rd bits are different. Therefore, for each x16 device, at least two ctl_doing_rd bits need to be driven, and two ctl_rdata_valid bits need to be interpreted.

The AFI has the following conventions:

- With the AFI, high and low signals are combined in one signal, so for a single chip select (ctl_cs_n) interface, ctl_cs_n[1:0], where location 0 appears on the memory bus on one mem_clk cycle and location 1 on the next mem_clk cycle.

  This convention is maintained for all signals so for an 8 bit memory interface, the write data (ctl_wdata) signal is ctl_wdata[31:0], where the first data on the DQ pins is ctl_wdata[7:0], then ctl_wdata[15:8], then ctl_wdata[23:16], then ctl_wdata[31:24].

- Word-aligned and word-unaligned reads and writes have the following definitions:
  - Word-aligned for the single chip select is active (low) in location 1 (_1). ctl_cs_n[1:0] = 01 when a write occurs. This alignment is the easiest alignment to design with.
  - Word-unaligned is the opposite, so ctl_cs_n[1:0] = 10 when a read or write occurs and the other control and data signals are distributed across consecutive ctl_clk cycles.

  The Altera high-performance controller uses word-aligned data only.

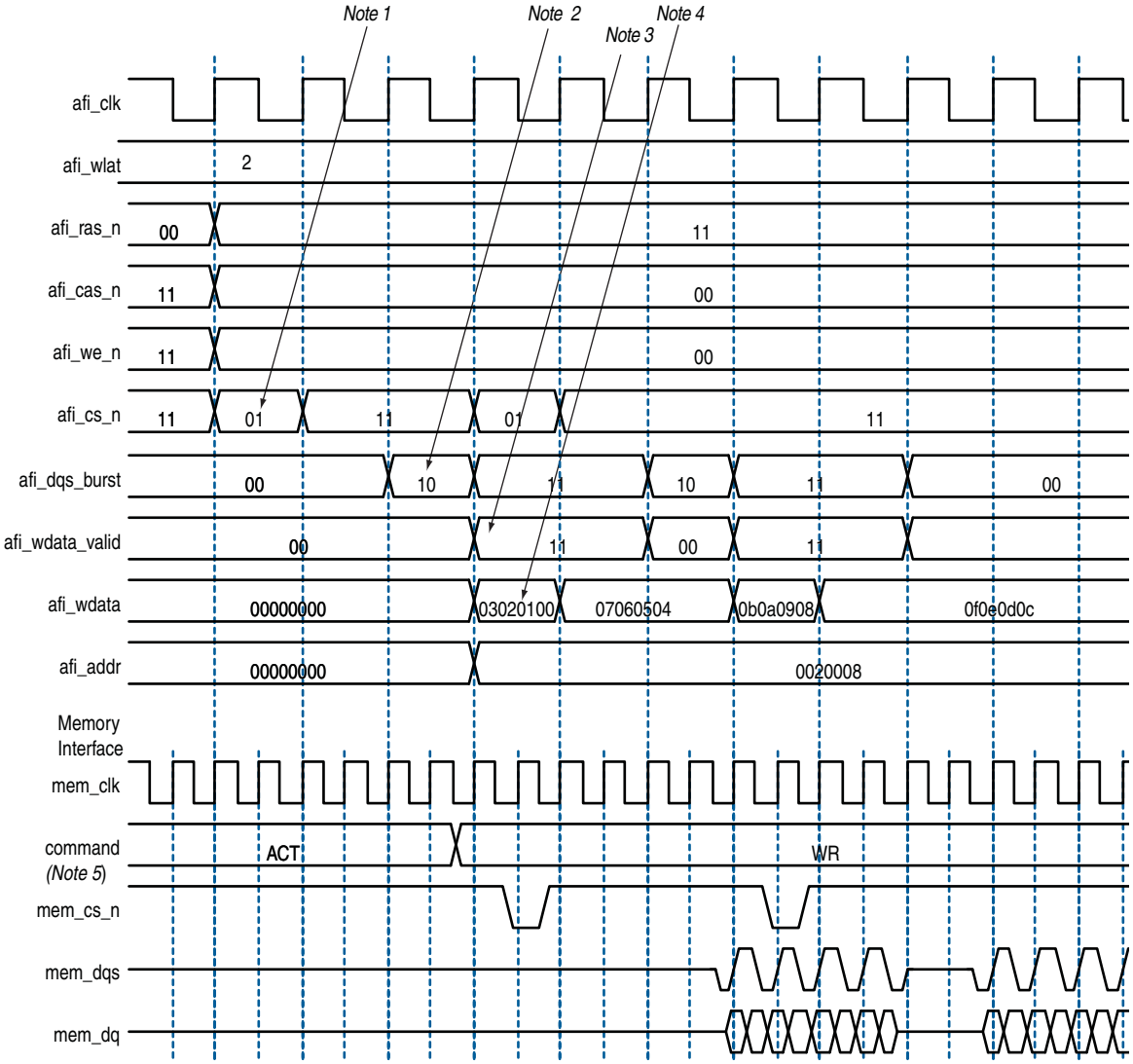
  The timing analysis script does not support word-unaligned reads and writes.

- Spaced reads and writes have the following definitions:
  - Spaced writes—write commands separated by a gap of one controller clock (ctl_clk) cycle
  - Spaced reads—read commands separated by a gap of one controller clock (ctl_clk) cycle

Figure 2–8 through Figure 2–11 assume the following general points:

- The burst length is four. A DDR2 SDRAM is used—the interface timing is identical for DDR3 devices.
- An 8-bit interface with one chip select.
- The data for one controller clock (ctl_clk) cycle represents data for two memory clock (mem_clk) cycles (half-rate interface).

Figure 2–8. Word-Aligned Writes



Notes to Figure 2–8:

(1) To show the even alignment of `ctl_cs_n`, expand the signal (this convention applies for all other signals).

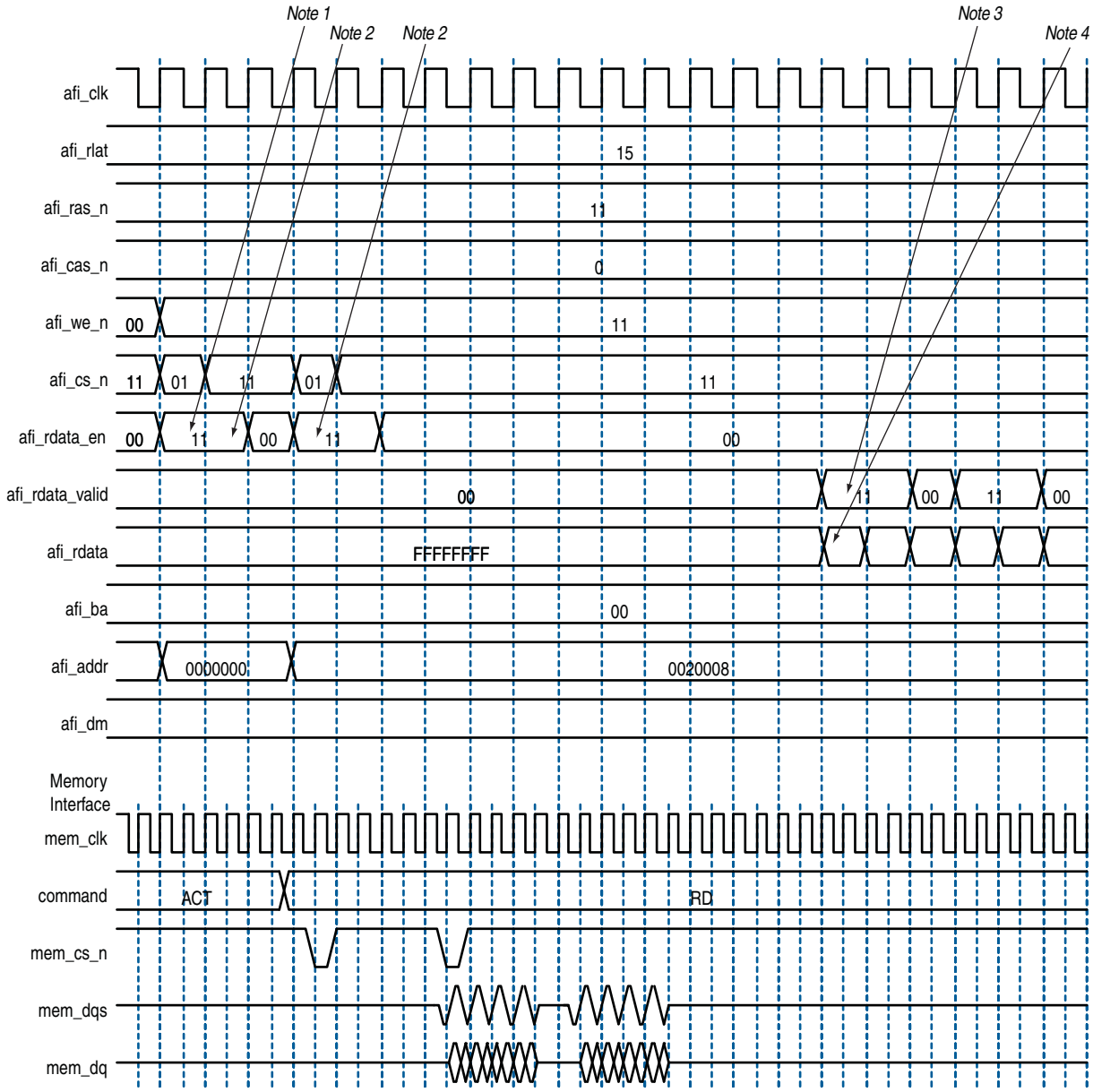
(2) The `ctl_dqs_burst` must go high one memory clock cycle before `ctl_wdata_valid`. Compare with the word-unaligned case.

(3) The `ctl_wdata_valid` is asserted two `ctl_wlat` controller clock (`ctl_clk`) cycles after chip select (`ctl_cs_n`) is asserted. The `ctl_wlat` indicates the required write latency in the system. The value is determined during calibration and is dependant upon the relative delays in the address and command path and the write datapath in both the PHY and the external DDR SDRAM subsystem. The controller must drive `ctl_cs_n` and then wait `ctl_wlat` (two in this example) `ctl_clks` before driving `ctl_wdata_valid`.

(4) Observe the ordering of write data (`ctl_wdata`). Compare this to data on the `mem_dq` signal.

(5) In all waveforms a command record is added that combines the memory pins `ras_n`, `cas_n` and `we_n` into the current command that is issued. This command is registered by the memory when chip select (`mem_cs_n`) is low. The important commands in the presented waveforms are WR = write, ACT = activate.

Figure 2–9. Word-Aligned Reads



Notes to Figure 2–9:

(1) For AFI, `ctl_doing_rd` is required to be asserted one memory clock cycle before chip select (`ctl_cs_n`) is asserted. In the half-rate `ctl_clk` domain, this requirement manifests as the controller driving `11` (as opposed to the `01`) on `ctl_doing_rd`.

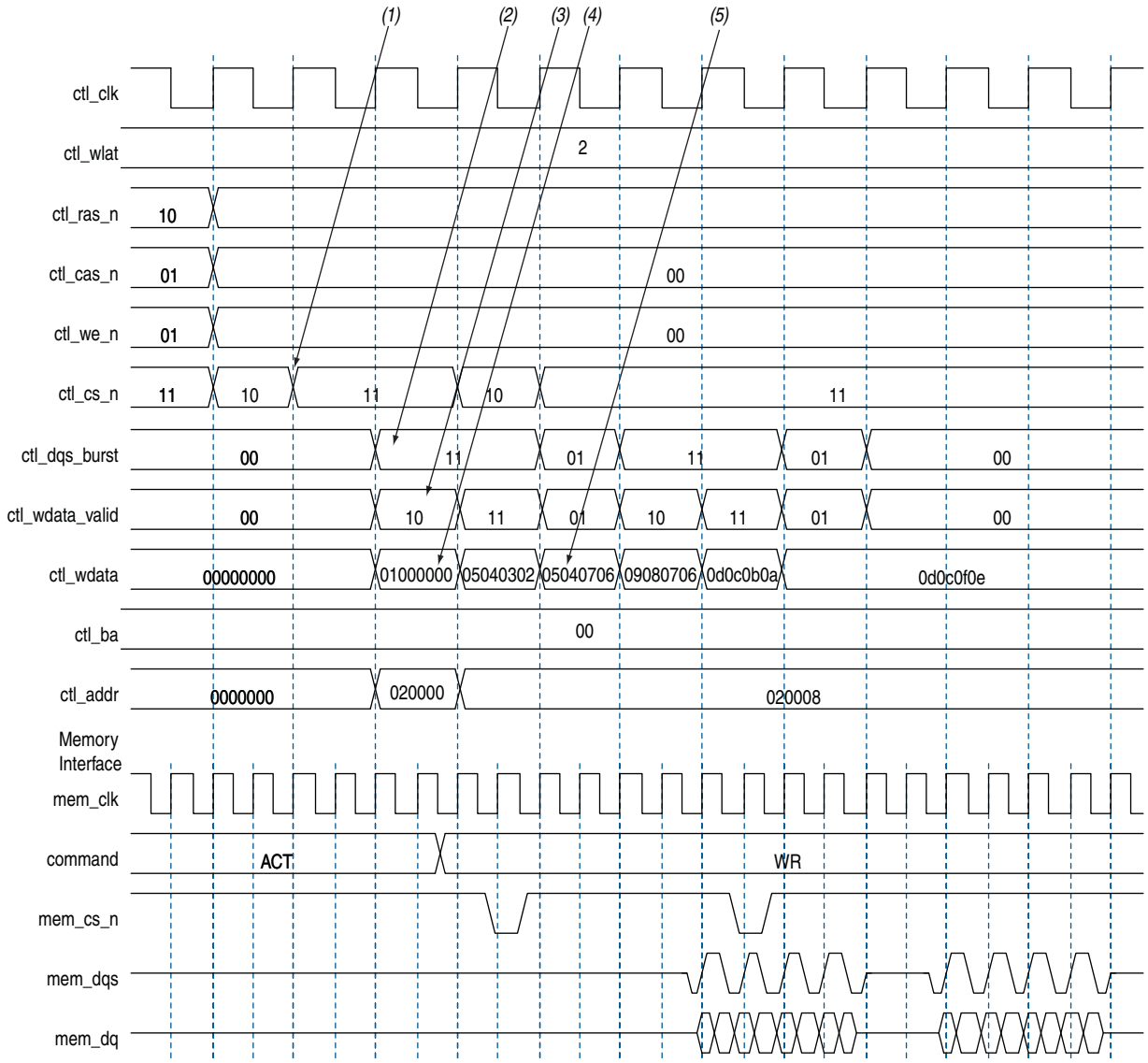
(2) AFI requires that `ctl_doing_rd` is driven for the duration of the read. In this example, it is driven to `11` for two half-rate `ctl_clks`, which equates to driving to `1`, for the four memory clock cycles of this four-beat burst.

(3) The `ctl_rdata_valid` returns 15 (`ctl_rlat`) controller clock (`ctl_clk`) cycles after `ctl_doing_rd` is asserted. Returned is when the `ctl_rdata_valid` signal is observed at the output of a register within the controller. A controller can use the `ctl_rlat` value to determine when to register to returned data, but this is unnecessary as the `ctl_rdata_valid` is provided for the controller to use as an enable when registering read data.

(4) Observe the alignment of returned read data with respect to data on the bus.

Figure 2–10 and Figure 2–11 show spaced word-unaligned writes and reads.

Figure 2–10. Word-Unaligned Writes



Notes to Figure 2–10:

(1) Alternative word-unaligned chip select (`ctl_cs_n`).

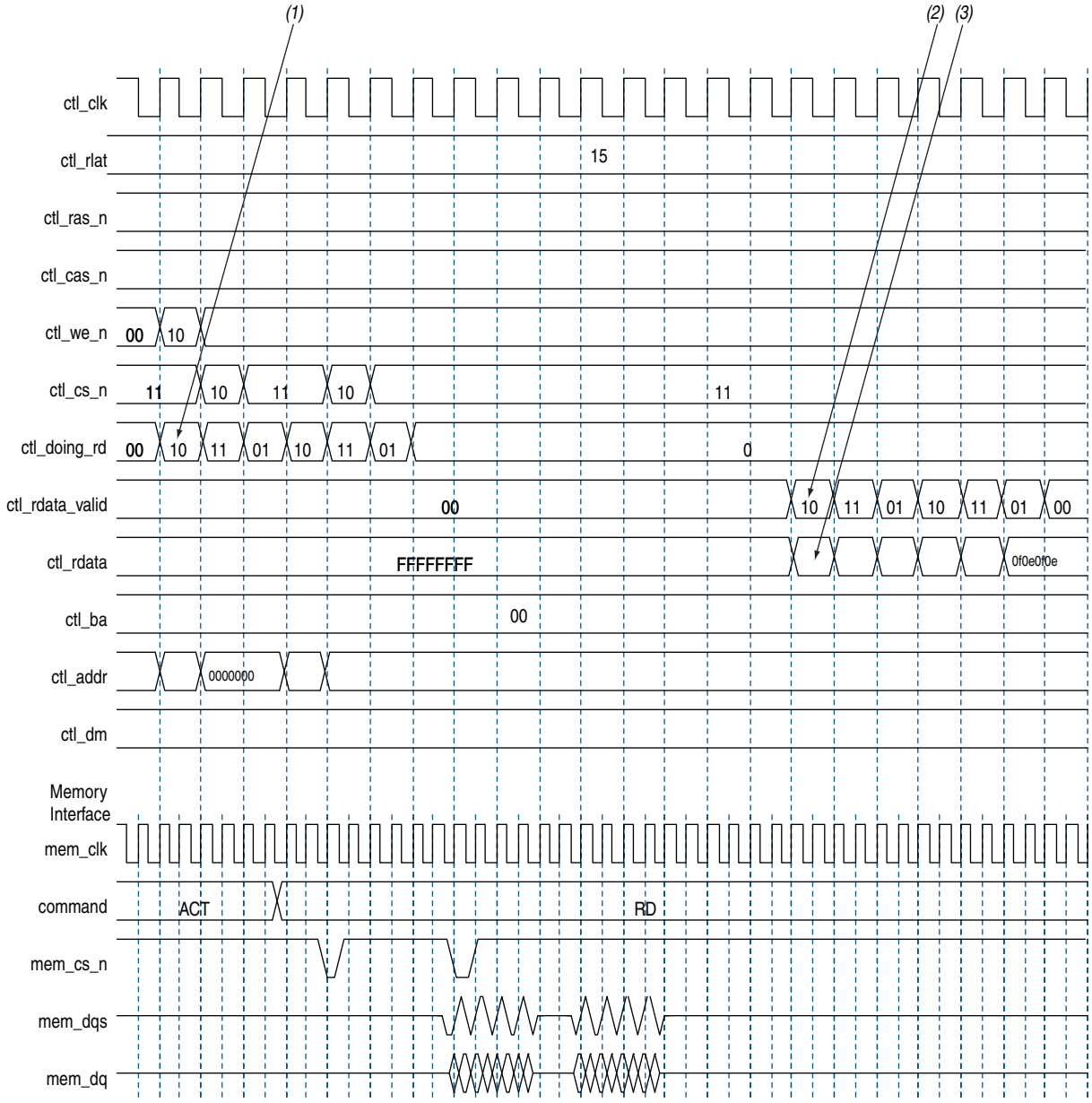
(2) As with word- aligned writes, `ctl_dqs_burst` is asserted one memory clock cycle before `ctl_wdata_valid`. You can see `ctl_dqs_burst` is 11 in the same cycle where `ctl_wdata_valid` is 10. The LSB of these two becomes the first value the signal takes in the `mem_clk` domain. You can see that `ctl_dqs_burst` has the necessary one `mem_clk` cycle lead on `ctl_wdata_valid`.

(3) The latency between ctl_cs_n being asserted and `ctl_wdata_valid` going high is effectively `ctl_wlat` (in this example, two) controller clock (ctl_clk) cycles. This can be thought of in terms of relative memory clock (`mem_clk`) cycles, in which case the latency is four `mem_clk` cycles.

(4) Only the upper half is valid (as the `ctl_wdata_valid` signal demonstrates, there is one `ctl_wdata_valid` bit to two 8-bit words). The write data bits go out on the bus in order, least significant byte first. So for a continuous burst of write data on the DQ pins, the most significant half of write data is used, which goes out on the bus last and is therefore contiguous with the following data. The converse is true for the end of the burst. Write data is spread across three controller clock (`ctl_clk`) cycles, but still only four memory clock (`mem_clk`) cycles. However, in relative memory clock cycles the latency is equivalent in the word-aligned and word-unaligned cases.

(5) The `0504` here is residual from the previous clock cycle. In the same way that only the upper half of the write data is used for the first beat of the write, only the lower half of the write data is used in the last beat of the write. These upper bits can be driven to any value in this alignment.

Figure 2–11. Word-Unaligned Reads



Notes to Figure 2–11:

(1) Similar to word-aligned reads, `ctl_doing_rd` is asserted one memory clock cycle before chip select (`ctl_cs_n`) is asserted, which for a word-unaligned read is in the previous controller clock (`ctl_clk`) cycle. In this example the `ctl_doing_rd` signal is now spread over three controller clock (`ctl_clk`) cycles, the high bits in the sequence '10','11','01','10','11','01' providing the required four memory clock cycles of assertion for `ctl_doing_rd` for the two 4-beat reads in the full-rate memory clock domain, '011110','011110'.

(2) The return pattern of `ctl_rdata_valid` is a delayed version of `ctl_doing_rd`. Advertised read latency (`ctl_rlat`) is the number of controller clock (`ctl_clk`) cycles delay inserted between `ctl_doing_rd` and `ctl_rdata_valid`.

(3) The read data (`ctl_rdata`) is spread over three controller clock cycles and in the pointed to vector only the upper half of the `ctl_rdata` bit vector is valid (denoted by `ctl_rdata_valid`).

Using a Custom Controller

The ALTMEMPHY megafunction can be integrated with your own controller. This section describes the interface requirement and the handshake mechanism for efficient read and write transactions.

Preliminary Steps

Perform the following steps to generate the ALTMEMPHY megafunction:

1. If you are creating a custom DDR3 SDRAM controller, generate the Altera high-performance controller targeting your chosen Altera and memory devices.
2. Compile and verify the timing. This step is optional.
3. If targeting a DDR3 SDRAM device, simulate the high-performance controller design so you can determine how to drive the PHY signals using your own controller.
4. Integrate the top-level ALTMEMPHY design with your controller. If you started with the high-performance controller, the PHY variation name is <*controller_name*>_phy.v/.vhd. Details about integrating your controller with Altera's ALTMEMPHY megafunction are described in the following sections.
5. Compile and simulate the whole interface to ensure that you are driving the PHY properly and that your commands are recognized by the memory device.

Design Considerations

This section discuss the important considerations for implementing your own controller with the ALTMEMPHY megafunction. This section describes the design considerations for AFI variants.

Simulating the high-performance controller is useful if you do not know how to drive the PHY signals.

Clocks and Resets

The ALTMEMPHY megafunction automatically generates a PLL instance, but you must still provide the reference clock input (`pll_ref_clk`) with a clock of the frequency that you specified in the MegaWizard Plug-In Manager. An active-low global reset input is also provided, which you can deassert asynchronously. The clock and reset management logic synchronizes this reset to the appropriate clock domains inside the ALTMEMPHY megafunction.

A clock output, half the memory clock frequency for a half-rate controller, is provided and all inputs and outputs of the ALTMEMPHY megafunction are synchronous to this clock. For AFIs, this signal is called `ctl_clk`.

There is also an active-low synchronous reset output signal provided, `ctl_reset_n`. This signal is synchronously de-asserted with respect to the `ctl_clk` or `phy_clk` clock domain and it can reset any additional user logic on that clock domain.

Calibration Process Requirements

When the global reset_n is released the ALTMEMPHY handles the initialization and calibration sequence automatically. The sequencer calibrates memory interfaces by issuing reads to multiple ranks of DDR3 SDRAM (multiple chip select). Timing margins decrease as the number of ranks increases. It is impractical to supply one dedicated resynchronization clock for each rank of memory, as it consumes PLL resources for the relatively small benefit of improved timing margin. When calibration is complete ctl_cal_success goes high if successful; ctl_cal_fail goes high if calibration fails. Calibration can be repeated by the controller using the soft_reset_n signal, which when asserted puts the sequencer into a reset state and when released the calibration process begins again.

Other Local Interface Requirements

The memory burst length for DDR3 SDRAM devices can be set at either four or eight; but when using the Altera high-performance controller, only burst length eight is supported. For a half-rate controller, the memory clock runs twice as fast as the clock provided to the local interface, so data buses on the local interface are four times as wide as the memory data bus.

Address and Command Interfacing

Address and command signals are automatically sized for 1T operation, such that for full-rate designs there is one input bit per pin (for example, one cs_n input per chip select configured); for half-rate designs there are two. If you require a more conservative 2T address and command scheme, use a full-rate design and drive the address/command inputs for two clock cycles, or in a half-rate design drive both address/command bits for a given pin identically.

Although the PHY inherently supports 1T addressing, the high-performance controller supports only 2T addressing, so PHY timing analysis is performed assuming 2T address and command signals.

Handshake Mechanism Between Read Commands and Read Data

When performing a read, a high-performance controller with the AFI asserts ctl_doing_read to indicate that a read command is requested and the byte lanes that it expects valid data to return on. ALTMEMPHY uses ctl_doing_read for the following actions:

- Control of the postamble circuit
- Generation of ctl_rdata_valid
- Dynamic termination (Rt) control timing

The read latency, ctl_rlat, is advertised back to the controller. This signal indicates how long it takes in ctl_clk clock cycles from assertion of ctl_doing_read to valid read data returning on ctl_rdata. The ctl_rlat signal is only valid when calibration has successfully completed and never changes values during normal user mode operation.

The ALTMEMPHY provides a signal, `ctl_rdata_valid`, to indicate that the data on read data bus is valid. The width of this signal varies between half-rate and full-rate designs to support the option to indicate that the read data is not word aligned. Figure 2–12 and Figure 2–13 show these relationships.

Figure 2–12. Address and Command and Read-Path Timing—Full-Rate Design

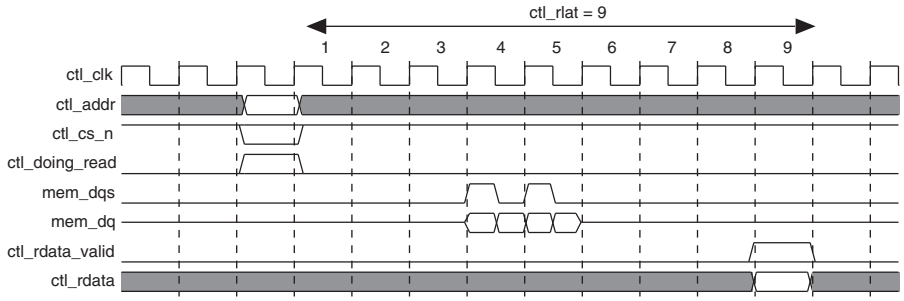
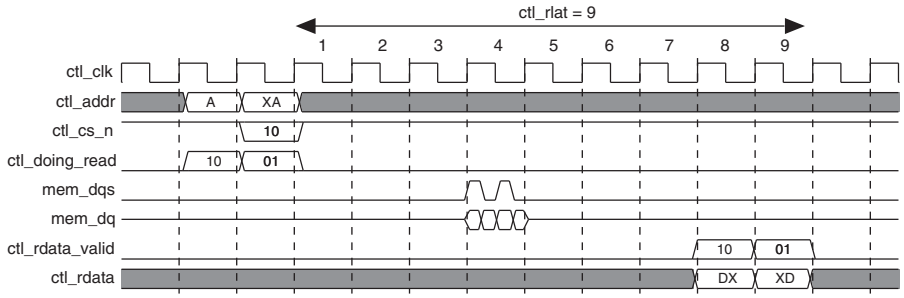


Figure 2–13. Second Read Alignment—Half-Rate Design



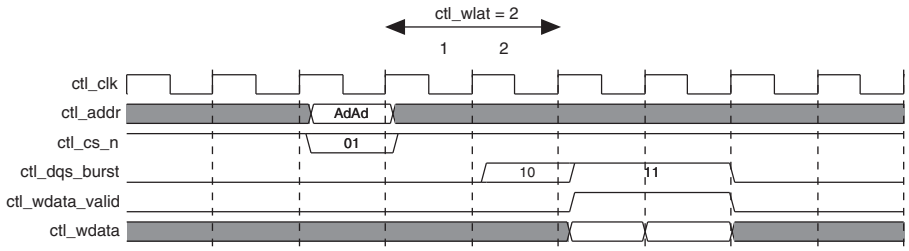
Handshake Mechanism Between Write Commands and Write Data

In the AFI, the ALTMEMPHY output `ctl_wlat` gives the number of `ctl_clk` cycles between the write command that is issued `ctl_cs_n` asserted and `ctl_dqs_burst` asserted. The `ctl_wlat` signal considers the following actions to provide a single value in `ctl_clk` clock cycles:

- CAS write latency
- Additive latency
- Datapath latencies and relative phases
- Board layout
- Address and command path latency and `1T` register setting, which is dynamically setup to take into account any leveling effects

The ctl_wlat signal is only valid when the calibration has been successfully completed by the ALTMEMPHY sequencer and does not change at any point during normal user mode operation. Figure 2–14 shows the operation of ctl_wlat port.

Figure 2–14. Timing for ctl_dqs_burst, ctl_wdata_valid, Address, and Command—Half-Rate Design



For a half-rate design ctl_cs_n is 2 bits, not 1. Also the ctl_dqs_burst and ctl_wdata_valid waveforms indicate a half-rate design. This write results in a burst of 8 at the DDR. Where ctl_cs_n is driven 2'b01, the LSB (1) is the first value driven out of mem_cs_n, and the MSB (0) follows on the next mem_clk. Similarly, for ctl_dqs_burst, the LSB is driven out of mem_dqs first (0), then a 1 follows on the next clock cycle. This sequence produces the continuous DQS pulse as required. Finally, the ctl_addr bus is twice MEM_IF_ADDR_WIDTH bits wide and so the address is concatenated to result in an address phase two mem_clk cycles wide.

Partial Writes

As part of the DDR3 SDRAM memory specifications, you have the option for partial write operations by asserting the DM pins for part of the write signal.

For designs targeting the Arria II devices, deassert the ctl_wdata_valid signal during partial writes, when the write data is invalid, to save power by not driving the DQ outputs.

For designs targeting other devices, use only the DM pins if you require partial writes. Assert the ctl_dqs_burst and ctl_wdata_valid signals as for full write operations, so that the DQ and DQS pins are driven during partial writes.

Controller Register Map

The controller register map allows you to control the memory controller settings.

For information on the controller register map, refer to "Controller Register Map" on page 5–29 of this volume.

ALTMEMPHY Calibration Stages

In all configurations, the noncalibrated address, command and control interfaces must be correctly constrained and meet timing.

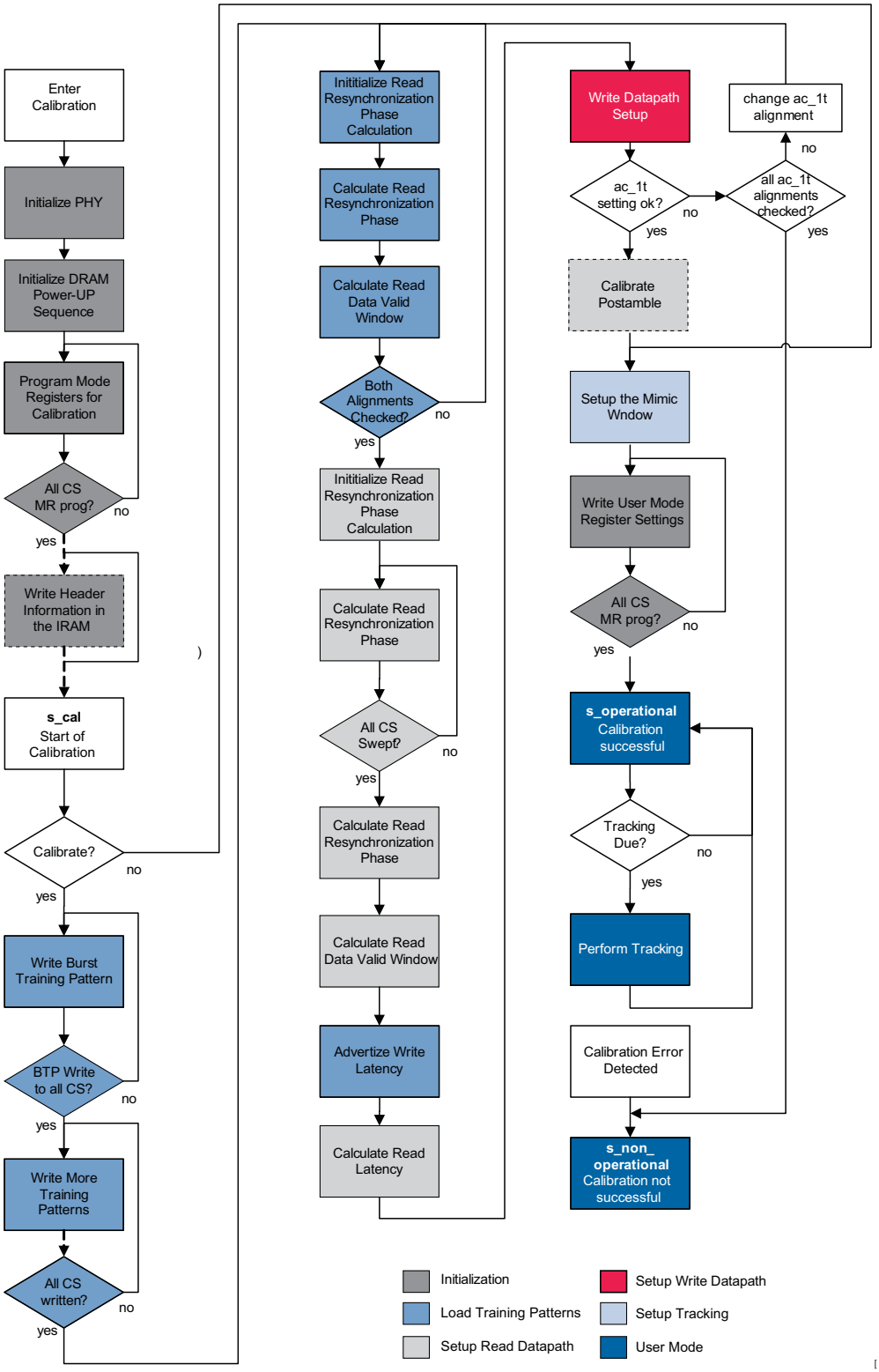
If calibration fails at a specific stage, use this chapter to understand what functionally happens at that stage, to assist with the debug.

The ALTMEMPHY IP performs the following calibration stages:

1. Enter Calibration (s_reset)
2. Initialize PHY (s_phy_initialize)
3. Initialize DRAM
4. Write Header Information in the internal RAM (s_write_ihi)
5. Load Training Patterns
6. Test More Pattern Writes
7. Calibrate Read Resynchronization Phase
8. Advertize Write Latency (s_was)
9. Calculate Read Latency (s_adv_rlat)
10. Output Write Latency (s_adv_wlat)
11. Calibrate Postamble (s_poa)
12. Set Up Address and Command Clock Cycle
13. Write User Mode Register Settings (s_prep_customer_mr_setup)
14. Voltage and Temperature Tracking

This chapter discusses these stages. Figure 2–15 on page 2–28 shows a flow chart of the calibration stages for the ALTMEMPHY IP.

Figure 2–15. Calibration Stages



Enter Calibration (s_reset)

Calibration starts when the ALTMEMPHY IP deasserts the PHY reset signal and the AFI signal `ctl_cal_req` is low.

Initialize PHY (s_phy_initialize)

This stage holds off calibration until the DLL has locked, and (if debug toolkit is enabled) internal RAM contents are all reset to zero.

Initialize DRAM

Initializing the DRAM has the following two stages:

- Initialize DRAM Power Up Sequence (s_int_dram)
- Program Mode Registers for Calibration (s_prog_mr)

Initialize DRAM Power Up Sequence (s_int_dram)

This stage brings the SDRAM out of a reset state (from any previous state) through the initialization sequence specified in the JEDEC specification for each device type, up to but not including mode register set commands. At the end of this stage, the SDRAM is ready to receive mode register load commands, which must occur (on each rank) before refreshes can occur.

This calibration stage applies to all chip selects in parallel.

Program Mode Registers for Calibration (s_prog_mr)

The ALTMEMPHY IP issues mode register set commands on a per chip select basis, which allows you great flexibility to issue different mode register settings to different chip selects. When all chip selects have mode registers programmed (the initialization of that chip select is complete), refreshes are enabled.

The following overrides apply to user settings:

- For DDR and DDR2 SDRAM:
  - DLL enable
  - Burst length 4
  - OCD calibration (DDR2 only)
- For DDR3 SDRAM:
  - DLL enable
  - Output buffer enable
  - Disable write leveling
  - Runtime burst length select
  - Test mode disabled
  - DLL reset

For DDR3 SDRAM this stage also includes a `ZQ-cal` long operation (refer to the JEDEC specification).

Write Header Information in the internal RAM (s_write_ihi)

In this stage, the ALTMEMPHY IP loads the internal header information in the first eight locations in the internal RAM through the parameterization of the ALTMEMPHY IP. The debug toolkit uses this information to provide the current ALTMEMPHY IP parameterization and IP version numbers.

The ALTMEMPHY IP only executes this stage when you enable debug toolkit.

For information about the debug toolkit, refer to the ALTMEMPHY External Memory Interface Debug Toolkit.

Load Training Patterns

In this stage, the ALTMEMPHY IP writes training patterns to the memory to be read in later calibration stages. Because of the matched trace lengths to DDR SDRAM components, after memory initialization, you can assume write capture works.

You can divide the training pattern writes into the following two stages:

- Write Block Training Pattern (s_write_btp)
- Write More Training Patterns (s_write_mtp)

The ALTMEMPHY IP writes further training pattern in the calibration of the write datapath, refer to Advertize Write Latency (s_was).

Write Block Training Pattern (s_write_btp)

This stage applies to read data valid alignment (`s_rdv`), advertise read latency (`s_adv_rd_lat`) and postamble calibration (`s_poa`). For these calibration stages a pattern of all 1s and all 0s is sufficient to set up the PHY.

Writing of these two patterns is trivial and requires all DDIO outputs (high and low phases (bits)) to be held at either 1 or 0, for all 1 and all 0 patterns, respectively. To write these patterns, the ALTMEMPHY IP holds the DDIO outputs low (or high) and toggles DQS for a predetermined length of time and issues a single write command. The ALTMEMPHY IP tests a full range of memory write latencies.

To support DDR3 SDRAM discrete components (burst of eight reads), the ALTMEMPHY IP loads eight memory locations with 1s and eight with 0s.

The following memory locations contain the following patterns:

- Locations [7.:0], all 0s
- Locations [15:8], all 1s

You need patterns of all 1s and all 0s for calibrating the read resynchronization phase.

Write More Training Patterns (s_write_mtp)

This stage calculates the read resynchronization phase (`s_rrp_sweep`).

The pattern is `0x30F5` and comprises separately written patterns. Th e ALTMEMPHY IP requires this pattern to match the characterization behavior for nonDQS capture based schemes (for example, Cyclone III devices). All device families use the following pattern:

- All 0: `'b0000` to DDIO high and low bits held at 0
- All 1: `'b1111` to DDIO high and low bits held at 1
- Toggle: `'b0101` to DDIO high bits held at 0 and DDIO low bits held at 1
- Mixed: `'b0011` to DDIO high and low bits have to toggle

While you can ensure that all zeros, all ones, or toggle are written into a burst of memory locations (output DDIO bits are held at constant values), it is challenging to ensure that a pattern of `0011` is written into memory. The challenge occurs because the write latency is unknown at this time. For example, if this pattern is repeated on DQ pins (`0011 0011 0011`) and a single write command issued (as for the other patterns), it is not known whether the memory location contains the pattern `0011` or `1100`.

The ALTMEMPHY IP provides a methodology to robustly write these patterns (Test More Pattern Writes). For this section two locations (X and Y) are populated with write data, one contains the pattern `0011`; the other contains `1100`.

The memory locations contain the following patterns:

- `x30` alignment 0 to location (X): `23..16`
- `x30` alignment 1 to location (Y): `31..24`
- `xF5` to location: `39..32`

The ALTMEMPHY IP writes these patterns in bursts of four beats, so in the pattern `xF5`, `F` is written separately to 5. The ALTMEMPHY IP writes patterns `F` and `0` as a part of the writing of the block training pattern (Write Block Training Pattern (s_write_btp)).

Test More Pattern Writes

This stage comprises a number of calibration stages of the PHY, but the stage is described as one entity. This stage comprises:

- Initialize read resynchronization phase calculation (`s_rrp_reset`)
- Calculate read resynchronization phase (`s_rrp_sweep`)
- Calculate read data valid window (`s_read_mtp`)

The algorithm ensures the pattern `0011` is written to a known location in memory. The following assumptions and PHY settings apply to this stage:

- Assumptions:
  - Burst length of 4 writes
  - Write capture in the memory device works. Data can be safely written to memory. No write leveling is required. (Refer to JEDEC specification for more information on DDR3 SDRAM).
  - Writes are aligned on a clock cycle basis. You know which beats of DQ data to write on the low and high phases of DQS (ultimately DQ and DQS board delays are well matched).
- Settings:
  - Address and command 1T setting. You can add additional latency to the address and command path (specified as a maximum of one memory clock cycles (t)). This setting aligns write data to address and command signals relative to the controller clock domain, where address and command signals are issued in a given alignment. This setting is not required (`0t`) for a full-rate PHY.
  - Read data 1T alignment. Additional delay of captured read data to align with read commands in the half rate controller cock domain. The interpretation of 1T is the same as for address and command, but applied to read data. This setting does not apply to a full-rate PHY.
  - Read resynchronization phase setting. This setting is the primary task of the training pattern to correctly set the resynchronization phase in the middle of data valid window for the read data (on DQ pins) to be captured.

From this algorithm, to determine PHY settings B and C, given that A is not set, follow these steps:

1. Try to write the pattern and indistinguishable variations of it to different memory locations. For example, write to different locations with the following two patterns on the DQ bus (timed to a local controller rate clock), refer to Write More Training Patterns (s_write_mtp):
   a. `0011 0011 0011` (try to write `0011`) in location X
   b. `1100 1100 1100` (try to write `1100`) in location Y
2. Perform two single-pin DQ pin and single-chip select read resynchronization phase calibrations using location X and location Y, as part of the larger training pattern (`0x30F5`). You do not know at this time which locations X and Y contain the pattern `0011`. This stage iterates through the following stages:
   - Initialize read resynchronization phase calculation (`s_rrp_reset`)
   - Calculate read resynchronization phase (`s_rrp_sweep`)
   - Calculate read data valid window (`s_read_mtp`)

   Calculate read data valid window (`s_read_mtp`) is a special case of calibrate read resynchronization phase (`s_rrp_sweep`), refer to Calibrate Read Resynchronization Phase (s_rrp_sweep). This stage reports the size of the returned window without setting up the PLL phase or producing an error if no window is observed.

3. The single pin read resynchronization calibration (using pattern X or Y), which results in the largest data valid window, contains the optimal pattern. The read resynchronization calibration with the largest window indicates the location (X or Y) that contains the correct alignment (`0011`). Read resynchronization phase calibration uses this alignment (X or Y) , to perform the full resynchronization phase calibration across all pins and chip selects.

During calibration of the read resynchronization phase, the ALTMEMPHY IP captures the DQ pin using a free running clock, phase shifted through a given number of steps. You should try to match a training pattern against read data, for each phase shift, and a window of valid phases is composed.

In waveforms, you observe two resynchronization phase sweeps, over single pins, before the larger phase sweep. However in fast simulation mode, you observe two resynchronization phase sweeps when the duration of all three sweeps is equal. For half-rate interfaces, you may observe a total of six phase sweeps, where the entire calibration is repeated when the address and command 1T setting is toggled.

Calibrate Read Resynchronization Phase

This stage encompasses the following calibration stages:

- Initialize Read Resynchronisation Phase Calibration (s_rrp_reset)
- Calibrate Read Resynchronization Phase (s_rrp_sweep)
- Calculate Read Resynchronization Phase (s_rrp_seek)

This stage adjusts the phase of the resynchronization (or capture) clock to determine the optimal phase that gives the greatest margin. For DQS based capture schemes the resynchronization clock captures the outputs of DQS capture registers (DQS is the capture clock). In a non-DQS capture based scheme, the capture clock captures the input DQ pin data (the DQS signal is unused).

For all half-rate PHY interfaces, a 720° resynchronization or capture clock phase sweep is performed. For a half-rate PHY this sweep is effectively 360° of the half-rate clock, because resynchronization or capture clock is at the memory clock rate. A 720° sweep is required, so that read data can be presented to a controller aligned to one half-rate controller clock cycle.

For full-rate DQS based capture, because the DQ pins are captured using the DQS signal, in a 360° phase sweep, all resynchronization clock phases may pass. In this case the correct resynchronization phase to set cannot be determined. The correct phase is the one in the center of a valid window, where returned read data is correct. Thus a 720° phase sweep is performed. From 360 to 720°, a clock cycle of latency may be added to a 0 to 360° sweep. The returned read data is compared to a training pattern pseudo half-rate (at half the clock rate of the sequencer), so data can only be valid for 360° of the sweep. This method introduces edges to the data valid window such that a correct phase can be chosen.

For non-DQS capture in general, up to half (180°) of the swept capture clock phases can result in correct capture data, because the DDR to SDR conversion is performed by the capture clock, and thus high and low phases of DQ are captured in the incorrect alignment for half of the capture clock phases. Therefore, only 360° of capture clock need be swept for full-rate non-DQS capture based PHYs.

The pseudo half-rate case potentially adds one clock cycle of latency into the read datapath because of the 720° sweep. The ALTMEMPHY IP detects this occurrence and removes the clock cycle of latency in the calculate read resynchronization phase (`s_rrp_seek`).

Initialize Read Resynchronisation Phase Calibration (s_rrp_reset)

This stage returns the PLL to a nominal zero phase shift.

Calibrate Read Resynchronization Phase (s_rrp_sweep)

This stage performs a sweep through a parameterised 360° or 720° of resynchronization clock phase. The ALTMEMPHY IP optionally stores these results in the internal RAM.

The command has the following attributes:

- `Single_pin` to indicate just to sweep DQ pin 0 as for the use in testing the write more training patterns stage.
- `mtp_alignment` to say which location (X/Y) to use from the write more training patterns stage.

Calculate Read Resynchronization Phase (s_rrp_seek)

This stage calculates the size and center (in phase steps) of the largest data valid window found during the calibrate read resynchronization phase and sets PLL phase to the center phase.

Calculate read data valid window (`s_read_mtp`) is a special case of this stage which reports no errors for an invalid window (a failure is expected in one case) and does not setup the PLL.

Calculate Read Data Valid Window (s_rdv)

This stage sets the latency on a delayed version of the `doing_rd` signal, so that it is aligned with the `rdata_valid` signal for the read data it is incident with the read command for.

This stage has the following process:

1. Reads a continuous stream of 1s followed by one read of zeros. The sequencer only asserts `doing_rd` when read command for zeros is issued.
2. Checks for alignment of read data valid signal (delayed version of `doing_rd`) to the zeros (`rdata` = 0, `rdata_valid` = 1).
3. If not aligned, reduces latency between `doing_rd` and `rdata_valid` signal and loop.

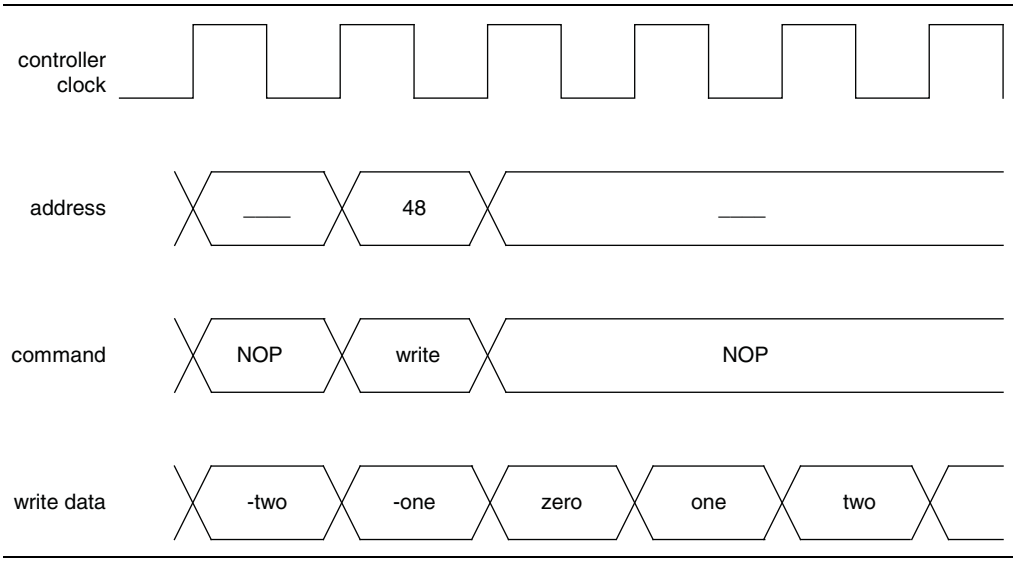
Read data valid latency is reset to a high value (before calibration) and then reduced until it matches the correct alignment.

Advertize Write Latency (s_was)

This stage writes a suitable pattern to the DRAM to calculate the write latency.

A write command is issued to a memory address 48 (Figure 2–16) while driving a count of frequency controller clock rate to the DQ pins (Figure 2–16), using the write datapath observed by the controller. For half-rate interfaces, each four beats of write data on DQ are identical. For full-rate interfaces, each two beats of write data are identical. In general, the write latency is written into addresses A... A + (n – 1), where n is two times the ratio of controller to memory clock rate and each n bits of write data must be identical. The pattern is written into memory locations 48 to 55.

Figure 2–16. Description of Write Pattern



Calculate Read Latency (s_adv_rlat)

In addition to read data valid window calculation (`s_rdv`), the advertised read latency is calculated in this stage in the following way:

- Issues a read command (with `doing_rd`) and starts a counter at PHY clock rate
- When `rdata_valid` returns, outputs the value of the counter to `ctl_rlat` signal.

This signal is redundant, because a controller can use the `rdata_valid` signal to determine when valid read data is returned.

The read data from the DRAM is not important here. The count is performed between the issue of `doing_rd` and `rdata_valid` returning.

Output Write Latency (s_adv_wlat)

To calibrate a PHY write datapath to a minimum latency, a robust process is required to determine the write latency (WL) between a memory controller write command and the associated write data. Factors that can contribute to WL include memory CAS latency, arbitrary additive delays in the PHY, and board trace lengths. The presented approach extends from calibrating a PHY, where the controller operates at the memory clock frequency, to controller operation at half or a quarter of the memory clock frequency.

After a predetermined maximum read latency clock cycles have passed, the contents of the chosen memory address (`0x48` in Figure 2–16) are read to recover the write latency. The first *n* beats of read data contain the write latency.

While this method recovers the write latency it can determine the address and command 1T setting in half rate mode.

The returned read data, in the controller clock domain, should be equal across the first four read data beats, as aligned to the controller clock domain. For this check to work an alternate read location must be read immediately before and after that containing the write latency.

If read data is not correctly aligned then the address and command 1T setting is toggled and calibration is rerun from write training patterns stage.

Calibrate Postamble (s_poa)

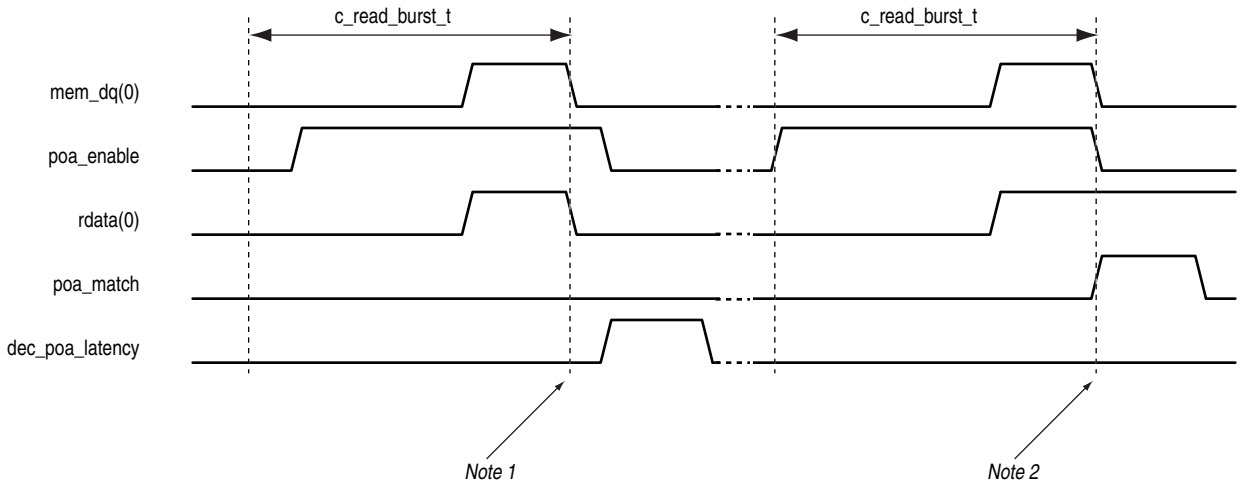
The ALTMEMPHY IP only implements this stage for DQS capture based PHYs (not used for Cyclone III and Cyclone IV devices).

For this stage, the PHY reads the pattern `0x30` from memory, so that the deassertion of the postamble protection signal (`poa_enable`) can be aligned to the 1's in this pattern.

This stage sets the correct clock cycle for the postamble path. The aim of the postamble path is to eliminate false DQ data capture because of postamble glitches on the DQS signal, through an override on DQS. This stage ensures the correct clock cycle timing of the postamble enable (override) signal.

The delay on the postamble enable signal starts off too large. It is then iteratively reduced until postamble enable de-asserts in the clock cycle before the last falling edge on DQS. Figure 2–17 shows the calibration timing diagram.

Figure 2–17. Calibration Timing



Note to Figure 2–17:

(1) The `poa_enable` signal is late, and the zeros on `mem_dq` after here are captured.

(2) The `poa_enable` signal is aligned. Zeros following here are not captured and `rdata` remains at 1.

Set Up Address and Command Clock Cycle

For half-rate interfaces, this stage also optionally adds an additional memory clock cycle of delay from the address and command path. This stage aligns write data to memory commands given in the controller clock domain. You see this stage in the waveform as a rerun of calibration (from the writing of training patterns) to calibrate to the new setting.

This stage is detected in the advertise write latency stage (`s_adv_wlat`)

Write User Mode Register Settings (s_prep_customer_mr_setup)

User mode register setting applies on a per chip select basis without the overrides in the program mode registers for calibration (s_prog_mr) stage.

Voltage and Temperature Tracking

Voltage and temperature tracking is a background process that tracks the voltage and temperature variations to maintain the relationship between the resynchronization or capture clock and the data valid window that were achieved at calibration. When the data calibration phase completes, the sequencer issues the mimic calibration sequence every 128 ms (in user mode).

Setup the Mimic Window (s_tracking_setup)

During initial calibration, the mimic path is sampled using the measure clock. The `measure_clk` signal has a `_1x` or `_2x` suffix, depending whether the ALTMEMPHY IP is a full-rate or half-rate design. The sampled value is then stored by the sequencer. After a sample value is stored, the sequencer uses the PLL reconfiguration logic to change the phase of the measure clock by one voltage-controlled oxcillator (VCO) phase tap. The sequencer then stores the sampled value for the new mimic path clock phase. This sequence continues until all mimic path clock phase steps are swept. After the sequencer stores all the mimic path sample values, it calculates the phase which corresponds to the center of the high period of the mimic path waveform. This reference mimic path sampling phase is used during the voltage and temperature tracking phase.

Perform Tracking (s_tracking)

In user mode, the sequencer periodically performs a tracking operation. At the end of the tracking calibration, the sequencer compares the most recent optimum tracking phase against the reference sampling phase. If the sampling phases do not match, the mimic path delays have changed because of voltage and temperature variations. When the sequencer detects that the mimic path reference and most recent sampling phases do not match, the sequencer uses the PLL reconfiguration logic to change the phase of the resynchronization clock by the VCO taps in the same direction. This procedure allows the tracking process to maintain the near-optimum capture clock phase setup during data tracking calibration as voltage and temperature vary over time. The relationship between the resynchronization or capture clock and the data valid window is maintained by measuring the mimic path variations because of the voltage and temperature variations and applying the same variation to the resynchronization clock.

Document Revision History

Table 2–6 lists the revision history for this document.

Table 2–6. Document Revision History

Date	Version	Changes
November 2012	3.3	■ Added Controller Register Map information.
June 2012	3.2	■ Added Feedback icon
November 2011	3.1	■ Consolidated ALTMEMPHY FD information from 11.0 version **DDR and DDR2 SDRAM Controller with ALTMEMPHY IP User Guide** and **DDR3 SDRAM Controller with ALTMEMPHY IP User Guide**. ■ Added ALTMEMPHY Calibration Stages information.

3. Functional Description—Hard Memory Interface

EMI_RM_003-2.1

This chapter describes the hard (on-chip) memory interface components available in the Arria V and Cyclone V device families.

Hard Memory Interface

The Arria V device family includes hard memory interface components supporting DDR2 and DDR3 SDRAM, LPDDR2, QDR II SRAM, and RLDRAM II memory protocols at speeds of up to 533 MHz. For the Quartus II software version 12.0 and later, the Cyclone V device family supports both hard and soft interface support.

High-Level Feature Description

The hard memory interface consists of three main parts, as follows:

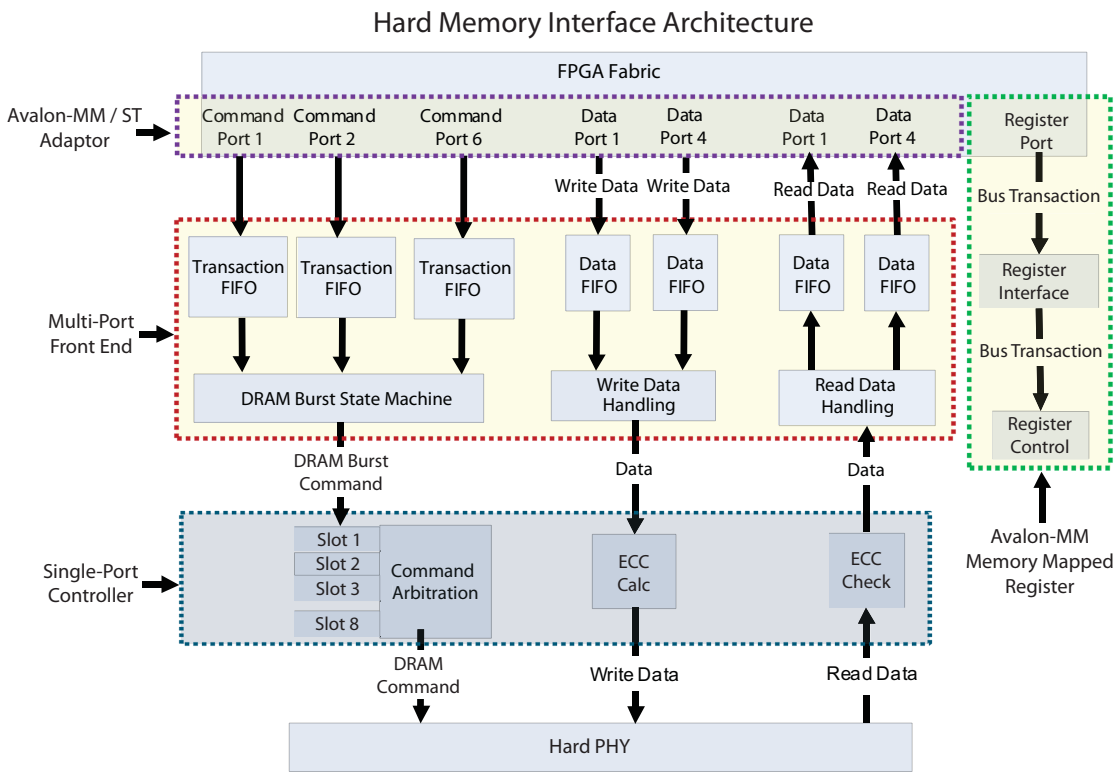
- The multi-port front end (MPFE), which allows multiple independent accesses to the hard memory controller.
- The hard memory controller, which initializes, refreshes, manages, and communicates with the external memory device.
- The hard PHY, which provides the physical layer interface to the external memory device.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure 3–1 shows the architecture of the Arria V hard memory interface.

Figure 3–1. Hard Memory Interface Architecture



Multi-Port Front End (MPFE)

The multi-port front end and its associated fabric interface provide up to six command ports, four read-data ports and four write-data ports, through which user logic can access the hard memory controller. Each port can be configured as read only or write only, or read and write ports may be combined to form bidirectional data ports. Ports can be 32, 64, 128, or 256 data bits wide, depending on the number of ports used and the type (unidirectional or bidirectional) of the port.

Fabric Interface

The fabric interface provides communication between the Avalon-ST-like internal protocol of the hard memory interface and the external Avalon-MM protocol. The fabric interface supports frequencies in the range of 10 MHz to one-half of the memory interface frequency. For example, for an interface running at 533 MHz, the maximum user logic frequency is 267 MHz. The MPFE handles the clock crossing between user logic and the hard memory interface.

Table 3–1 lists the types of ports available in the fabric interface.

Table 3–1. Fabric Interface Port Types

Port Type	Description
Fabric command ports	Accept both read and write commands.
Fabric 64-bit read data ports	Read and write data ports can be concatenated to form wider interfaces in power-of-two sizes (128-, 256-bit busses). Application which require only 32-bit interface can connect a 32-bit interface to the lower 32 bits of 64-bit port and configure that interface to be 32-bits wide. The remaining 32 nets to the hard memory controller are not usable when a port is configured to be 32-bits wide. 32-bit interfaces cannot be concatenated.
Fabric 64-bit write data ports	
Fabric register port	Provides access to address registers that control operation of the memory controller. Register values written across this interface can override values loaded during FPGA configuration.
Fabric write response port	Paired with the fabric write data ports to provide return acknowledgement of write operations being committed. A read operation received after the write acknowledgement on any controller port for the same address will see the updated data.

Operation Ordering

Requests arriving at a given port are executed in the order in which they are received.

Requests arriving at different ports have no guaranteed order of service, except when a first transaction has completed before the second arrives.

Multi-port Scheduling

User-configurable priority and weight settings determine the absolute and relative scheduling policy for each port.

Port Scheduling

Multi-port scheduling is governed by two considerations: the absolute priority of a request and the weighting of a port.

The evaluation of absolute priority ensures that ports carrying higher-priority traffic are served ahead of ports carrying lower-priority traffic. The scheduler recognizes eight priority levels, with higher values representing higher priorities. Priority is absolute; for example, any transaction with priority seven will always be scheduled before transactions of priority six or lower.

When ports carry traffic of the same absolute priority, relative priority is determined based on port weighting. Port weighting is a five-bit value, and is determined by a weighted round robin (WRR) algorithm.

The scheduler can alter priority if the latency target for a transaction is exceeded. The scheduler tracks latency on a per-port basis, and counts the cycles that a transaction is pending. Each port has a priority escalation register and a pending counter engagement register. If the number of cycles in the pending counter engagement register elapse without a pending transaction being served, that transaction's priority is esclated.

To ensure that high-priority traffic is served quickly and that long and short bursts are effectively interleaved on ports, bus transactions longer than a single DRAM burst are scheduled as a series of DRAM bursts, with each burst arbitrated separately.

The scheduler uses a form of deficit round robin (DRR) scheduling algorithm which corrects for past over-servicing or under-servicing of a port. Each port has an associated weight which is updated every cycle, with a user-configured weight added to it and the amount of traffic served subtracted from it. The port with the highest weighting is considered the most eligible.

To ensure that lower priority ports do not build up large running weights while higher priority ports monoplize bandwidth, the hard memory controller's DRR weights are updated only when a port matches the scheduled priority. Hence, if three ports have traffic, two being priority 7 and one being priority 4, the weights for both ports at priority 7 are updated but the port with priority 4 remains unchanged.

The scheduler can be configured to lock onto a given port for a specified number of transactions when the scheduler schedules traffic at that priority level. The number of transactions is configurable on a per-port basis. For ports with large numbers of sequential addresses, you can use this feature to allow efficient open page accesses without risk of the open page being pushed out by other transactions.

DRAM Burst Scheduling

DRAM burst scheduling recognizes addresses that access the same column/row combination—also known as open page accesses. Such operations are always served in the order in which they are received in the single-port controller.

Selection of DRAM operations is a two-stage process; first, each pending transaction must wait for its timers to be eligible for execution, then the transaction arbitrates against other transactions that are also eligible for execution.

The following rules govern transaction arbitration:

- High priority operations take precedence over lower priority operations
- If multiple operations are in arbitration, read operations have precedence over write operations
- If multiple operations still exist, the oldest is served first

A high-priority transaction in the DRAM burst scheduler wins arbitration for that bank immediately if the bank is idle and the high-priority transaction's chip select/row/column address does not match an address already in the single-port controller. If the bank is not idle, other operations to that bank yield until the high-priority operation is finished. If the address matches another chip select/row/column, the high-priority transaction yeilds until the earlier transaction is completed.

You can force the DRAM burst scheduler to serve transactions in the order that they are received, by setting a bit in the register set.

DRAM Power Saving Modes

The hard memory controller supports two DRAM power-saving modes: self-refresh, and fast/slow all-bank precharge powerdown exit.

Engagement of a DRAM power saving mode can occur due to inactivity, or in response to a user command.

The user command to enter power-down mode forces the DRAM burst-scheduling bank-management logic to close all banks and issue the power-down command. You can program the controller to power down when the DRAM burst-scheduling queue is empty for a specified number of cycles; the DRAM is reactivated when an active DRAM command is received.

MPFE Signal Descriptions

Table 3–2 describes the signals for the multi-port front end.

Table 3–2. MPFE Signals

Signal	Direction	Description
avl_<signal_name>_#(1)	—	Local interface signals.
mp_cmd_clk_#_clk (1)	Input	Clock for the command FIFO buffer. Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.
mp_cmd_reset_n_#_reset_n (1)	Input	Reset signal for command FIFO buffer.
mp_rfifo_clk_#_clk (2)	Input	Clock for the read data FIFO buffer. Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.
mp_rfifo_reset_n_#_reset_n (2)	Input	Reset signal for read data FIFO buffer.
mp_wfifo_clk_#_clk (2)	Input	Clock for the write data FIFO buffer. Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.
mp_wfifo_reset_n_#_reset_n (2)	Input	Reset signal for write data FIFO buffer.
bonding_in_1/2/3	Input	Bonding interface input port. Connect second controller bonding output port to this port according to the port sequence.
bonding_out_1/2/3	Output	Bonding interface output port. Connect this port to the second controller bonding intput port according to the port sequence.

Note:
(1) # represents the number of the slave port. Values are 0—5.
(2) # represents the number of the slave port. Values are 0—3.

Every input interface (command, read data, and write data) has its own clock domain. Each command port can be connected to a different clock, but the read data and write data ports associated with a command port must connect to the same clock as that command port. Each input interface uses the same reset signal as its clock.

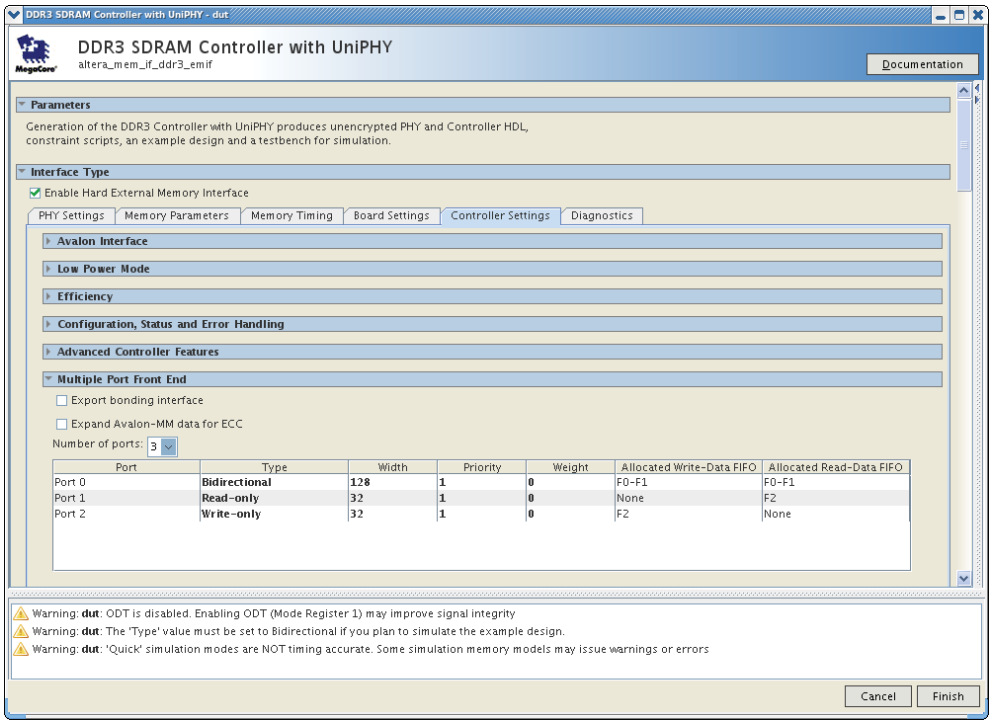
By default, the IP generates all clock signals regardless of the MPFE settings, but all unused ports and FIFO buffers are connected to ground.

The command ports can be used only in unidirectional configurations, with either 4 write and 2 read, 3 write and 3 read, or 2 write and 4 read scenarios. For bidirectional ports, the number of clocks is reduced from 6 to a maximum of 4.

For the scenario depicted in Figure 3–2:

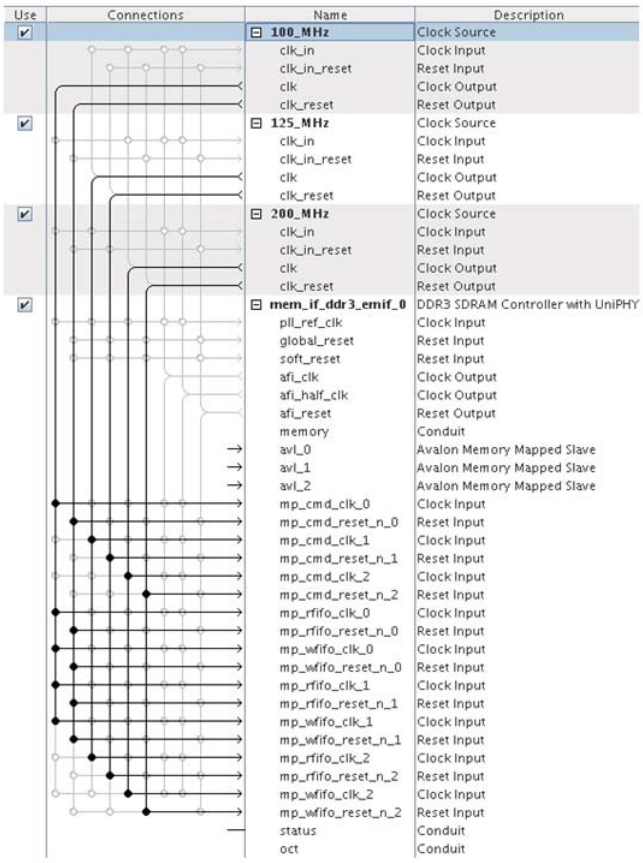
- command port 0 is associated with read and write data FIFO 0 and 1
- command port 1 is associated with read data FIFO 2
- command port 2 is associated with write data FIFO 2

Figure 3–2. Sample MPFE Configuration



Therefore, if port 0 (avl_0) is clocked by a 100 MHz clock signal, mp_cmd_clk_0, mp_rfifo_clk_0, mp_rfifo_clk_1, mp_wfifo_clk_o, and mp_wfifo_clk_1 must all be connected to the same 100 MHz clock, as illustrated in Figure 3–3.

Figure 3–3. Sample Connection Mapping



Hard Memory Controller

The following sections describe the memory controller portion of the hard memory interface.

The hard memory controller is functionally similar to the High Performance Controller II (HPC II). For information on signals, refer to the Functional Description—HPC II Controller chapter.

Clocking

The ports on the MPFE can be clocked at different frequencies, and synchronization is maintained by cross-domain clocking logic in the MPFE. Command ports can connect to different clocks, but the data ports associated with a given command port must be attached to the same clock as that command port. For example, a bidirectional command port that performs a 64-bit read/write function has its read port and write port connected to the same clock as the command port. Note that these clocks are separate from the EMIF core generated clocks.

DRAM Interface

The DRAM interface is 40 bits wide, and can accommodate 8-bit, 16-bit, 16-bit plus ECC, 32-bit, or 32-bit plus ECC configurations. Any unused I/Os in the DRAM interface can be reused as user I/Os. The DRAM interface supports DDR2 and DDR3 memory protocols, and LPDDR2 for Cyclone V only. Fast and medium speed grade devices are supported to 533 MHz for Arria V and 400 MHz for Cyclone V.

ECC

The hard controller supports both error-correcting code (ECC) calculated by the controller and by the user.

Controller ECC code employs standard Hamming logic which can detect and correct single-bit errors and detect double-bit errors. The controller ECC is available for 16-bit and 32-bit widths, each requiring an additional 8 bits of memory, resulting in an actual memory width of 24-bits and 40-bits, respectively.

In user ECC mode, all bits are treated as data bits, and are written to and read from memory. User ECC can implement nonstandard memory widths such as 24-bit or 40-bit, where ECC is not required.

Controller ECC

Controller ECC provides the following features:

Byte Writes—The memory controller performs a read/modify/write operation to keep ECC valid when a subset of the bits of a word is being written. If an entire word is being written (but less than a full burst) and the DM pins are connected, no read is necessary and only that word is updated. If controller ECC is disabled, byte-writes have no performance impact.

ECC Write Backs—When a read operation detects a correctable error, the memory location is scheduled for a read/modify/write operation to correct the single-bit error.

User ECC—User ECC is 24-bits or 40-bits wide; with user ECC, the controller performs no ECC checking. The controller employs memory word addressing with byte enables, and can handle arbitrary memory widths. User ECC does not disable byte writes; hence, you must ensure that any byte writes do not result in corrupted ECC.

Bonding of Memory Controllers

Bonding is a feature that allows data to be split between two memory controllers, providing the ability to service bandwidth streams similar to a single 64-bit controller. Bonding works by dividing data buses in proportion to the memory widths, and always sending a transaction to both controllers. When signals are returned, bonding ensures that both sets of signals are returned identically.

Bonding can be applied to asymetric controllers, and allows controllers to have different memory clocks. Bonding does not attempt to synchronize the controllers. Bonding supports only one port. The Avalon port width can be varied from 64-bit to 256-bit; 32-bit port width is not supported.

The following signals require bonding circuitry:

Read data return—This bonding allows read data from the two controllers to return with effectively one ready signal to the bus master that initiated the bus transaction.

Write ready—For Avalon-MM, this is effectively bonding on the `waitrequest` signal.

Write acknowledge—Synchronization on returning the write completed signal.

For each of the above implementations, data is returned in order, hence the circuitry must match up for each valid cycle.

Bonded FIFO buffers must have identical FIFO numbers; that is, read FIFO 1 on controller 1 must be paired with Read FIFO 1 on controller 2.

Data Return Bonding

Long loop times can lead to communications problems when using bonded controllers. The following effects are possible when using bonded controllers:

- If one memory controller completes its transaction and receives new data before the other controller, then the second controller can send data as soon as it arrives, and before the first controller acknowledges that the second controller has data.
- If the first controller has a single word in its FIFO buffer and the second controller receives single-word transactions, the second controller must determine whether the second word is a valid signal or not.

To accommodate the above effects, the hard controller maintains two counters for each bonded pair of FIFO buffers and implements logic that monitors those counters to ensure that the bonded controllers receive the same data on the same cycle, and that they send the data out on the same cycle.

FIFO Ready

FIFO ready bonding is used for write command and write data buses. The implementation is similar to the data return bonding.

Bonding Latency Impact

Bonding has no latency impact on ports that are not bonded.

Bonding Controller Usage

Arria V and Cyclone V devices employ three shared bonding controllers to manage the read data return bonding, write acknowledge bonding, and command/write data ready bonding.

The three bonding controllers require three pairs of bonding I/Os, each based on a six port count; this means that a bonded hard memory controller requires 21 input signals and 21 output signals for its connection to the fabric, and another 21 input signals and 21 output signals to the paired hard memory controller.

The hard processor system (HPS) hard memory controller cannot be bonded with another hard memory controller on the FPGA portion of the device.

Hard PHY

A physical layer interface (PHY) is embedded in the periphery of the Arria V device, and can run at the same high speed as the hard controller and hard sequencer. The hard PHY is located next to the hard controller. Differing device configurations have different numbers and sizes of hard controller and hard PHY pairs.

The hard PHY implements logic that connects the hard controller to the I/O ports. Because the hard controller and AFI interface support high frequencies, a portion of the sequencer is implemented as hard logic. The Nios II processor, the instruction/data RAM, and the Avalon fabric of the sequencer are implemented as core soft logic. The read/write manger and PHY manager components of the sequencer, which must operate at full rate, are implemented as hard logic in the hard PHY.

Interconnections

The hard PHY resides on the device between the hard controller and the I/O register blocks. The hard PHY is instantiated or bypassed entirely, depending on the parameterization that you specify.

The hard PHY connects to the hard memory controller and the core, enabling the use of either the hard memory controller or a software-based controller. (You can have the hard controller and hard PHY, or the soft controller and soft PHY; however, the combination of soft controller with hard PHY is not supported.) The hard PHY also connects to the I/O register blocks and the DQS logic. The path between the hard PHY and the I/O register blocks can be bypassed, but not reconfigured—in other words, if you use the hard PHY datapath, the pins to which it connects are predefined and specified by the device pin table.

Clock Domains

The hard PHY contains circuitry that uses the following clock domains:

AFI clock domain (pll_afi_clk)—The main full-rate clock signal that synchronizes most of the circuit logic.

Avalon clock domain (pll_avl_clk)—Synchronizes data on the internal Avalon bus, namely the Read/Write Manager, PHY Manager, and Data Manager data. The data is then transferred to the AFI clock domain. To ensure reliable data transfer between clock domains, the Avalon clock period must be an integer multiple of the AFI clock period, and the phases of the two clocks must be aligned.

Address and Command clock domain (pll_addr_cmd_clk)—Synchronizes the global asychronous reset signal, used by the I/Os in this clock domain.

Hard Sequencer

The sequencer initializes the memory device and calibrates the I/Os, with the objective of maximizing timing margins and achieving the highest possible performance. When the hard memory controller is in use, a portion of the sequncer must run at full rate; for this reason, the Read/Write Manager, PHY Manager, and Data Manager are implemented as hard components within the hard PHY. The hard sequencer communicates with the soft-logic sequencer components (including the Nios II processor) via an Avalon bus.

Hard Memory Interface Implementation Guidelines

The following sections provide guidelines for implementing a hard memory interface.

MPFE Setup Guidelines

This section provides information on configuring the multi-port front end of the hard memory interface.

1. To enable the hard memory interface, turn on **Enable Hard External Memory Interface** in the **Interface Type** tab in the parameter editor.
2. To export bonding interface ports to the top level, turn on **Export bonding interface** in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

   ☞ The system exports three bonding-in ports and three bonding-out ports. You must generate two controllers and connect the bonding ports manually.

3. To expand the interface data width from a maximum of 32 bits to a maximum of 40 bits, turn on **Enable Avalon-MM data for ECC** in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

   ☞ The controller does not perform ECC checking when this option is turned on.

4. Select the required **Number of ports** for the multi-port front end in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

   ☞ The maximum number of ports is 6, depending on the port type and width. The maximum port width is 256 bits, which is the maximum data width of the read data FIFO and write data FIFO buffers.

5. The table in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor lists the ports that are created. The columns in the table describe each port, as follows:
   - **Port**: Indicates the port number.
   - **Type**: Indicates whether the port is read only, write only, or bidirectional.
   - **Width**: To achieve optimum MPFE throughput, Altera recommends setting the MPFE data port width according to the following calculation:

     ```
     2 x (frequency ratio of HMC to user logic) x (interface data width)
     ```

     For example, if the frequency of your user logic is one-half the frequency of the hard memory controller, you should set the port width to be 4x the interface data width. If the frequency ratio of the hard memory controller to user logic is a fractional value, you should use a larger value; for example, if the ratio is 1.5, you can use 2.
   - **Priority**: The priority setting specifies the priority of the slave port, with higher values representing higher priority. The slave port with highest priority is served first.
   - **Weight**: The weight setting has a range of values of 0–31, and specifies the relative priority of a slave port, with higher weight representing higher priority. The weight value can determine relative bandwidth allocations for slave ports with the same priority values.

     For example, if two ports have the same priority value, and weight values of 4 and 6, respectively, the port with a weight of 4 will receive 40% of the bus bandwidth, while the port with a weight of 6 will receive 60% of the bus bandwidth—assuming 100% total available bus bandwidth.

Soft Memory Interface to Hard Memory Interface Migration Guidelines

This section provides information on mapping your soft memory interface to a hard memory interface.

Pin Connections

1. The hard and soft memory controllers have compatible pinouts. Assign interface pins to the hard memory interface according to the pin table.
2. Ensure that your soft memory interface pins can fit into the hard memory interface. The hard memory interface can support a maximum of a 40-bit interface with user ECC, or a maximum of 80-bits with same-side bonding. The soft memory interface does not support bonding.
3. Follow the recommended board layout guidelines for the hard memory interface.

Software Interface Preparation

Observe the following points in preparing your soft memory interface for migration to a hard memory interface:

- You cannot use the hard PHY without also using the hard memory controller.
- The hard memory interface supports only full-rate controller mode.

- Ensure that the MPFE data port width is set according to the soft memory interface half-rate mode Avalon data width.
- The hard memory interface uses a different Avalon port signal naming convention than the software memory interface. Ensure that you change the `.avl_<signal_name>` signals in the soft memory interface to `.avl_<signal_name>_0` signals in the hard memory interface.
- The hard memory controller MPFE includes an additional three clocks and three reset ports (CMD port, RFIFO port, and WFIFO port) that do not exist with the soft memory controller. You should connect the user logic clock signal to the MPFE clock port, and the user logic reset signal to the MPFE reset port.
- In the soft memory interface, the half-rate `afi_clk` is a user logic clock. In the hard memory interface, `afi_clk` is a full-rate clock, because the core fabric may not be able to achieve full-rate speed. When you migrate your soft memory interface to a hard memory interface, you need to supply an additional slower rate clock. The maximum clock rate supported by core logic is one-half of the maximum interface frequency.

Latency

Overall, you should expect to see slightly more latency when using the hard memory controller and multi-port front end, than when using the soft memory controller.

The hard memory controller typically exhibits lower latency than the soft memory controller; however, the multi-port front end does introduce additional latency cycles due to FIFO buffer stages used for synchronization. The MPFE cannot be bypassed, even if only one port is needed.

Bonding Interface Guidelines

Bonding allows a single data stream to be split between two memory controllers, providing the ability to expand the interface data width similar to a single 64-bit controller. This section provides some guidelines for setting up the bonding interface.

1. Bonding interface ports are exported to the top level in your design. You should connect each `bonding_in_*` port in one hard memory controller to the corresponding `bonding_out_*` port in the other hard memory controller, and vice versa.

2. You should modify the Avalon signal connections to drive the bonding interface with a single user logic/master, as follows:
   a. AND both `avl_ready` signals from both hard memory controllers before the signals enter the user logic.
   b. AND both `avl_rdata_valid` signals from both hard memory controllers before the signals enter the user logic. (The `avl_rdata_valid` signals should be identical for both hard memory controllers.)
   c. Branch the following signals from the user logic to both hard memory controllers:
      - `avl_burstbegin`
      - `avl_addr`
      - `avl_read_req`
      - `avl_write_req`
      - `avl_size`
   d. Split the following signals according to each multi-port front end data port width:
      - `avl_rdata`
      - `avl_wdata`
      - `avl_be`

Document Revision History

Table 3–3 lists the revision history for this document.

Table 3–3. Document Revision History

Date	Version	Changes
November 2012	2.1	■ Added Hard Memory Interface Implementation Guidelines. ■ Moved content of EMI-Related HPS Features in SoC Devices section to chapter 4. Functional Description—HPS Memory Controller.
June 2012	2.0	■ Added EMI-Related HPS Features in SoC Devices. ■ Added LPDDR2 support. ■ Added Feedback icon.
November 2011	1.0	Initial release.

4. Functional Description—HPS Memory Controller

EMI_RM_017-1.0

The hard processor system (HPS) SDRAM controller subsystem provides efficient access to external SDRAM for the ARM® Cortex™-A9 microprocessor unit (MPU) subsystem, the level 3 (L3) interconnect, and the FPGA fabric. The SDRAM controller provides an interface between the FPGA fabric and HPS. The interface accepts Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI™) and Avalon® Memory-Mapped (Avalon-MM) transactions, converts those commands to the correct commands for the SDRAM, and manages the details of the SDRAM access.

Features of the SDRAM Controller Subsystem

The SDRAM controller subsystem offers the following features:

- Support for double data rate 2 (DDR2), DDR3, and low-power DDR2 (LPDDR2) SDRAM
- User-configurable timing parameters
- Up to 4 Gb density parts
- Two chip selects
- Integrated error correction code (ECC), 24- and 40-bit widths
- User-configurable memory width of 8, 16, 16+ECC, 32, 32+ECC
- Command reordering (look-ahead bank management)
- Data reordering (out of order transactions)
- User-controllable bank policy on a per port basis for either closed page or conditional open page accesses
- User-configurable priority support with both absolute and relative priority scheduling
- Flexible FPGA fabric interface configuration with up to 6 ports and data widths up to 256 bits wide using Avalon-MM and AXI interfaces.
- Power management supporting self refresh, partial array self-refresh (PASR), power down, and LPDDR2 deep power down

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

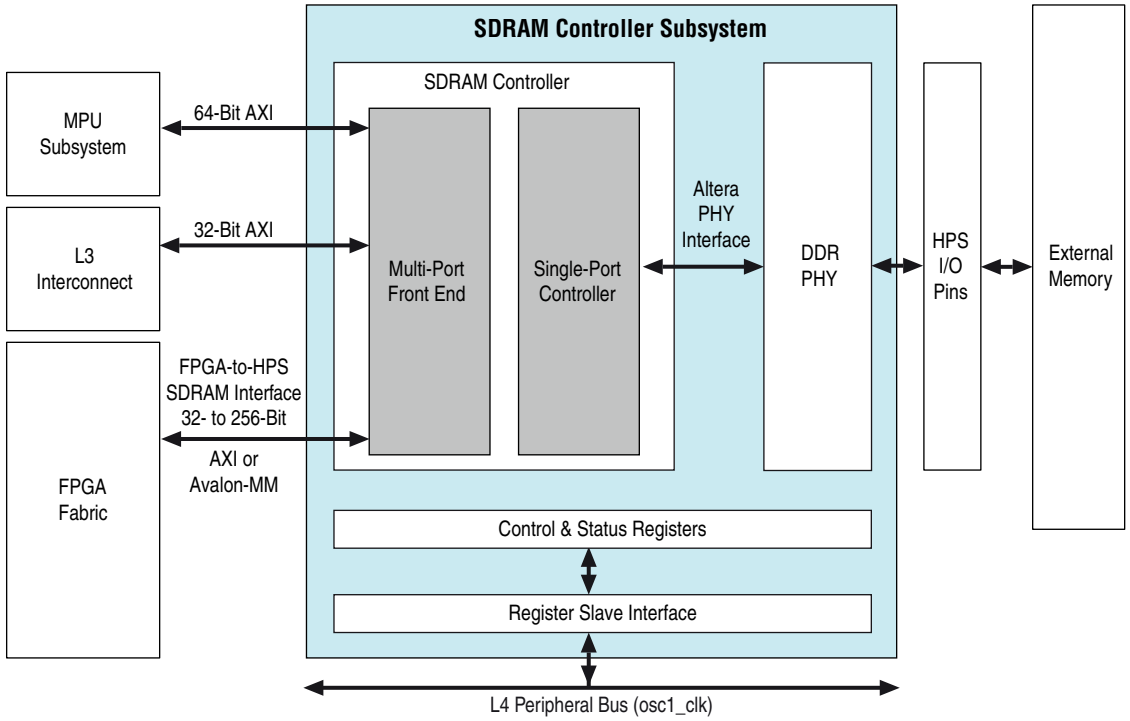
ISO 9001:2008 Registered

SDRAM Controller Subsystem Block Diagram and System Integration

The SDRAM controller subsystem connects to the MPU subsystem, the main switch of the L3 interconnect, and the FPGA fabric. The memory interface consists of the SDRAM controller, the physical layer (PHY), control and status registers (CSRs), and their associated interfaces.

Figure 4–1 shows a high-level block diagram of the SDRAM controller subsystem.

Figure 4–1. SDRAM Controller Subsystem High-Level Block Diagram



SDRAM Controller

The SDRAM controller provides high performance data access and run-time programmability. The controller reorders data to reduce row conflicts and bus turn-around time by grouping read and write transactions together, allowing for efficient traffic patterns and reduced latency.

The SDRAM controller consists of a multiport front end (MPFE) and a single-port controller. The MPFE provides multiple independent interfaces to the single-port controller. The single-port controller communicates with and manages each external memory device. For more information, refer to "Memory Controller Architecture" on page 4–4.

DDR PHY

The DDR PHY provides a physical layer interface between the memory controller and memory devices, which performs read and write memory operations. The DDR PHY has dataflow components, control components, and calibration logic that handle the calibration for the SDRAM interface timing.

SDRAM Controller Subsystem Interfaces

The following sections describe the SDRAM controller subsystem interfaces.

MPU Subsystem Interface

The SDRAM controller is connected to the MPU subsystem with a dedicated 64-bit AXI interface, operating on the `mpu_l2_ram_clk` clock domain.

L3 Interconnect Interface

The SDRAM controller is connected to the L3 interconnect with a dedicated 32-bit AXI interface, operating on the `l3_main_clk` clock domain.

CSR Interface

The CSR interface is connected the level 4 (L4) bus and operates on the `l4_sp_clk` clock domain. The MPU subsystem uses the CSR interface to configure the controller and PHY, for example, setting the memory timing parameter values or placing the memory to a low power state. The CSR interface also provides access to the status registers in the controller and PHY.

FPGA-to-HPS SDRAM Interface

The FPGA-to-HPS SDRAM interface provides masters implemented in the FPGA fabric access to the SDRAM controller subsystem in the HPS. The interface has three ports types that are used to construct the following AXI or Avalon-MM interfaces:

- Command ports—issue read and write commands, and for receive write acknowledge responses
- 64-bit read data ports—receive data returned from a memory read
- 64-bit write data ports—transmit write data

The FPGA-to-HPS SDRAM interface supports six command ports, allowing up to six Avalon-MM interfaces or three AXI interfaces. Each command port can be used to implement either a read or write command port for AXI, or be used as part of an Avalon-MM interface. The AXI and Avalon-MM interfaces can be configured to support 32-, 64-, 128-, and 256-bit data.

Table 4–1 lists the FPGA-to-HPS SDRAM controller interface ports connected to the FPGA.

Table 4–1. FPGA-to-HPS SDRAM Controller Port Types

Port Type	Number
Command	6
64-bit read data	4
64-bit write data	4

The FPGA-to-HPS SDRAM controller interface can be configured with the following characteristics:

- Avalon-MM interfaces and AXI interfaces can be mixed and matched as required by the fabric logic, within the bounds of the number of ports provided to the fabric.

- Each Avalon-MM or AXI interface of the FPGA-to-HPS SDRAM interface operates on an independent clock domain.
- The FPGA-to-HPS SDRAM interfaces are configured during FPGA configuration.

Table 4–2 shows the number of ports needed to configure different bus protocols, based on type and data width.

Table 4–2. FPGA-to-HPS SDRAM Port Utilization

Bus Protocol	Command	Read Data	Write Data
32- or 64-bit AXI	2 (1)	1	1
128-bit AXI	2 (1)	2 (2)	2 (2)
256-bit AXI	2 (1)	4 (2)	4 (2)
32- or 64-bit Avalon-MM	1	1	1
128-bit Avalon-MM	1	2	2
256-bit Avalon-MM	1	4	4
32- or 64-bit Avalon-MM write-only	1	0	1
128-bit Avalon-MM write-only	1	0	2
256-bit Avalon-MM write-only	1	0	4
32- or 64-bit Avalon-MM read-only	1	1	0
128-bit Avalon-MM read-only	1	2	0
256-bit Avalon-MM read-only	1	4	0

Notes to Table 4–2:

(1) Because the AXI protocol allows simultaneous read and write commands to be issued, two SDRAM control ports are required to form an AXI interface.

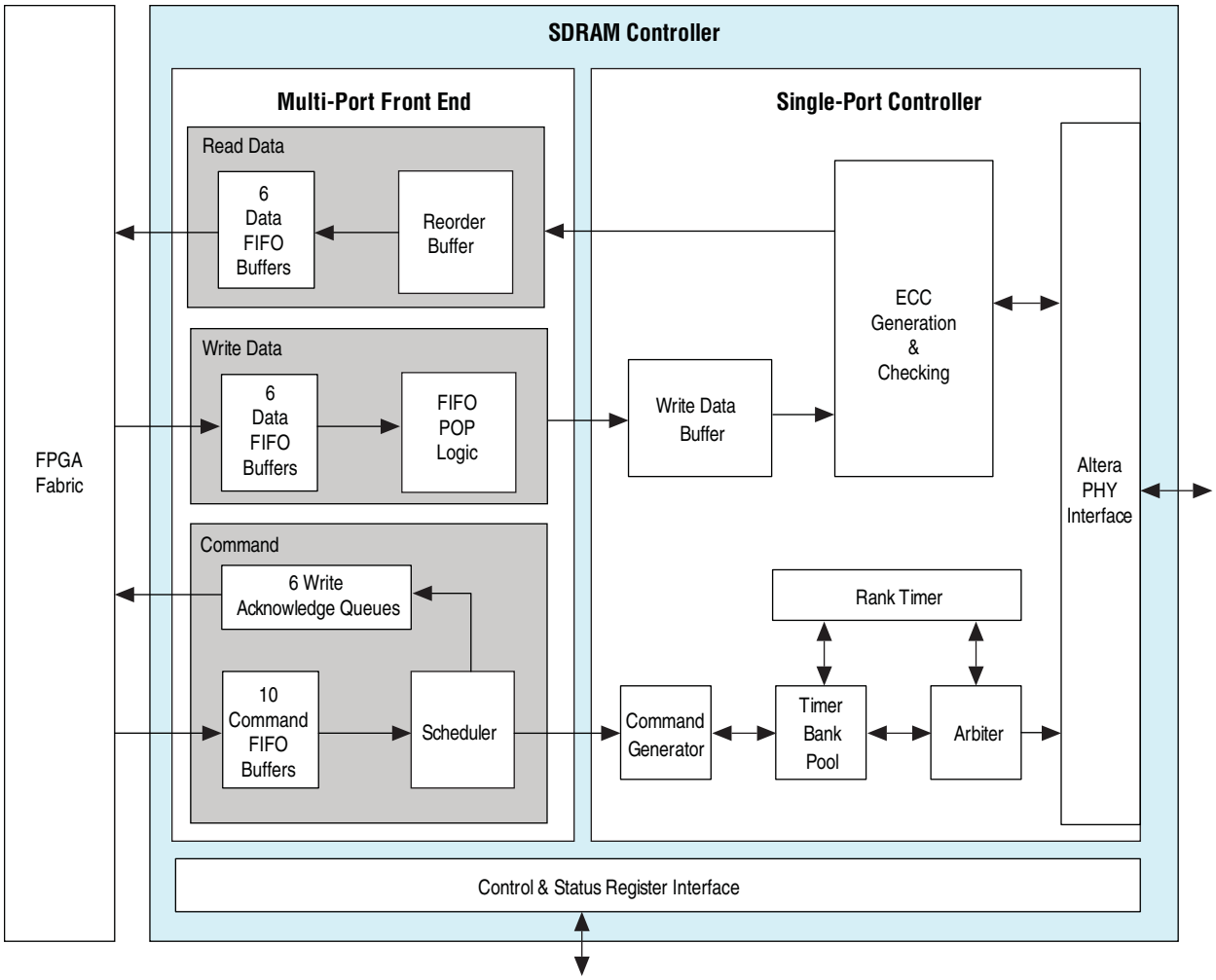
(2) Because the native size of the data ports is 64 bits, extra read and write ports are required to form an AXI interface.

Memory Controller Architecture

The SDRAM controller consists of an MPFE, a single-port controller, and an interface to the CSRs.

Figure 4–2 shows a block diagram of the SDRAM controller portion of the SDRAM controller subsystem.

Figure 4–2. SDRAM Controller Block Diagram



MPFE

The MPFE is responsible for scheduling pending transactions from the configured interfaces and sending the scheduled memory transactions to the single-port controller. The MPFE handles all functions related to individual ports.

The MPFE consists of the following three primary sub-blocks.

Command Block

The command block accepts read and write transactions from the FPGA fabric and the HPS. When the command FIFO buffer is full, the command block applies backpressure by deasserting the ready signal. For each pending transaction, the command block calculates the next SDRAM burst needed to progress on that transaction. The command block schedules pending SDRAM burst commands based on the user-supplied configuration, available write data, and unallocated read data space.

Write Data Block

The write data block transmits data to the single-port controller. The write data block maintains write data FIFO buffers and clock boundary crossing for the write data. The write data block informs the command block of the amount of pending write data for each transaction so that the command block can calculate eligibility for the next SDRAM write burst.

Read Data Block

The read data block receives data from the single-port controller. Depending on the port state, the read data block either buffers the data in its internal buffer or passes the data straight to the clock boundary crossing FIFO buffer. The read data block reorders out-of-order data for Avalon-MM ports.

In order to prevent the read FIFO buffer from overflowing, the read data block informs the command block of the available buffer area so the command block can pace read transaction dispatch.

Single-Port Controller

The single-port logic is responsible for following actions:

- Queuing the pending SDRAM bursts
- Choosing the most efficient burst to send next
- Keeping the SDRAM pipeline full
- Ensuring all SDRAM timing parameters are met

Transactions passed to the single-port logic for a single page in SDRAM are guaranteed to be executed in order, but transactions can be reordered between pages. Each SDRAM burst read or write is converted to the appropriate Altera PHY interface (AFI) command to open a bank on the correct row for the transaction (if required), execute the read or write command, and precharge the bank (if required).

The single-port logic implements command reordering (looking ahead at the command sequence to see which banks can be put into the correct state to allow a read or write command to be executed) and data reordering (allowing data transactions to be dispatched even if the data transactions are executed in an order different than they were received from the multiport logic).

Command Generator

The command generator accepts commands from the MPFE and from the internal ECC logic, and provides those commands to the timer bank pool.

Timer Bank Pool

The timer bank pool is a parallel queue that operates with the arbiter to enable data reordering. The timer bank pool tracks incoming requests, ensures that all timing requirements are met, and, on receiving write-data-ready notifications from the write data buffer, passes the requests to the arbiter.

Arbiter

The arbiter determines the order in which requests are passed to the memory device. When the arbiter receives a single request, that request is passed immediately. When multiple requests are received, the arbiter uses arbitration rules to determine the order to pass requests to the memory device.

Rank Timer

The rank timer performs the following functions:

- Maintains rank-specific timing information
- Ensures that only four activates occur within a specified timing window
- Manages the read-to-write and write-to-read bus turnaround time
- Manages the time-to-activate delay between different banks

Write Data Buffer

The write data buffer receives write data from the MPFE and passes the data to the PHY, on approval of the write request.

ECC Block

The ECC block consists of an encoder and a decoder-corrector, which can detect and correct single-bit errors, and detect double-bit errors. The ECC block can correct single- bit errors and detect double-bit errors resulting from noise or other impairments during data transmission.

AFI Interface

The AFI interface provides communication between the controller and the PHY.

CSR Interface

The CSR interface is accessible from the L4 bus. The interface allows code executing in the HPS MPU and FPGA fabric to configure and monitor the SDRAM controller.

Functional Description of the SDRAM Controller Subsystem

This section provides a functional description of the SDRAM controller subsystem.

MPFE Operational Behavior

This section describes the operational behavior of the MPFE.

Operation Ordering

Requests to the same SDRAM page arriving at a given port are executed in the order in which they are received. Requests arriving at different ports have no guaranteed order of service, except when a first transaction has completed before the second arrives.

Operation ordering is defined and enforced within a port, but not between ports. All transactions received on a single port for overlapping addresses execute in order. Transactions received on different ports have no guaranteed order unless the second transaction is presented after the first has completed.

Avalon-MM does not support write acknowledgement. When a port is configured to support Avalon-MM, you should read from the location that was previously written to ensure that the write operation has completed. When a port is configured to support AXI, the master accessing the port can safely issue a read operation to the same address as a write operation as soon as the write has been acknowledged. To keep write latency low, writes are acknowledged as soon as the transaction order is guaranteed—meaning that any operations received on any port to the same address as the write operation are executed after the write operation.

To ensure that the overall latency of traffic is as low as possible, the single port logic can return read data out of order to the multi-port logic which will reorder it when transactions return out of order. A large percentage of traffic reordering will be between ports and transactions only are ordered within a port. For traffic which is reordered between ports but not within a port, no reordering needs to be done. Eliminating unnecessary reordering reduces average latency.

Multiport Scheduling

Multiport scheduling is governed by two factors, the absolute priority of a request and the weighting of a port.

The evaluation of absolute priority ensures that ports carrying higher-priority traffic are served ahead of ports carrying lower-priority traffic. The scheduler recognizes eight priority levels (0-7), with higher values representing higher priorities. For example, any transaction with priority seven is scheduled before transactions of priority six or lower.

When ports carry traffic of the same absolute priority, relative priority is determined based on port weighting. Port weighting is a five-bit value (0-31), and is determined by a deficit-weighted round robin (DWRR) algorithm, which corrects for past over-servicing or under-servicing of a port. Each port has an associated weight which is updated every cycle, with a user-configured weight added to it and the amount of traffic served subtracted from it. The port with the highest weighting is considered the most eligible.

To ensure that high-priority traffic is served quickly and that long and short bursts are effectively interleaved between ports, incoming transactions longer than a single SDRAM burst are scheduled as a series of SDRAM bursts, with each burst arbitrated separately.

To ensure that lower priority ports do not build up large running weights while higher priority ports monopolize bandwidth, the controller's DWRR weights are updated only when a port matches the scheduled priority. Therefore, if three ports are being accessed, two being priority seven and one being priority four, the weights for both ports at priority seven are updated but the port with priority four remains unchanged.

Multiport scheduling is performed between all of the ports connected to the FPGA fabric and internally in the HPS to determine which transaction is serviced next. Arbitration is performed on a SDRAM burst basis to ensure that a long transaction does not lock other transactions or cause latency to significantly increase for high-priority ports.

Arbitration supports both absolute and relative priority. Absolute priority is intended for applications where one master should always get priority above or below others. Relative priority is supported through a programmable weight field which controls scheduling between ports at the same priority.

The scheduler is work-conserving. Write operations can only be scheduled when enough data for the SDRAM burst has been received. Read operations can only be scheduled when sufficient internal memory is free and the port is not occupying too much of the read buffer.

The multiport scheduling configuration can be updated while traffic is flowing. Both priority and weight for a port can be updated without interrupting traffic on a port. Updates are used in scheduling decisions within 10 memory clock cycles of being updated, so priority can be updated frequently if needed.

Read Data Handling

The MPFE contains a read buffer shared by all ports. If a port is capable of receiving returned data then the read buffer is bypassed. If the size of a read transaction is smaller than twice the memory interface width, the buffer RAM cannot be bypassed.

SDRAM Burst Scheduling

SDRAM burst scheduling recognizes addresses that access the same row/bank combination, known as open page accesses. Operations to a page are served in the order in which they are received by the single-port controller.

Selection of SDRAM operations is a two-stage process. First, each pending transaction must wait for its timers to be eligible for execution. Next, the transaction arbitrates against other transactions that are also eligible for execution.

The following rules govern transaction arbitration:

- High-priority operations take precedence over lower-priority operations
- If multiple operations are in arbitration, read operations have precedence over write operations
- If multiple operations still exist, the oldest is served first

A high-priority transaction in the SDRAM burst scheduler wins arbitration for that bank immediately if the bank is idle and the high-priority transaction's chip select, row, or column fields of the address do not match an address already in the single-port controller. If the bank is not idle, other operations to that bank yield until the high-priority operation is finished. If the chip select, row, and column fields match an earlier transaction, the high-priority transaction yields until the earlier transaction is completed.

Clocking

The FPGA fabric ports of the MPFE can be clocked at different frequencies. Synchronization is maintained by clock-domain crossing logic in the MPFE. Command ports can operate on different clock domains, but the data ports associated with a given command port must be attached to the same clock as that command port. For example, a command port paired with a read and write port to form an Avalon-MM interface must operate at the same clock frequency as the data ports associated with it.

Single-Port Controller Operational Behavior

This section describes the operational behavior of the single-port controller.

SDRAM Interface

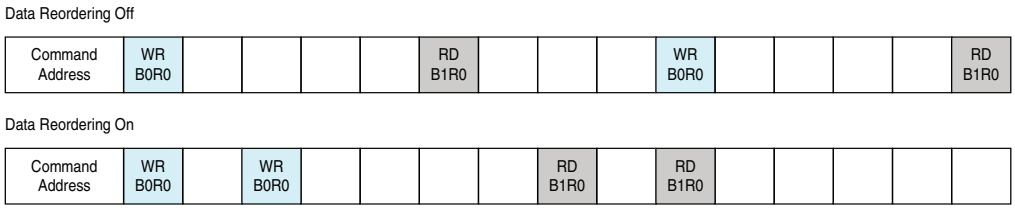
The SDRAM interface is up to 40 bits wide and can accommodate 8-bit, 16-bit, 16-bit plus ECC, 32-bit, or 32-bit plus ECC configurations. The SDRAM interface supports LPDDR2, DDR2, and DDR3 memory protocols.

Command and Data Reordering

The heart of the SDRAM controller is a command and data reordering engine. Command reordering allows banks for future transactions to be opened before the current transaction finishes. Data reordering allows transactions to be serviced in a different order than they were received when that new order allows for improved utilization of the SDRAM bandwidth. Operations to the same bank and row are performed in order to ensure that operations which impact the same address preserve the data integrity.

Figure 4–3 shows the relative timing for a write/read/write/read command sequence performed in order and then the same command sequence performed with data reordering. Data reordering allows the write and read operations to occur in bursts, without bus turnaround timing delay or bank reassignment.

Figure 4–3. Data Reordering Effect



The SDRAM controller schedules among all pending row and column commands every clock cycle.

Bank Policy

The bank policy of the SDRAM controller allows users to request that a transaction's bank remain open after an operation has finished so that future accesses do not delay in activating the same bank and row combination. The controller supports only eight simultaneously-opened banks, so an open bank might get closed if the bank resource is needed for other operations.

Open bank resources are allocated dynamically as SDRAM burst transactions are scheduled. Bank allocation is requested automatically by the controller when an incoming transaction spans multiple SDRAM bursts or by the extended command interface. When a bank must be reallocated, the least-recently-used open bank is used as the replacement.

If the controller determines that the next pending command will cause the bank request to not be honored, the bank might be held open or closed depending on the pending operation. A request to close a bank with a pending operation in the timer bank pool to the same row address causes the bank to remain open. A request to leave a bank open with a pending command to the same bank but a different row address causes a precharge operation to occur.

Write Combining

The SDRAM controller combines write operations from successive bursts on a port where the starting address of the second burst is one greater than the ending address of the first burst and the resulting burst length does not overflow the 11-bit burst-length counters. Write combining does not occur if the previous bus command has finished execution before the new command has been received.

Burst Length Support

The controller supports burst lengths of 2, 4, 8, and 16, and data widths of 8, 16, and 32 bits for non-ECC operation, and widths of 24 and 40 operations with ECC enabled. Table 4–3 shows the type of SDRAM for each burst length.

Table 4–3. SDRAM Burst Lengths

Burst Length	SDRAM
4	LPDDR2, DDR2
8	DDR2, DDR3, LPDDR2
16	LPDDR2

Width Matching

The SDRAM controller automatically performs data width conversion.

ECC

The single-port controller supports memory ECC calculated by the controller. The controller ECC employs standard Hamming logic to detect and correct single-bit errors and detect double-bit errors. The controller ECC is available for 16-bit and 32-bit widths, each requiring an additional 8 bits of memory, resulting in an actual memory width of 24-bits and 40-bits, respectively.

Controller ECC provides the following features:

- Byte writes—The memory controller performs a read-modify-write operation to ensure that the ECC data remains valid when a subset of the bits of a word is being written. If an entire word is being written (but less than a full burst) and the DM pins are connected, no read is necessary and only that word is updated. If controller ECC is disabled, byte-writes have no performance impact.

- ECC write backs—When a read operation detects a correctable error, the memory location is scheduled for a read-modify-write operation to correct the single-bit error. ECC write backs are enabled and disabled through the `cfg_enable_ecc_code_overwrites` field in the `ctrlcfg` register.
- Notification of ECC errors—The memory controller provides interrupts for single-bit and double-bit errors. The status of interrupts and errors are recorded in status registers, as follows:
  - The `dramsts` register records interrupt status.
  - The `dramintr` register records interrupt masks.
  - The `sbecount` register records the single-bit error count.
  - The `dbecount` register records the double-bit error count.
  - The `erraddr` register records the address of the most recent error.

Byte Writes

Byte writes with ECC enabled are executed as a read-modify-write. Typical operations only use a single entry in the timer bank pool. Controller ECC enabled sub-word writes use two entries. The first operation is a read and the second operation is a write. These two operations are transferred to the timer bank pool with an address dependency so that the write cannot be performed until the read data has returned. This approach ensures that any subsequent operations to the same address (from the same port) are executed after the write operation, because they are ordered on the row list after the write operation.

If an entire word is being written (but less than a full burst), then no read is necessary and only that word is updated.

ECC Write Backs

If the controller ECC is enabled and a read operation results in a correctable ECC error, the controller corrects the location in memory, if write backs are enabled. The correction results in scheduling a new read-modify-write. A new read is performed at the location to ensure that a write operation modifying the location is not overwritten. The actual ECC correction operation is performed as a read-modify-write operation.

User Notification of ECC Errors

The following methods notify you of an ECC error:

For the MPU subsystem, an interrupt signal provides notification and the ECC error information is stored in the status registers.

For more information, refer to the *Cortex-A9 Microprocessor Unit Subsystem* chapter in volume 3 of the *Arria V Device Handbook* or the *Cortex-A9 Microprocessor Unit SubSystem* chapter in volume 3 of the *Cyclone V Device Handbook*.

Interleaving Options

The controller supports the following address-interleaving options:

- Noninterleaved
- Bank interleave without chip select interleave

- Bank interleave with chip select interleave

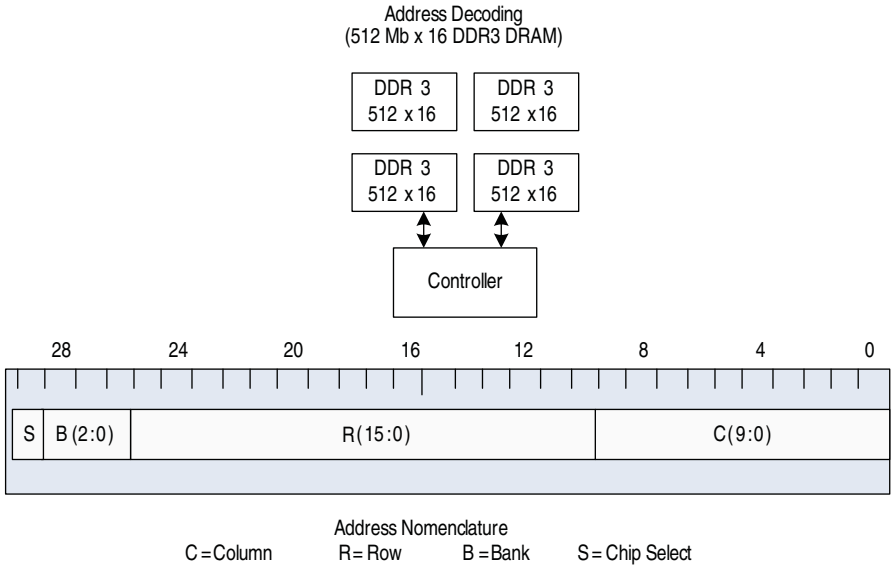
All of the interleaving examples use 512 megabits (Mb) x 16 DDR3 chips and are documented as byte addresses. For RAMs with smaller address fields, the order of the fields stays the same but the widths may change.

Noninterleaved

RAM mapping is noninterleaved.

Figure 4–4 shows noninterleaved address decoding.

Figure 4–4. Noninterleaved Address Decoding

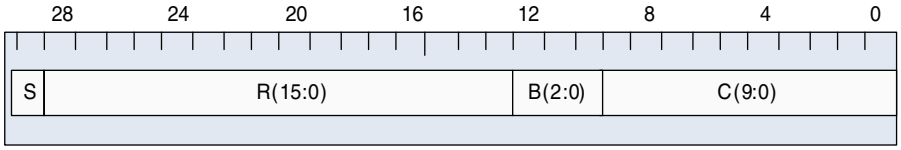


Bank Interleave Without Chip Select Interleave

Bank interleave without chip select interleave swaps row and bank from the noninterleaved address mapping. This interleaving allows smaller data structures to spread across all banks in a chip.

Figure 4–5 shows bank interleave without chip select interleave address decoding.

Figure 4–5. Bank Interleave Without Chip Select Interleave Address Decoding

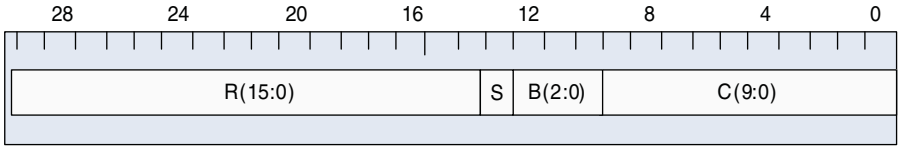


Bank Interleave with Chip Select Interleave

Bank interleave with chip select interleave moves the row address to the top, followed by chip select, then bank, and finally column address. This interleaving allows smaller data structures to spread across multiple banks and chips (giving access to 16 total banks for multithreaded access to blocks of memory). Memory timing is degraded when switching between chips.

Figure 4–6 shows bank interleave with chip select interleave address decoding.

Figure 4–6. Bank Interleave With Chip Select Interleave Address Decoding



AXI-Exclusive Support

The single-port controller supports AXI-exclusive operations. The controller implements a table shared across all masters, which can store up to 16 pending writes. Table entries are allocated on an exclusive read and table entries are deallocated on a successful write to the same address by any master.

Any exclusive write operation that is not present in the table returns an exclusive fail as acknowledgement to the operation. If the table is full when the exclusive read is performed, the table replaces a random entry.

When using AXI-exclusive operations, accessing the same location from Avalon-MM interfaces can result in unpredictable results.

Memory Protection

The single-port controller has address protection to allow the software to configure basic protection of memory from all masters in the system. If the system has been designed exclusively with AXI masters, TrustZone® is supported. Ports that use Avalon-MM can be configured for port level protection.

For information about TrustZone®, refer to the ARM website (www.arm.com).

Memory protection is based on physical addresses in memory. You can set rules to allow or disallow accesses to a range of memory, or to enable only secure accesses to a range of memory (or a combination of the two).

Secure and non-secure regions are specified by rules containing a starting address and ending address with 1 MB boundaries for both addresses. You can override the port defaults and allow or disallow all transactions.

The memory protection table, which is an internal table addressed through the CSR interface, contains rules to permit or deny memory access. You can configure up to a maximum of twenty rules to control memory access. Table 4–4 lists the fields that you can specify for each rule.

Table 4–4. Fields for Rules in Memory Protection Table (Part 1 of 2)

Field	Width	Description
Valid	1	Set to 1 to activate the rule. Set to 0 to deactivate the rule.
Port Mask (1)	10	Specifies the set of ports to which the rule applies, with one bit representing each port, as follows: bits 0 to 5 correspond to FPGA fabric ports 0 to 5, bit 6 corresponds to AXI L3 switch read, bit 7 is the CPU read, bit 8 is L3 switch write, and bit 9 is the CPU write.

Table 4–4. Fields for Rules in Memory Protection Table (Part 2 of 2)

Field	Width	Description
TID_low (1)	12	Low transfer ID of the rules to which this rule applies. Incoming transactions match if they are greater than or equal to this value. Ports with smaller TIDs have the TID shifted to the lower bits and zero padded at the top.
TID_high (1)	12	High transfer ID of the rules to which this rule applies. Incoming transactions match if they are less than or equal to this value.
Address_low	12	Points to a 1MB block and is the lower address. Incoming addresses match if they are greater than or equal to this value.
Address_high	12	Upper limit of address. Incoming addresses match if they are less than or equal to this value.
Protection	2	A value of 00 indicates that the protection bit is not set; a value of 01 sets the protection bit. Systems that do not set AXI protection to a known value should program this for either protection value.
Fail/allow	1	Set this value to 1 to force the operation to fail or succeed.

Note to Table 4–4:

(1) Although TID and Port Mask could be redundant, including both in the table allows possible compression of rules. If masters connected to a port do not have contiguous TIDs, a port-based rule might be more efficient than a TID-based rule, in terms of the number of rules needed.

A port has a default access status of either allow or fail, and rules with the opposite allow/fail value can override the default. The system evaluates each transaction against every rule in the memory protection table. A transaction received on a port which by default allows access, would fail only if a rule with the fail bit matches the transaction. Conversely, a port which by default prevents access, would allow access only if a rule allows that transaction to pass.

Exclusive transactions are security checked on the read operation only. A write operation can occur only if a valid read is marked in the internal exclusive table. Consequently, a master performing an exclusive read followed by a write, can write to memory only if the exclusive read was successful.

Example of Configuration for TrustZone

For a TrustZone configuration, memory is divided into a range of memory accessible by secure masters and a range of memory accessible by nonsecure masters. The two memory address ranges may have a range of memory that overlaps.

This example implements the following memory configuration:

- 2 GB total RAM size
- 0—512 MB dedicated secure area
- 513—576 MB shared area
- 577—2048 MB dedicated nonsecure area

In this example, each port is configured by default to disallow all accesses. Table 4–5 shows the two rules programmed into the memory protection table.

Table 4–5. Rules in Memory Protection Table for Example Configuration

Rule #	Port Mask	TID Low	TID High	Address Low	Address High	Prot	Fail/Allow
1	0'b1111111111	0	4095	0	576	b01	allow
2	0'b1111111111	0	4095	512	2047	b00	allow

The port mask value, TID Low, and TID High, apply to all ports and all transfers within those ports. Each access request is evaluated against the memory protection table, and will fail unless a rule matches allowing a transaction to complete successfully.

Table 4–6 shows the result for a sample set of transactions.

Table 4–6. Result for a Sample Set of Transactions

Operation	Source	Address	Prot	Result	Comments
Read	CPU	4096	1	Allow	Matches rule 1.
Write	CPU	536, 870, 912 (512 MB)	1	Allow	Matches rule 1.
Write	L3 attached masters	605, 028, 350 (577 MB)	1	Fail	Does not match rule 1 (out of range of the address field), does not match rule 2 (protection bit incorrect).
Read	L3 attached masters	4096	0	Fail	Does not match rule 1 (prot value wrong), does not match rule 2 (not in address range).
Write	CPU	536, 870, 912 (512 MB)	0	Allow	Matches rule 2.
Write	L3 attached masters	605, 028, 350 (577 MB)	0	Allow	Matches rule 2.

If you did not want any overlap between the memory blocks, you could specify the address ranges in the two rules of Table 4–5 to be mutually exclusive. Depending on your desired TrustZone configuration, you can add rules to the memory protection table to create multiple blocks of protected or unprotected space.

SDRAM Power Management

The SDRAM controller subsystem supports the following power saving features in the SDRAM:

- Partial array self-refresh (PASR)
- Power down
- Deep power down for LPDDR2

Power-saving mode initiates either due to a user command or from inactivity.

Power-down mode is initiated by writing to the appropriate control register. It forces the SDRAM burst-scheduling bank-management logic to close all banks and issue the power down command. You can program the controller to enable power-down when the SDRAM burst-scheduling queue is empty for a specified number of clock cycles. The SDRAM automatically reactivates when an active SDRAM command is received.

Other power-down modes are performed only under user control.

DDR PHY

The DDR PHY connects the memory controller and external memory devices in the speed critical command path.

The DDR PHY implements the following functions:

- Calibration—the DDR PHY supports the JEDEC-specified steps to synchronize the memory timing between the controller and the SDRAM chips. The calibration algorithm is implemented in software.
- Memory device initialization—the DDR PHY performs the mode register write operations to initialize the devices. The DDR PHY handles re-initialization after a deep power down.
- Single-data-rate to double-data-rate conversion.

Clocks

All clocks are assumed to be asynchronous with respect to the `ddr_dqs_clk` memory clock. All transactions are synchronized to memory clock domain.

Table 4–7 shows the SDRAM controller subsystem clock domains.

Table 4–7. SDRAM Controller Subsystem Clock Domains

Clock Name	Description
`ddr_dq_clk`	Clock for PHY
`ddr_dqs_clk`	Clock for MPFE, single-port controller, CSR access, and PHY
`ddr_2x_dqs_clk`	Clock for PHY
`l4_sp_clk`	Clock for CSR interface
`mpu_l2_ram_clk`	Clock for MPU interface
`l3_main_clk`	Clock for L3 interface
`f2h_sdram_clk[5:0]`	Six separate clocks used for the FPGA-to-HPS SDRAM ports to the FPGA fabric

In terms of clock relationships, the FPGA fabric connects the appropriate clocks to write data, read data, and command ports for the constructed ports.

For more information, refer to the *Clock Manager* chapter in volume 3 of the *Arria V Device Handbook* or the *Clock Manager* chapter in volume 3 of the *Cyclone V Device Handbook*.

Resets

The SDRAM controller subsystem supports a full reset (cold reset) and a warm reset, which may or may not preserve the contents of memory. In order to preserve the memory contents, the reset manager can request that the single-port controller place the SDRAM in self-refresh mode prior to issuing the warm reset. If memory contents are preserved, the PHY and the memory timing logic is not reset, but the rest of the controller is reset.

For more information, refer to the *Reset Manager* chapter in volume 3 of the *Arria V Device Handbook* or the *Reset Manager* chapter in volume 3 of the *Cyclone V Device Handbook*.

Initialization

The SDRAM controller subsystem has CSRs which control the operation of the controller including DRAM type, DRAM timing parameters and relative port priorities. It also has a small set of bits which depend on the FPGA fabric to configure ports between the memory controller and the FPGA fabric; these bits are set for you when you configure your implementation using the HPS GUI in Qsys.

The CSRs are configured using a dedicated slave interface, which provides accesses to registers. This region controls all SDRAM operation, MPFE scheduler configuration, and PHY calibration.

The FPGA fabric interface configuration is programmed into the FPGA fabric and the values of these register bits can be read by software. The ports can be configured without software developers needing to know how the FPGA-to-HPS SDRAM interface has been configured.

Protocol Details

Avalon-MM Bidirectional Port

The Avalon-MM bidirectional ports are standard Avalon-MM ports used to dispatch read and write operations. Each configured Avalon-MM bidirectional port consists of the signals listed in Table 4–8.

Table 4–8. Avalon-MM Bidirectional Port Signals (Part 1 of 2)

Name	Bits	Direction	Function
`clk`	1	In	Clock for the Avalon-MM interface
`read`	1	In	Indicates read transaction
`write`	1	In	Indicates write transaction
`address`	32	In	Address of the transaction
`readdata`	32, 64, 128, or 256	Out	Read data return
`readdatavalid`	1	Out	Valid cycle flag for read data return
`writedata`	32, 64, 128, or 256	In	Write data for a transaction

Table 4–8. Avalon-MM Bidirectional Port Signals (Part 2 of 2)

Name	Bits	Direction	Function
byteenable	4 (32-bit data), 8(64-bit data), 16(128-bit data), 32(256-bit data)	In	Byte enables for each write byte
waitrequest	1	Out	Indicates need for additional cycles to complete a transaction
burstcount	11	In	Transaction burst length

The read and write interfaces are configured to the same size. The byte-enable size scales with the data bus size.

For information about the Avalon-MM protocol, refer to the *Avalon Interface Specifications*.

Avalon-MM Write Port

The Avalon-MM write ports are standard Avalon-MM ports used only to dispatch write operations. Each configured Avalon-MM write port consists of the signals listed in Table 4–9.

Table 4–9. Avalon-MM Write Port Signals

Name	Bits	Direction	Function
reset	1	In	Reset
clk	1	In	Clock
write	1	In	Indicates write transaction
address	32	In	Address of the transaction
writedata	32, 64, 128, or 256	In	Write data for a transaction
byteenable	4 (32-bit data), 8(64-bit data), 16(128-bit data), 32(256-bit data)	In	Byte enables for each write byte
waitrequest	1	Out	Indicates need for additional cycles to complete a transaction
burstcount	11	In	Transaction burst length

For information about the Avalon-MM protocol, refer to the *Avalon Interface Specifications*.

Avalon-MM Read Port

The Avalon-MM read ports are standard Avalon-MM ports used only to dispatch read operations. Each configured Avalon-MM read port consists of the signals listed in Table 4–10.

Table 4–10. Avalon-MM Read Port Signals

Name	Bits	Direction	Function
reset	1	In	Reset
clk	1	In	Clock
read	1	In	Indicates read transaction
address	32	In	Address of the transaction
readdata	32, 64, 128, or 256	Out	Read data return
readdatavalid	1	Out	Flags valid cycles for read data return
waitrequest	1	Out	Indicates the need for additional cycles to complete a transaction. Needed for read operations when delay is needed to accept the read command.
burstcount	11	In	Transaction burst length

For information about the Avalon-MM protocol, refer to the *Avalon Interface Specifications*.

AXI Port

The AXI port uses an AXI-3 interface.

For information about the AXI-3 interface, refer to the *AMBA Open Specifications* on the ARM website (www.arm.com).

For information about the AXI interface ports in the high-performance II controller (HPC II), refer to the *Functional Description—HPC II Controller* chapter, in the *External Memory Interface Handbook*.

Each configured AXI port consists of the signals listed in Table 4–11. Each AXI interface signal is independent of the other interfaces for all signals, including clock and reset.

Table 4–11. AXI Port Signals (Part 1 of 2)

Name	Bits	Direction	Function
ARESETn	1	In	Reset
ACLK	1	In	Clock
Write Address Channel Signals			
AWID	4	In	Write identification tag
AWADDR	32	In	Write address
AWLEN	4	In	Write burst length
AWSIZE	3	In	Width of the transfer size
AWBURST	2	In	Burst type

Table 4–11. AXI Port Signals (Part 2 of 2)

Name	Bits	Direction	Function
AWREADY	1	Out	Indicates ready for a write command
AWVALID	1	In	Indicates valid write command.
Write Data Channel Signals			
WID	4	In	Write data transfer ID
WDATA	32, 64, 128 or 256	In	Write data
WSTRB	4, 8, 16, 32	In	Byte-based write data strobe. Each bit width corresponds to 8 bit wide transfer for 32-bit wide to 256-bit wide transfer.
WLAST	1	In	Last transfer in a burst
WVALID	1	In	Indicates write data+strobes are valid
WREADY	1	Out	Indicates ready for write data and strobes
Write Response Channel Signals			
BID	4	Out	Write response transfer ID
BRESP	2	Out	Write response status
BVALID	1	Out	Write response valid signal
BREADY	1	In	Write response ready signal
Read Address Channel Signals			
ARID	4	In	Read identification tag
ARADDR	32	In	Read address
ARLEN	4	In	Read burst length
ARSIZE	3	In	Width of the transfer size
ARBURST	2	In	Burst type
ARREADY	1	Out	Indicates ready for a read command
ARVALID	1	In	Indicates valid read command
Read Data Channel Signals			
RID	4	Out	Read data transfer ID
RDATA	32, 64, 128 or 256	Out	Read data
RRESP	2	Out	Read response status
RLAST	1	Out	Last transfer in a burs
RVALID	1	Out	Indicates read data is valid
RREADY	1	In	Read data channel ready signal

SDRAM Controller Subsystem Programming Model

Initialization

SDRAM controller configuration occurs through software programming of the configuration registers using the CSR interface. Initialization of the SDRAM controller has two separate regions with different controls.

Timing Parameters

The SDRAM controller supports a complete set of timing parameters, configurable at run time.

SDRAM Controller Address Map and Register Definitions

To access address map and register definitions, open the file hps.html.

To view the module description and base address, scroll to and click the link for the following module instance:

- **sdr**

To then view the register and field descriptions, scroll to and click the register names. The register addresses are offsets relative to the base address of each module instance.

The base addresses of all modules are also listed in the *Introduction to the Hard Processor System* chapter in volume 3 of the *Arria V Device Handbook* and the *Introduction to the Hard Processor System* chapter in volume 3 of the *Cyclone V Device Handbook*.

Using EMI-Related HPS Features in SoC Devices

This section provides information on using HPS features in your external memory interface.

Architecture

The configuration and initialization of the memory interface by the ARM processor is a significant difference compared to the FPGA memory interfaces, and results in several key differences in the way the HPS memory interface is defined and configured.

Boot-up configuration of the HPS memory interface is handled by the initial software boot code, not by the FPGA programmer, as is the case for the FPGA memory interfaces. The Quartus II software is involved in defining the configuration of I/O ports which is used by the boot-up code, as well as timing analysis of the memory interface. Therefore, the memory interface must be configured with the correct PHY-level timing information. Although configuration of the memory interface in Qsys is still necessary, it is limited to PHY- and board-level settings.

Configuration

To configure the external memory interface components of the HPS, you open the HPS interface by selecting the Hard Processor System component in Qsys. Within the HPS interface, select the EMIF tab to open the EMIF parameter editor.

The EMIF parameter editor contains four additional tabs: PHY Settings, Memory Parameters, Memory Timing, and Board Settings. The parameters available on these tabs are similar to those available in the parameter editors for non-SoC device families.

There are significant differences between the EMIF parameter editor for the Hard Processor System and the parameter editors for non-SoC devices, as follows:

- Because the HPS memory controller is not configurable through the Quartus II software, the Controller and Diagnostic tabs, which exist for non-SoC devices, are not present in the EMIF parameter editor for the hard processor system.
- Unlike the protocol-specific parameter editors for non-SoC devices, the EMIF parameter editor for the Hard Processor System supports multiple protocols, therefore there is an SDRAM Protocol parameter, where you can specify your external memory interface protocol.

  By default, the EMIF parameter editor assumes the DDR3 protocol, and other parameters are automatically populated with DDR3-appropriate values. If you select a protocol other than DDR3, you will have to change other associated parameter values appropriately.
- Unlike the memory interface clocks in the FPGA, the memory interface clocks for the HPS are initialized by the boot-up code using values provided by the configuration process. You may accept the values provided by UniPHY, or you may use your own PLL settings. If you choose to specify your own PLL settings, you must indicate that the clock frequency that UniPHY should use is the requested clock frequency, and not the achieved clock frequency calculated by UniPHY.

The HPS does not support EMIF synthesis generation, compilation, or timing analysis.

The HPS hard memory controller cannot be bonded with another hard memory controller on the FPGA portion of the device.

Simulation

The HPS component provides a complete simulation model of the HPS memory interface controller and PHY, providing cycle-level accuracy, comparable to the simulation models for the FPGA memory interface.

The simulation model supports only the skip-cal simulation mode; quick-cal and full-cal are not supported. An example design is not provided, however you can create a test design by adding the traffic generator component to your design using Qsys. Also, the HPS simulation model does not use external memory pins to connect to the DDR memory model; instead, the memory model is incorporated directly into the HPS SDRAM interface simulation modules.

Document Revision History

Table 4–12 lists the revision history for this document.

Table 4–12. Document Revision History

Date	Version	Changes
November 2012	1.0	■ Initial release. ■ Moved Using EMI-Related HPS Features in SoC Devices from Hard Memory Interface chapter to this chapter.

EMI_RM_004-2.1

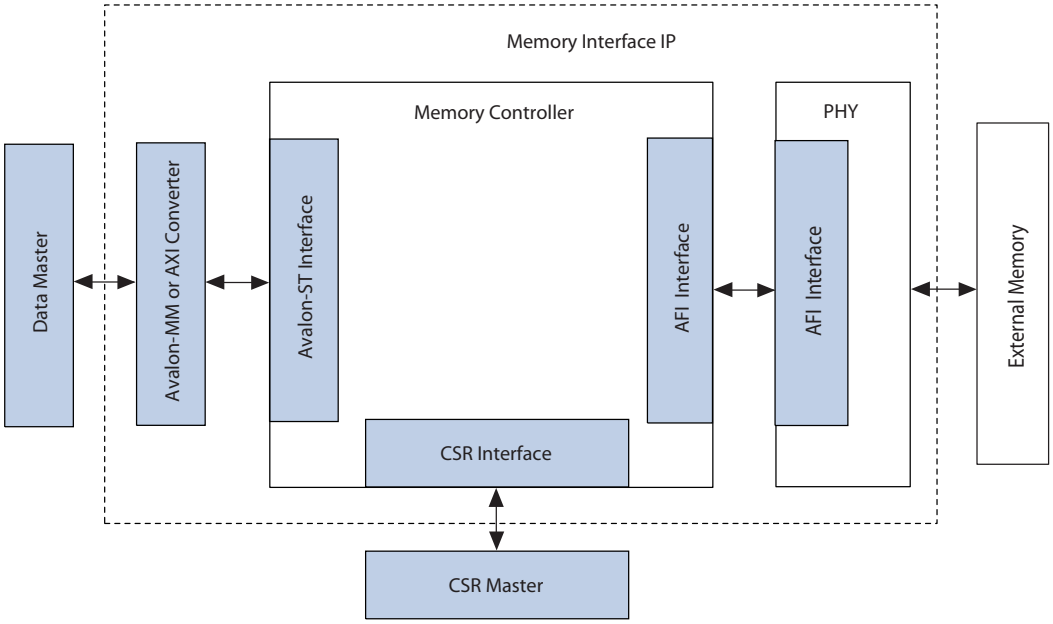
This chapter describes the High Performance Controller II (HPC II) with advanced features for designs generated in the Quartus II software version 11.0 and later. Designs created in earlier versions and regenerated in version 11.0 and later do not inherit the new advanced features; for information on HPC II without the version 11.0 and later advanced features, refer to the External Memory Interface Handbook for Quartus II version 10.1, available in the *External Memory Interfaces* section of the Altera Literature website.

The High Performance Controller II works with the UniPHY-based DDR2, DDR3, and LPDDR2 interfaces, and with the ALTMEMPHY-based DDR, DDR2, and DDR3 interfaces. The controller provides high memory bandwidth, high clock rate performance, and run-time programmability. The controller can reorder data to reduce row conflicts and bus turn-around time by grouping reads and writes together, allowing for efficient traffic patterns and reduced latency.

Memory Controller Architecture

Figure 5–1 shows a high-level block diagram of the overall HPC II memory interface architecture.

Figure 5–1. High-Level Diagram of Memory Interface Architecture



© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

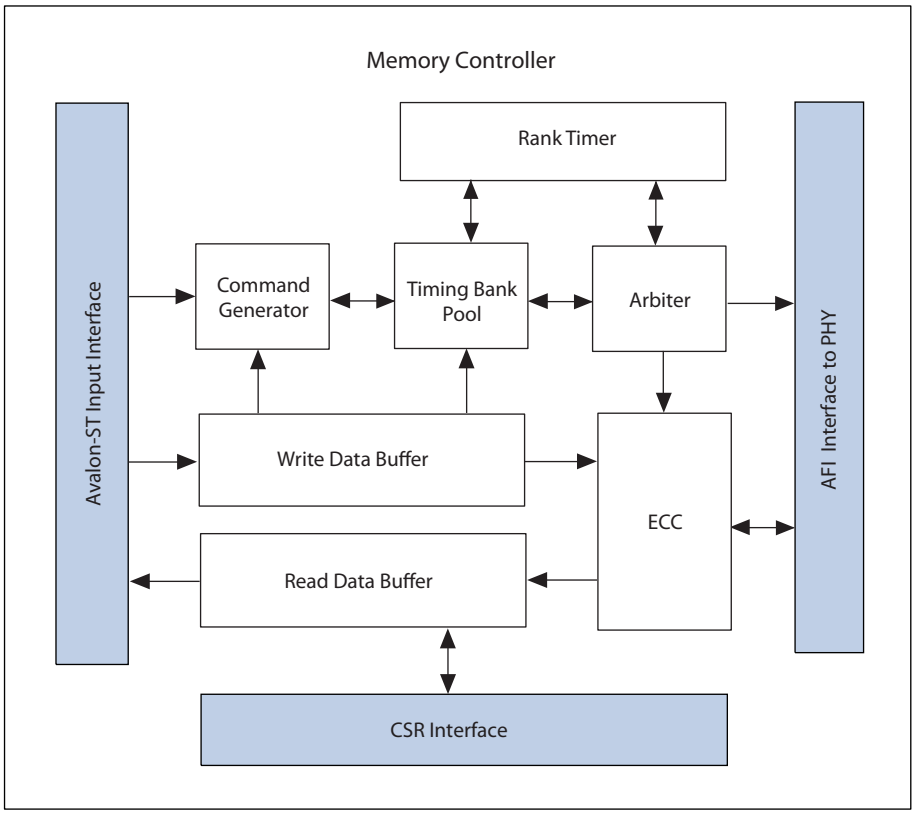
ISO 9001:2008 Registered

The memory interface consists of the memory controller logic block, the physical logic layer (PHY), and their associated interfaces.

The memory controller logic block uses an Avalon Streaming (Avalon-ST) interface as its native interface, and communicates with the PHY layer by the Altera PHY Interface (AFI).

Figure 5–2 shows a block diagram of the memory controller architecture.

Figure 5–2. Memory Controller Architecture Block Diagram



The following sections describe the blocks in Figure 5–2.

Avalon-ST Input Interface

The Avalon-ST interface serves as the entry point to the memory controller, and provides communication with the requesting data masters.

For information about the Avalon interface, refer to *Avalon Interface Specifications*.

AXI to Avalon-ST Converter

The HPC II memory controller includes an AXI to Avalon-ST converter for communication with the AXI protocol. The AXI to Avalon-ST converter provides write address, write data, write response, read address, and read data channels on the AXI interface side, and command, write data, and read data channels on the Avalon-ST interface side.

Handshaking

The AXI protocol employs a handshaking process similar to the Avalon-ST protocol, based on `ready` and `valid` signals.

Command Channel Implementation

The AXI interface includes separate read and write channels, while the Avalon-ST interface has only one command channel. Arbitration of the read and write channels is based on these policies:

- Round robin
- Write priority—write channel has priority over read channel
- Read priority—read channel has priority over write channel

You can choose an arbitration policy by setting the `COMMAND_ARB_TYPE` parameter to one of `ROUND_ROBIN`, `WRITE_PRIORITY`, or `READ_PRIORITY`.

Data Ordering

The AXI specification requires that write data IDs must arrive in the same order as write address IDs are received. Similarly, read data must be returned in the same order as its associated read address is received.

Consequently, the AXI to Avalon-ST converter does not support interleaving of write data; all data must arrive in the same order as its associated write address IDs. On the read side, the controller returns read data based on the read addresses received.

Burst Types

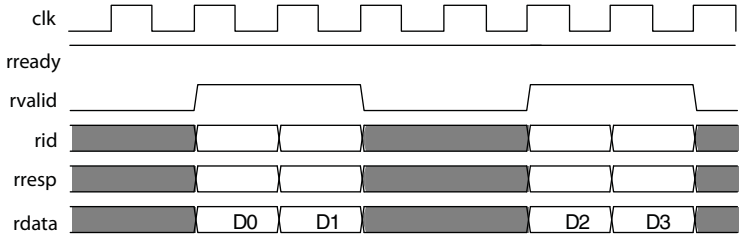
The AXI to Avalon-ST converter supports the following burst types:

- Incrementing burst—the address for each transfer is an increment of the previous transfer address; the increment value depends on the size of the transfer.
- Wrapping burst—similar to the incrementing burst, but wraps to the lower address when the burst boundary is reached. The starting address must be aligned to the size of the transfer. Burst length must be 2, 4, 8, or 16. The burst wrap boundary = burst size * burst length.

Backpressure Support

The write response and read data channels do not support data transfer with backpressure; consequently, you must assert the ready signal for the write response and read data channels to 1 as shown in Figure 5–3, to ensure acceptance of data at any time.

Figure 5–3. Data Transfer Without Backpressure



For information about data transfer with and without backpressure, refer to the *Avalon Interface Specifications*.

Command Generator

The command generator accepts commands from the front-end Avalon-ST interface and from local ECC internal logic, and provides those commands to the timing bank pool.

Timing Bank Pool

The timing bank pool is a parallel queue that works with the arbiter to enable data reordering. The timing bank pool tracks incoming requests, ensures that all timing requirements are met and, upon receiving write-data-ready notification from the write data buffer, passes the requests to the arbiter in an ordered and efficient manner.

Arbiter

The arbiter determines the order in which requests are passed to the memory device. When the arbiter receives a single request, that request is passed immediately; however, when multiple requests are received, the arbiter uses arbitration rules to determine the order in which to pass requests to the memory device.

Arbitration Rules

The arbiter uses the following arbitration rules:

- If only one master is issuing a request, grant that request immediately.
- If there are outstanding requests from two or more masters, the arbiter applies the following tests, in order:
  a. Is there a read request? If so, the arbiter grants the read request ahead of any write requests.
  b. If neither of the above conditions apply, the arbiter grants the oldest request first.

Rank Timer

The rank timer maintains rank-specific timing information, and performs the following functions:

- Ensures that only four activates occur within a specified timing window.
- Manages the read-to-write and write-to-read bus turnaround time.
- Manages the time-to-activate delay between different banks.

Read Data Buffer

The read data buffer receives data from the PHY and passes that data through the input interface to the master.

Write Data Buffer

The write data buffer receives write data from the input interface and passes that data to the PHY, upon approval of the write request.

ECC Block

The error-correcting code (ECC) block comprises an encoder and a decoder-corrector, which can detect and correct single-bit errors, and detect double-bit errors. The ECC block can remedy errors resulting from noise or other impairments during data transmission.

AFI Interface

The AFI interface provides communication between the controller and the physical layer logic (PHY).

For more information about AFI signals, refer to AFI 3.0 Specification.

Unaligned reads and writes on the AFI interface are not supported.

CSR Interface

The CSR interface provides communication with your system's internal control status registers.

Controller Features Descriptions

The following sections describe main features of the memory controller.

Data Reordering

The controller implements data reordering to maximize efficiency for read and write commands. The controller can reorder read and write commands as necessary to mitigate bus turn-around time and reduce conflict between rows.

Inter-bank data reordering reorders commands going to different bank addresses. Commands going to the same bank address are not reordered. This reordering method implements simple hazard detection on the bank address level.

The controller implements logic to limit the length of time that a command can go unserved. This logic is known as starvation control. In starvation control, a counter is incremented for every command served. You can set a starvation limit, to ensure that a waiting command is served immediately, when the starvation counter reaches the specified limit.

Pre-emptive Bank Management

Data reordering allows the controller to issue bank-management commands pre-emptively, based on the patterns of incoming commands; consequently, the desired page in memory can be already open when a command reaches the AFI interface.

Quasi-1T and Quasi-2T

One controller clock cycle equals two memory clock cycles in a half-rate interface, and to four memory clock cycles in a quarter-rate interface. To fully utilize the command bandwidth, the controller can operate in Quasi-1T half-rate and Quasi-2T quarter-rate modes.

In Quasi-1T and Quasi-2T modes, the controller issues two commands on every controller clock cycle. The controller is constrained to issue a row command on the first clock phase and a column command on the second clock phase, or vice versa. Row commands include activate and precharge commands; column commands include read and write commands.

The controller operates in Quasi-1T in half-rate mode, and in Quasi-2T in quarter-rate mode; this operation is transparent and has no user settings.

User Autoprecharge Commands

The autoprecharge read and autoprecharge write commands allow you to indicate to the memory device that this read or write command is the last access to the currently open row. The memory device automatically closes or autoprecharges the page it is currently accessing so that the next access to the same bank is quicker.

This command is useful for applications that require fast random accesses.

Since the HPC II controller can reorder transactions for best efficiency, when you assert the `local_autopch_req` signal, the controller evaluates the current command and buffered commands to determine the best autoprecharge operation.

Half-Rate Bridge

Half-rate bridge support is available for ALTMEMPHY-based cores targeting device families other than Arria II GX. Half-rate bridge support is not available for UniPHY-based cores.

When using the half-rate bridge feature, you must ensure that the `local_size` data for each write command remains constant until the next write command is issued. In other words, the `local_size` bus should not be allowed to change unless the `burst_begin` signal is high.

Before using the half-rate bridge feature, you should perform the following steps:

1. Open the Synopsis Design Constraints file (**.sdc**) for the half-rate bridge and set `slow_clk` to the path of the clock connected to the half-rate bridge's slave interface.
2. Before running the constraint, ensure that the clock `$slow_clk` is already created or declared using the `derive_pll_clocks`, `create_clock`, or `create_generated_clock` function; otherwise, the constraint may be ignored.

Address and Command Decoding Logic

When the main state machine issues a command to the memory, it asserts a set of internal signals. The address and command decoding logic turns these signals into AFI-specific commands and address. This block generates the following signals:

- Clock enable and reset signals: `afi_cke`, `afi_rst_n`
- Command and address signals: `afi_cs_n`, `afi_ba`, `afi_addr`, `afi_ras_n`, `afi_cas_n`, `afi_we_n`

Low-Power Logic

There are two types of low-power logic: the user-controlled self-refresh logic and automatic power-down with programmable time-out logic.

User-Controlled Self-Refresh

When you assert the `local_self_rfsh_req signal`, the controller completes any currently executing reads and writes, and then interrupts the command queue and immediately places the memory into self-refresh mode. When the controller places the memory into self-refresh mode, it responds by asserting an acknowledge signal, `local_self_rfsh_ack`. You can leave the memory in self-refresh mode for as long as you choose.

To bring the memory out of self-refresh mode, you must deassert the request signal, and the controller responds by deasserting the acknowledge signal when the memory is no longer in self-refresh mode.

If a user-controlled refresh request and a system-generated refresh request occur at the same time, the user-controlled refresh takes priority; the system-generated refresh is processed only after the user-controlled refresh request is completed.

Automatic Power-Down with Programmable Time-Out

The controller automatically places the memory in power-down mode to save power if the requested number of idle controller clock cycles is observed in the controller. The **Auto Power Down Cycles** parameter on the **Controller Settings** tab allows you to specify a range between 1 to 65,535 idle controller clock cycles. The counter for the programmable time-out starts when there are no user read or write requests in the command queue. Once the controller places the memory in power-down mode, it responds by asserting the acknowledge signal, `local_power_down_ack`.

ODT Generation Logic

The on-die termination (ODT) generation logic generates the necessary ODT signals for the controller, based on the scheme that Altera recommends.

DDR2 SDRAM

Table 5–1 lists which ODT signal is enabled for single-slot single chip-select per DIMM.

There is no ODT for reads.

Table 5–1. ODT—DDR2 SDRAM Single Slot Single Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[0]	mem_odt[0]

Table 5–2 lists which ODT signal is enabled for single-slot dual chip-select per DIMM.

There is no ODT for reads.

Table 5–2. ODT—DDR2 SDRAM Single Slot Dual Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[0]	mem_odt[0]
mem_cs[1]	mem_odt[1]

Table 5–3 lists which ODT signal is enabled for dual-slot single chip-select per DIMM.

Table 5–3. ODT—DDR2 SDRAM Dual Slot Single Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[0]	mem_odt[1]
mem_cs[1]	mem_odt[0]

Table 5–4 lists which ODT signal is enabled for dual-slot dual chip-select per DIMM.

Table 5–4. ODT—DDR2 SDRAM Dual Slot Dual Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[0]	mem_odt[2]
mem_cs[1]	mem_odt[3]

Table 5–4. ODT—DDR2 SDRAM Dual Slot Dual Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[2]	mem_odt[0]
mem_cs[3]	mem_odt[1]

DDR3 SDRAM

Table 5–5 lists which ODT signal is enabled for single-slot single chip-select per DIMM.

There is no ODT for reads.

Table 5–5. ODT—DDR3 SDRAM Single Slot Single Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[0]	mem_odt[0]

Table 5–6 lists which ODT signal is enabled for single-slot dual chip-select per DIMM.

There is no ODT for reads.

Table 5–6. ODT—DDR3 SDRAM Single Slot Dual Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[0]	mem_odt[0]
mem_cs[1]	mem_odt[1]

Table 5–7 lists which ODT signal is enabled for dual-slot single chip-select per DIMM.

Table 5–7. ODT—DDR3 SDRAM Dual Slot Single Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[0]	mem_odt[0] and mem_odt[1]
mem_cs[1]	mem_odt[0] and mem_odt[1]

Table 5–8 lists which ODT signal is enabled for dual-slot single chip-select per DIMM.

Table 5–8. ODT—DDR3 SDRAM Dual Slot Single Chip-select Per DIMM (Read)

Read On	ODT Enabled
mem_cs[0]	mem_odt[1]
mem_cs[1]	mem_odt[0]

Table 5–9 lists which ODT signal is enabled for dual-slot dual chip-select per DIMM.

Table 5–9. ODT—DDR3 SDRAM Dual Slot Dual Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[0]	mem_odt[0] and mem_odt[2]
mem_cs[1]	mem_odt[1] and mem_odt[3]

Table 5–9. ODT—DDR3 SDRAM Dual Slot Dual Chip-select Per DIMM (Write)

Write On	ODT Enabled
mem_cs[2]	mem_odt[0] and mem_odt[2]
mem_cs[3]	mem_odt[1] and mem_odt[3]

Table 5–10 lists which ODT signal is enabled for dual-slot dual chip-select per DIMM.

Table 5–10. ODT—DDR3 SDRAM Dual Slot Dual Rank Per DIMM (Read)

Read On	ODT Enabled
mem_cs[0]	mem_odt[2]
mem_cs[1]	mem_odt[3]
mem_cs[2]	mem_odt[0]
mem_cs[3]	mem_odt[1]

Burst Merging

The burst merging feature is available to improve throughput for sequential addresses, when the Timing Bank Pool receives requests faster than they can be processed.

For designs created in a version of the high-performance controller II earlier than 11.0, burst merging is turned off by default. To turn on burst merging, perform these steps:

1. In a text editor, open the <*variation_name*>.v file for your design.
2. Search for the ENABLE_BURST_MERGE parameter in the .v file.
3. Change the ENABLE_BURST_MERGE value from 0 to 1.

ECC

The ECC logic comprises an encoder and a decoder-corrector, which can detect and correct single-bit errors, and detect double-bit errors. The ECC logic is available in widths of 16, 24, 40, and 72 bits.

For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, ECC logic is limited to widths of 24 and 40.

- The ECC logic has the following features:
- Has Hamming code ECC logic that encodes every 64, 32, 16, or 8 bits of data into 72, 40, 24, or 16 bits of codeword.
- Has a latency increase of one clock for both writes and reads.
- For a 128-bit interface, ECC is generated as one 64-bit data path with 8-bits of ECC path, plus a second 64-bit data path with 8-bits of ECC path.
- Detects and corrects all single-bit errors.
- Detects all double-bit errors.
- Counts the number of single-bit and double-bit errors.

- Accepts partial writes, which trigger a read-modify-write cycle, for memory devices with DM pins.
- Can inject single-bit and double-bit errors to trigger ECC correction for testing and debugging purposes.
- Generates an interrupt signal when an error occurs.

When using ECC, you must initialize memory before writing to it.

When a single-bit or double-bit error occurs, the ECC logic triggers the `ecc_interrupt` signal to inform you that an ECC error has occurred. When a single-bit error occurs, the ECC logic reads the error address, and writes back the corrected data. When a double-bit error occurs, the ECC logic does not do any error correction but it asserts the `avl_rdata_error` signal to indicate that the data is incorrect. The `avl_rdata_error` signal follows the same timing as the `avl_rdata_valid` signal.

Enabling autocorrection allows the ECC logic to delay all controller pending activities until the correction completes. You can disable autocorrection and schedule the correction manually when the controller is idle to ensure better system efficiency. To manually correct ECC errors, follow these steps:

1. When an interrupt occurs, read out the `SBE_ERROR` register. When a single-bit error occurs, the `SBE_ERROR` register is equal to one.
2. Read out the `ERR_ADDR` register.
3. Correct the single-bit error by issuing a dummy write to the memory address stored in the `ERR_ADDR` register. A dummy write is a write request with the `local_be` signal zero, that triggers a partial write which is effectively a read-modify-write event. The partial write corrects the data at that address and writes it back.

Partial Writes

The ECC logic supports partial writes. Along with the address, data, and burst signals, the Avalon-MM interface also supports a signal vector, `local_be`, that is responsible for byte-enable. Every bit of this signal vector represents a byte on the data-bus. Thus, a logic low on any of these bits instructs the controller not to write to that particular byte, resulting in a partial write. The ECC code is calculated on all bytes of the data-bus. If any bytes are changed, the IP core must recalculate the ECC code and write the new code back to the memory.

For partial writes, the ECC logic performs the following steps:

1. The ECC logic sends a read command to the partial write address.
2. Upon receiving a return data from the memory for the particular address, the ECC logic decodes the data, checks for errors, and then merges the corrected or correct dataword with the incoming information.
3. The ECC logic issues a write to write back the updated data and the new ECC code.

The following corner cases can occur:

- A single-bit error during the read phase of the read-modify-write process. In this case, the IP core corrects the single-bit error first, increments the single-bit error counter and then performs a partial write to this corrected decoded data word.
- A double-bit error during the read phase of the read-modify-write process. In this case, the IP core increments the double-bit error counter and issues an interrupt. The IP core writes a new write word to the location of the error. The ECC status register keeps track of the error information.

Figure 5–4 and Figure 5–5 show partial write operations for the controller, for full and half rate configurations, respectively.

Figure 5–4. Partial Write for the Controller—Full Rate

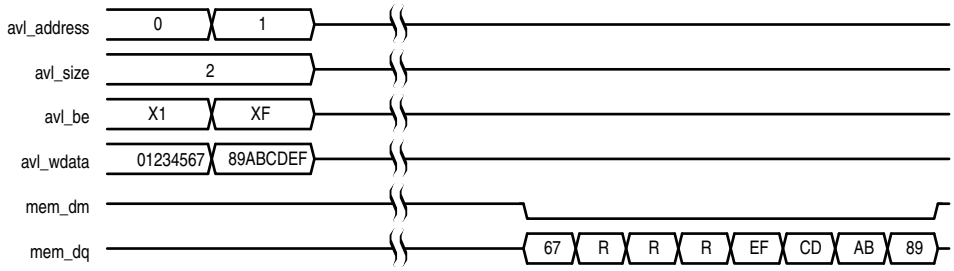
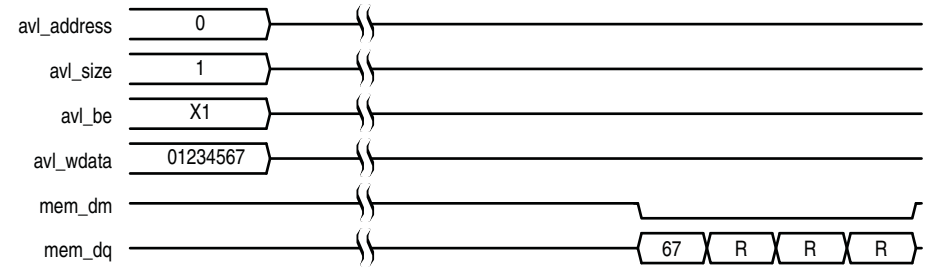


Figure 5–5. Partial Write for the Controller—Half Rate

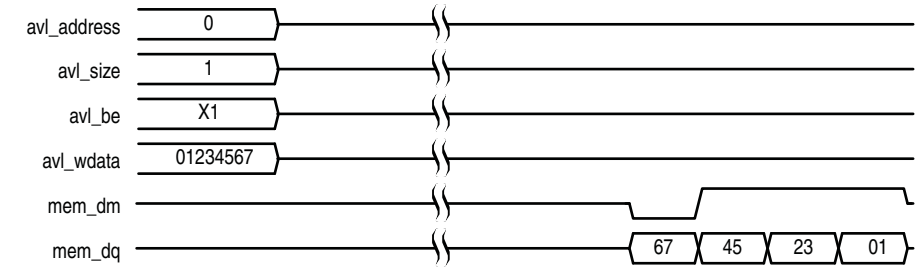


Partial Bursts

DIMMs that do not have the DM pins do not support partial bursts. You must write a minimum (or multiples) of memory-burst-length-equivalent words to the memory at the same time.

Figure 5–6 shows a partial burst operation for the controller.

Figure 5–6. Partial Burst for Controller



External Interfaces

This section discusses the interfaces between the controller and other external memory interface components.

Clock and Reset Interface

The clock and reset interface is part of the AFI interface.

The controller can have up to two clock domains, which are synchronous to each other. The controller operates with a single clock domain when there is no integrated half-rate bridge, and with two-clock domains when there is an integrated half-rate bridge. The clocks are provided by UniPHY.

The main controller clock is `afi_clk`, and the optional half-rate controller clock is `afi_half_clk`. The main and half-rate clocks must be synchronous and have a 2:1 frequency ratio. The optional quarter-rate controller clock is `afi_quarter_clk`, which must also be synchronous and have a 4:1 frequency ratio.

Avalon-ST Data Slave Interface

The Avalon-ST data slave interface consists of the following Avalon-ST channels, which together form a single data slave:

- The command channel, which serves as command and address for both read and write operations.
- The write data channel, which carries write data.
- The read data channel, which carries read data.

For information about the Avalon interface, refer to *Avalon Interface Specifications*.

AXI Data Slave Interface

The AXI data interface consists of the following channels, which communicate with the Avalon-ST interface through the AXI to Avalon-ST converter:

- The write address channel, which carries address information for write operations.
- The write data channel, which carries write data.
- The write response channel, which carries write response data.
- The read address channel, which carries address information for read operations.
- The read data channel, which carries read data.

Enabling the AXI Interface

This section provides guidance for enabling the AXI interface.

1. To enable the AXI interface, first open in an editor the file appropriate for the required flow, as indicated below:
   - For synthesis flow: <*working_dir*>/<*variation_name*>/<*variation_name*>_c0.v
   - For simulation flow: <*working_dir*>/<*variation_name*>_sim/<*variation_name*>/<*variation_name*>_c0.v
   - Example design fileset for synthesis: <*working_dir*>/<*variation_name*>_example_design/example_project/<*variation_name*>_example/submodules/<*variation_name*>_example_if0_c0.v
   - Example design fileset for simulation: <*working_dir*>/<*variation_name*>_example_design/simulation/verilog/submodules/<*variation_name*>_example_sim_e0_if0_c0.v
2. Locate and remove the **alt_mem_ddrx_mm_st_converter** instantiation from the .v file opened in the preceding step.
3. Instantiate the **alt_mem_ddrx_axi_st_converter** module into the open .v file. Refer to the following code fragment as a guide:

```
module ? # ( parameter
// AXI parameters
AXI_ID_WIDTH      = <replace parameter value>,
AXI_ADDR_WIDTH    = <replace parameter value>,
AXI_LEN_WIDTH     = <replace parameter value>,
AXI_SIZE_WIDTH    = <replace parameter value>,
AXI_BURST_WIDTH   = <replace parameter value>,
AXI_LOCK_WIDTH    = <replace parameter value>,
AXI_CACHE_WIDTH   = <replace parameter value>,
AXI_PROT_WIDTH    = <replace parameter value>,
AXI_DATA_WIDTH    = <replace parameter value>,
AXI_RESP_WIDTH    = <replace parameter value>
)
(
// Existing ports
...
// AXI Interface ports
// Write address channel
input  wire [AXI_ID_WIDTH - 1 : 0] awid,
input  wire [AXI_ADDR_WIDTH - 1 : 0] awaddr,
input  wire [AXI_LEN_WIDTH - 1 : 0] awlen,
input  wire [AXI_SIZE_WIDTH - 1 : 0] awsize,
input  wire [AXI_BURST_WIDTH - 1 : 0] awburst,
input  wire [AXI_LOCK_WIDTH - 1 : 0] awlock,
input  wire [AXI_CACHE_WIDTH - 1 : 0] awcache,
input  wire [AXI_PROT_WIDTH - 1 : 0] awprot,
input  wire                          awvalid,
output wire                          awready,

// Write data channel
input  wire [AXI_ID_WIDTH - 1 : 0] wid,
input  wire [AXI_DATA_WIDTH - 1 : 0] wdata,
input  wire [AXI_DATA_WIDTH / 8 - 1 : 0] wstrb,
```

```
input  wire                              wlast,
input  wire                              wvalid,
output wire                              wready,

// Write response channel
output wire [AXI_ID_WIDTH - 1 : 0] bid,
output wire [AXI_RESP_WIDTH - 1 : 0] bresp,
output wire                              bvalid,
input  wire                              bready,

// Read address channel
input  wire [AXI_ID_WIDTH - 1 : 0] arid,
input  wire [AXI_ADDR_WIDTH - 1 : 0] araddr,
input  wire [AXI_LEN_WIDTH - 1 : 0] arlen,
input  wire [AXI_SIZE_WIDTH - 1 : 0] arsize,
input  wire [AXI_BURST_WIDTH - 1 : 0] arburst,
input  wire [AXI_LOCK_WIDTH - 1 : 0] arlock,
input  wire [AXI_CACHE_WIDTH - 1 : 0] arcache,
input  wire [AXI_PROT_WIDTH - 1 : 0] arprot,
input  wire                              arvalid,
output wire                              arready,

// Read data channel
output wire [AXI_ID_WIDTH - 1 : 0] rid,
output wire [AXI_DATA_WIDTH - 1 : 0] rdata,
output wire [AXI_RESP_WIDTH - 1 : 0] rresp,
output wire                              rlast,
output wire                              rvalid,
input  wire                              rready
);

// Existing wire, register declaration and instantiation
...
// AXI interface instantiation
alt_mem_ddrx_axi_st_converter #
(
.AXI_ID_WIDTH              (AXI_ID_WIDTH    ),
.AXI_ADDR_WIDTH            (AXI_ADDR_WIDTH  ),
.AXI_LEN_WIDTH             (AXI_LEN_WIDTH   ),
.AXI_SIZE_WIDTH            (AXI_SIZE_WIDTH  ),
.AXI_BURST_WIDTH           (AXI_BURST_WIDTH ),
.AXI_LOCK_WIDTH            (AXI_LOCK_WIDTH  ),
.AXI_CACHE_WIDTH           (AXI_CACHE_WIDTH ),
.AXI_PROT_WIDTH            (AXI_PROT_WIDTH  ),
.AXI_DATA_WIDTH            (AXI_DATA_WIDTH  ),
.AXI_RESP_WIDTH            (AXI_RESP_WIDTH  ),
.ST_ADDR_WIDTH             (ST_ADDR_WIDTH   ),
.ST_SIZE_WIDTH             (ST_SIZE_WIDTH   ),
.ST_ID_WIDTH               (ST_ID_WIDTH     ),
.ST_DATA_WIDTH             (ST_DATA_WIDTH   ),
.COMMAND_ARB_TYPE          (COMMAND_ARB_TYPE)
)
```

```
a0
(
.ctl_clk                  (afi_clk),
.ctl_reset_n              (afi_reset_n),
.awid                     (awid),
.awaddr                   (awaddr),
.awlen                    (awlen),
.awsize                   (awsize),
.awburst                  (awburst),
.awlock                   (awlock),
.awcache                  (awcache),
.awprot                   (awprot),
.awvalid                  (awvalid),
.awready                  (awready),
.wid                      (wid),
.wdata                    (wdata),
.wstrb                    (wstrb),
.wlast                    (wlast),
.wvalid                   (wvalid),
.wready                   (wready),
.bid                      (bid),
.bresp                    (bresp),
.bvalid                   (bvalid),
.bready                   (bready),
.arid                     (arid),
.araddr                   (araddr),
.arlen                    (arlen),
.arsize                   (arsize),
.arburst                  (arburst),
.arlock                   (arlock),
.arcache                  (arcache),
.arprot                   (arprot),
.arvalid                  (arvalid),
.arready                  (arready),
.rid                      (rid),
.rdata                    (rdata),
.rresp                    (rresp),
.rlast                    (rlast),
.rvalid                   (rvalid),
.rready                   (rready),
.itf_cmd_ready            (ng0_native_st_itf_cmd_ready),
.itf_cmd_valid            (a0_native_st_itf_cmd_valid),
.itf_cmd                  (a0_native_st_itf_cmd),
.itf_cmd_address          (a0_native_st_itf_cmd_address),
.itf_cmd_burstlen         (a0_native_st_itf_cmd_burstlen),
.itf_cmd_id               (a0_native_st_itf_cmd_id),
.itf_cmd_priority         (a0_native_st_itf_cmd_priority),
.itf_cmd_autoprecharge   (a0_native_st_itf_cmd_autopercharge),
.itf_cmd_multicast       (a0_native_st_itf_cmd_multicast),
.itf_wr_data_ready        (ng0_native_st_itf_wr_data_ready),
.itf_wr_data_valid        (a0_native_st_itf_wr_data_valid),
.itf_wr_data              (a0_native_st_itf_wr_data),
```

```
.itf_wr_data_byte_en      (a0_native_st_itf_wr_data_byte_en),
.itf_wr_data_begin         (a0_native_st_itf_wr_data_begin),
.itf_wr_data_last          (a0_native_st_itf_wr_data_last),
.itf_wr_data_id            (a0_native_st_itf_wr_data_id),
.itf_rd_data_ready         (a0_native_st_itf_rd_data_ready),
.itf_rd_data_valid         (ng0_native_st_itf_rd_data_valid),
.itf_rd_data               (ng0_native_st_itf_rd_data),
.itf_rd_data_error         (ng0_native_st_itf_rd_data_error),
.itf_rd_data_begin         (ng0_native_st_itf_rd_data_begin),
.itf_rd_data_last          (ng0_native_st_itf_rd_data_last),
.itf_rd_data_id            (ng0_native_st_itf_rd_data_id)
);
```

4. Set the required parameters for the AXI interface. Table 5–11 lists the available parameters.
5. Export the AXI interface to the top-level wrapper, making it accessible to the AXI master.
6. To add the AXI interface to the Quartus II project:
   a. On the Assignments > Settings menu in the Quartus II software, open the File tab.
   b. Add the **alt_mem_ddrx_axi_st_converter.v** file to the project.

Table 5–11. AXI Interface Parameters (Part 1 of 2)

Parameter Name	Description / Value
AXI_ID_WIDTH	Width of the AXI ID bus. Default value is 4.
AXI_ADDR_WIDTH	Width of the AXI address bus. Must be set according to the Avalon interface address and data bus width as shown below: AXI_ADDR_WIDTH = LOCAL_ADDR_WIDTH + log2(LOCAL_DATA_WIDTH/8) LOCAL_ADDR_WIDTH is the memory controller Avalon interface address width. LOCAL_DATA_WIDTH is the memory controller Avalon data interface width.
AXI_LEN_WIDTH	Width of the AXI length bus. Default value is 8. Should be set to LOCAL_SIZE_WIDTH - 1, where LOCAL_SIZE_WIDTH is the memory controller Avalon interface burst size width
AXI_SIZE_WIDTH	Width of the AXI size bus. Default value is 3.
AXI_BURST_WIDTH	Width of the AXI burst bus. Default value is 2.
AXI_LOCK_WIDTH	Width of the AXI lock bus. Default value is 2.
AXI_CACHE_WIDTH	Width of the AXI cache bus. Default value is 4.
AXI_PROT_WIDTH	Width of the AXI protection bus. Default value is 3.

Table 5–11. AXI Interface Parameters (Part 2 of 2)

Parameter Name	Description / Value
AXI_DATA_WIDTH	Width of the AXI data bus. Should be set to match the Avalon interface data bus width. AXI_DATA_WIDTH = LOCAL_DATA_WIDTH, where LOCAL_DATA_WIDTH is the memory controller Avalon interface input data width.
AXI_RESP_WIDTH	Width of the AXI response bus. Default value is 2.
ST_ADDR_WIDTH	Width of the Avalon interface address. Must be set to match the Avalon interface address bus width. ST_ADDR_WIDTH = LOCAL_ADDR_WIDTH, where LOCAL_ADDR_WIDTH is the memory controller Avalon interface address width.
ST_SIZE_WIDTH	Width of the Avalon interface burst size. ST_SIZE_WIDTH = AXI_LEN_WIDTH + 1
ST_ID_WIDTH	Width of the Avalon interface ID. Default value is 4. ST_ID_WIDTH = AXI_ID_WIDTH
ST_DATA_WIDTH	Width of the Avalon interface data. ST_DATA_WIDTH = AXI_DATA_WIDTH.
COMMAND_ARB_TYPE	Specifies the AXI command arbitration type, as shown: ROUND_ROBIN: arbitrates between read and write address channel in round robin fashion. Default option. WRITE_PRIORITY: write address channel has priority if both channels send request simultaneously. READ_PRIORITY: read address channel has priority if both channels send request simultaneously.
REGISTERED	Setting this parameter to 1 adds an extra register stage in the AXI interface and incurs one extra clock cycle of latency. Default value is 1.

Table 5–12 lists the AXI interface ports.

Table 5–12. AXI Interface Ports (Part 1 of 3)

Name	Direction	Description
awid	Input	AXI write address channel ID bus.
awaddr	Input	AXI write address channel address bus.
awlen	Input	AXI write address channel length bus.
awsize	Input	AXI write address channel size bus.
awburst	Input	AXI write address channel burst bus. (Interface supports only INCR and WRAP burst types.)

Table 5–12. AXI Interface Ports (Part 2 of 3)

Name	Direction	Description
awlock	Input	AXI write address channel lock bus. (Interface does not support this feature.)
awcache	Input	AXI write address channel cache bus. (Interface does not support this feature.)
awprot	Input	AXI write address channel protection bus. (Interface does not support this feature.)
awvalid	Input	AXI write address channel valid signal.
awready	Output	AXI write address channel ready signal.
wid	Input	AXI write address channel ID bus.
wdata	Input	AXI write address channel data bus.
wstrb	Input	AXI write data channel strobe bus.
wlast	Input	AXI write data channel last burst signal.
wvalid	Input	AXI write data channel valid signal.
wready	Output	AXI write data channel ready signal.
bid	Output	AXI write response channel ID bus.
bresp	Output	AXI write response channel response bus. Response encoding information: 'b00 - OKAY 'b01 - Reserved 'b10 - Reserved 'b11 - Reserved
bvalid	Output	AXI write response channel valid signal.
bready	Input	AXI write response channel ready signal. Must be set to 1. Interface does not support back pressure for write response channel.
arid	Input	AXI read address channel ID bus.
araddr	Input	AXI read address channel address bus.
arlen	Input	AXI read address channel length bus.
arsize	Input	AXI read address channel size bus.
arburst	Input	AXI read address channel burst bus. (Interface supports only INCR and WRAP burst types.)
arlock	Input	AXI read address channel lock bus. (Interface does not support this feature.)
arcache	Input	AXI read address channel cache bus. (Interface does not support this feature.)

Table 5–12. AXI Interface Ports (Part 3 of 3)

Name	Direction	Description
arprot	Input	AXI read address channel protection bus. (Interface does not support this feature.)
arvalid	Input	AXI read address channel valid signal.
arready	Output	AXI read address channel ready signal.
rid	Output	AXI read data channel ID bus.
rdata	Output	AXI read data channel data bus.
rresp	Output	AXI read data channel response bus. Response encoding information: 'b00 - OKAY 'b01 - Reserved 'b10 - Data error 'b11 - Reserved
rlast	Output	AXI read data channel last burst signal.
rvalid	Output	AXI read data channel valid signal.
rready	Input	AXI read data channel ready signal. Must be set to 1. Interface does not support back pressure for write response channel.

For information about the AXI specification, refer to the ARM website, at *www.arm.com*.

Controller-PHY Interface

The interface between the controller and the PHY is part of the AFI interface.

The controller assumes that the PHY performs all necessary calibration processes without any interaction with the controller.

For more information about AFI signals, refer to AFI 3.0 Specification.

Memory Side-Band Signals

This section describes supported side-band signals.

Self-Refresh (Low Power) Interface

The optional low power self-refresh interface consists of a request signal and an acknowledgement signal, which you can use to instruct the controller to place the memory device into self-refresh mode. This interface is clocked by `afi_clk`.

When you assert the request signal, the controller places the memory device into self-refresh mode and asserts the acknowledge signal. To bring the memory device out of self-refresh mode, you deassert the request signal; the controller then deasserts the acknowledge signal when the memory device is no longer in self-refresh mode.

For multi-rank designs using the HPC II memory controller, a self-refresh and a user-refresh cannot be made to the same memory chip simultaneously.

Also, the self-refresh `ack` signal indicates that at least one device has entered self-refresh, but does not necessarily mean that all devices have entered self-refresh.

User-Controlled Refresh Interface

The optional user-controlled refresh interface consists of a request signal, a chip select signal, and an acknowledgement signal. This interface provides increased control over worst-case read latency and enables you to issue refresh bursts during idle periods. This interface is clocked by `afi_clk`.

When you assert a refresh request signal to instruct the controller to perform a refresh operation, that request takes priority over any outstanding read or write requests that might be in the command queue. In addition to the request signal, you must also choose the chip to be refreshed by asserting the refresh chip select signal along with the request signal. If you do not assert the chip select signal with the request signal, unexpected behavior may result.

The controller attempts to perform a refresh as long as the refresh request signal is asserted; if you require only one refresh, you should deassert the refresh request signal after the acknowledgement signal is received. If you maintain the request signal high after the acknowledgement is sent, it would indicate that further refresh is required. You should deassert the request signal after the required number of acknowledgement/refresh is received from the controller. You can issue up to a maximum of nine consecutive refresh commands.

For multi-rank designs using the HPC II memory controller, a self-refresh and a user-refresh cannot be made to the same memory chip simultaneously.

Configuration and Status Register (CSR) Interface

The controller has a configuration and status register (CSR) interface that allows you to configure timing parameters, address widths, and the behavior of the controller. The CSR interface is a 32-bit Avalon-MM slave of fixed address width; if you do not need this feature, you can disable it to save area.

This interface is clocked by `csr_clk`, which is the same as `afi_clk`, and is always synchronous relative to the main data slave interface.

Table 5–13 lists the controller's external interfaces.

Table 5–13. Summary of Controller External Interfaces (Part 1 of 2)

Interface Name	Display Name	Type	Description
Clock and Reset Interface			
Clock and Reset Interface	Clock and Reset Interface	AFI *(1)*	Clock and reset generated by UniPHY to the controller.
Avalon-ST Data Slave Interface			
Command Channel	Avalon-ST Data Slave Interface	Avalon-ST *(2)*	Address and command channel for read and write, single command single data (SCSD).

Table 5–13. Summary of Controller External Interfaces (Part 2 of 2)

Interface Name	Display Name	Type	Description
Write Data Channel	Avalon-ST Data Slave Interface	Avalon-ST (2)	Write Data Channel, single command multiple data (SCMD).
Read Data Channel	Avalon-ST Data Slave Interface	Avalon-ST (2)	Read data channel, SCMD with read data error response.
Controller-PHY Interface			
AFI 3.0	AFI Interface	AFI (1)	Interface between controller and PHY.
Memory Side-Band Signals			
Self Refresh (Low Power) Interface	Self Refresh (Low Power) Interface	Avalon Control & Status Interface (2)	SDRAM-specific signals to place memory into low-power mode.
User-Controller Refresh Interface	User-Controller Refresh Interface	Avalon Control & Status Interface (2)	SDRAM-specific signals to request memory refresh.
Configuration and Status Register (CSR) Interface			
CSR	Configuration and Status Register Interface	Avalon-MM (2)	Enables on-the-fly configuration of memory timing parameters, address widths, and controller behaviour.

Notes:

(1) For information about AFI signals, refer to AFI 3.0 Specification.

(2) For information about Avalon signals, refer to *Avalon Interface Specifications*.

Top-Level Signals Description

Table 5–14 lists the clock and reset signals.

The suffix _n denotes active low signals.

Table 5–14. Clock and Reset Signals (Part 1 of 2)

Name	Direction	Description
global_reset_n	Input	The asynchronous reset input to the controller. The IP core derives all other reset signals from resynchronized versions of this signal. This signal holds the PHY, including the PLL, in reset while low.
pll_ref_clk	Input	The reference clock input to PLL.
phy_clk	Output	The system clock that the PHY provides to the user. All user inputs to and outputs from the controller must be synchronous to this clock.
reset_phy_clk_n	Output	The reset signal that the PHY provides to the user. The IP core asserts reset_phy_clk_n asynchronously and deasserts synchronously to phy_clk clock domain.
aux_full_rate_clk	Output	An alternative clock that the PHY provides to the user. This clock always runs at the same frequency as the external memory interface. In half-rate designs, this clock is twice the frequency of the phy_clk and you can use it whenever you require a 2x clock. In full-rate designs, the same PLL output as the phy_clk signal drives this clock.

Table 5–14. Clock and Reset Signals (Part 2 of 2)

Name	Direction	Description
aux_half_rate_clk	Output	An alternative clock that the PHY provides to the user. This clock always runs at half the frequency as the external memory interface. In full-rate designs, this clock is half the frequency of the phy_clk and you can use it, for example to clock the user side of a half-rate bridge. In half-rate designs, or if the **Enable Half Rate Bridge** option is turned on. The same PLL output that drives the phy_clk signal drives this clock.
dll_reference_clk	Output	Reference clock to feed to an externally instantiated DLL.
reset_request_n	Output	Reset request output that indicates when the PLL outputs are not locked. Use this signal as a reset request input to any system-level reset controller you may have. This signal is always low when the PLL is trying to lock, and so any reset logic using Altera advises you detect a reset request on a falling edge rather than by level detection.
soft_reset_n	Input	Edge detect reset input for SOPC Builder or for control by other system reset logic. Assert to cause a complete reset to the PHY, but not to the PLL that the PHY uses.
seriesterminationcontrol	Input (for OCT slave)	Required signal for PHY to provide series termination calibration value. Must be connected to a user-instantiated OCT control block (alt_oct) or another UniPHY instance that is set to OCT master mode.
	Output (for OCT master)	Unconnected PHY signal, available for sharing with another PHY.
parallelterminationcontrol	Input (for OCT slave)	Required signal for PHY to provide series termination calibration value. Must be connected to a user-instantiated OCT control block (alt_oct) or another UniPHY instance that is set to OCT master mode.
	Output (for OCT master)	Unconnected PHY signal, available for sharing with another PHY.
oct_rdn	Input (for OCT master)	Must connect to calibration resistor tied to GND on the appropriate RDN pin on the device. (Refer to appropriate device handbook.)
oct_rup	Input (for OCT master)	Must connect to calibration resistor tied to V_{ccio} on the appropriate RUP pin on the device. (See appropriate device handbook.)
dqs_delay_ctrl_import	Input	Allows the use of DLL in another PHY instance in this PHY instance. Connect the export port on the PHY instance with a DLL to the import port on the other PHY instance.
csr_clk (1)	Output	Clock for the configuration and status register (CSR) interface, which is the same as afi_clk and is always synchronous relative to the main data slave interface.

Note for Table 5–14:

(1) Applies only to the hard memory controller with multiport front end available in Arria V and Cyclone V devices.

Table 5–15 lists the controller local interface signals.

Table 5–15. Local Interface Signals (Part 1 of 4)

Signal Name	Direction	Description
avl_addr[] (1)	Input	Memory address at which the burst should start. By default, the IP core maps local address to the bank interleaving scheme. You can change the ordering via the **Local-to-Memory Address Mapping** option in the **Controller Settings** page. This signal needs to remain stable only during the first transaction of a burst. The constantBurstBehavior property is always false for UniPHY controllers. The IP core sizes the width of this bus according to the following equations: ■ Full rate controllers For one chip select: width = row bits + bank bits + column bits – 1 For multiple chip selects: width = chip bits* + row bits + bank bits + column bits – 1 If the bank address is 2 bits wide, row is 13 bits wide and column is 10 bits wide, the local address is 24 bits wide. To map local_address to bank, row and column address: avl_addr is 24 bits wide avl_addr[23:11] = row address [12:0] avl_addr[10:9] = bank address [1:0] avl_addr [8:0] = column address [9:1] The IP core ignores the least significant bit (LSB) of the column address (multiples of two) on the memory side, because the local data width is twice that of the memory data bus width. ■ Half rate controllers For one chip select: width = row bits + bank bits + column bits – 2 For multiple chip selects: width = chip bits* + row bits + bank bits + column bits – 2 If the bank address is 2 bits wide, row is 13 bits wide and column is 10 bits wide, the local address is 23 bits wide. To map local_address to bank, row and column address: avl_addr is 23 bits wide avl_addr[22:10] = row address [12:0] avl_addr[9:8] = bank address [1:0] avl_addr[7:0] = column address [9:2] The IP core ignores two LSBs of the column address (multiples of four) on the memory side, because the local data width is four times that of the memory data bus width. ■ Quarter rate controllers For one chip select: width = row bits + bank bits + column bits – 3 For multiple chip selects: width = chip bits* + row bits + bank bits + column bits – 3 If the bank address is 2 bits wide, row is 13 bits wide and column is 10 bits wide, the local address is 22 bits wide. (* *chip bits* is a derived value indicating the number of address bits necessary to uniquely address every memory rank in the system; this value is not user configurable.)

Table 5–15. Local Interface Signals (Part 2 of 4)

Signal Name	Direction	Description
avl_be[] (2)	Input	Byte enable signal, which you use to mask off individual bytes during writes. avl_be is active high; mem_dm is active low. To map avl_wdata and avl_be to mem_dq and mem_dm, consider a full-rate design with 32-bit avl_wdata and 16-bit mem_dq. avl_wdata = < 22334455 >< 667788AA >< BBCCDDEE > avl_be = < 1100 >< 0110 >< 1010 > These values map to: Mem_dq = <4455><2233><88AA><6677><DDEE><BBCC> Mem_dm = <1 1 ><0 0 ><0 1 ><1 0 ><0 1 ><0 1 >
avl_burstbegin (3)	Input	The Avalon burst begin strobe, which indicates the beginning of an Avalon burst. Unlike all other Avalon-MM signals, the burst begin signal does not stay asserted if avl_ready is deasserted. For write transactions, assert this signal at the beginning of each burst transfer and keep this signal high for one cycle per burst transfer, even if the slave deasserts avl_ready. The IP core samples this signal at the rising edge of phy_clk when avl_write_req is asserted. After the slave deasserts the avl_ready signal, the master keeps all the write request signals asserted until avl_ready signal becomes high again. For read transactions, assert this signal for one clock cycle when read request is asserted and avl_addr from which the data should be read is given to the memory. After the slave deasserts avl_ready (waitrequest_n in Avalon interface), the master keeps all the read request signals asserted until avl_ready becomes high again.
avl_read_req (4)	Input	Read request signal. You cannot assert read request and write request signals at the same time. The controller must deassert reset_phy_clk_n before you can assert local_autopch_req.
local_refresh_req	Input	User-controlled refresh request. If **Enable User Auto-Refresh Controls** option is turned on, local_refresh_req becomes available and you are responsible for issuing sufficient refresh requests to meet the memory requirements. This option allows complete control over when refreshes are issued to the memory including grouping together multiple refresh commands. Refresh requests take priority over read and write requests, unless the IP core is already processing the requests.
local_refresh_chip	Input	Controls which chip to issue the user refresh to. The IP core uses this active high signal with local_refresh_req. This signal is as wide as the memory chip select. This signal asserts a high value to each bit that represents the refresh for the corresponding memory chip. For example: If local_refresh_chip signal is assigned with a value of 4'b0101, the controller refreshes the memory chips 0 and 2, and memory chips 1 and 3 are not refreshed.
avl_size[] (5)	Input	Controls the number of beats in the requested read or write access to memory, encoded as a binary number. The IP core supports Avalon burst lengths from 1 to 64. The IP core derives the width of this signal based on the burst count that you specify in the **Local Maximum Burst Count** option. With the derived width, you specify a value ranging from 1 to the local maximum burst count specified. This signal needs to remain stable only during the first transaction of a burst. The constantBurstBehavior property is always false for UniPHY controllers.

Table 5–15. Local Interface Signals (Part 3 of 4)

Signal Name	Direction	Description
avl_wdata[] (6)	Input	Write data bus. The width of avl_wdata is twice that of the memory data bus for a full-rate controller, four times the memory data bus for a half-rate controller, and eight times the memory data bus for a quarter-rate controller. If **Generate power-of-2 data bus widths for Qsys and SOPC Builder** is enabled, the width is rounded down to the nearest power of 2.
avl_write_req (7)	Input	Write request signal. You cannot assert read request and write request signal at the same time. The controller must deassert reset_phy_clk_n before you can assert avl_write_req.
local_autopch_req (8)	Input	User control of autoprecharge. If you turn on **Enable Auto-Precharge Control**, the local_autopch_req signal becomes available and you can request the controller to issue an autoprecharge write or autoprecharge read command. These commands cause the memory to issue a precharge command to the current bank at the appropriate time without an explicit precharge command from the controller. This feature is particularly useful if you know the current read or write is the last one you intend to issue to the currently open row. The next time you need to use that bank, the access could be quicker as the controller does not need to precharge the bank before activating the row you wish to access. Upon receipt of the local_autopch_req signal, the controller evaluates the pending commands in the command buffer and determines the most efficient autoprecharge operation to perform, reordering commands if necessary. The controller must deassert reset_phy_clk_n before you can assert local_autopch_req.
local_self_rfsh_chip	Input	Controls which chip to issue the user refresh to. The IP core uses this active high signal with local_self_rfsh_req. This signal is as wide as the memory chip select. This signal asserts a high value to each bit that represents the refresh for the corresponding memory chip. For example: If local_self_rfsh_chip signal is assigned with a value of 4'b0101, the controller refreshes the memory chips 0 and 2, and memory chips 1 and 3 are not refreshed.
local_self_rfsh_req	Input	User control of the self-refresh feature. If you turn on **Enable Self-Refresh Controls**, you can request that the controller place the memory devices into a self-refresh state by asserting this signal. The controller places the memory in the self-refresh state as soon as it can without violating the relevant timing parameters and responds by asserting local_self_rfsh_ack. You can hold the memory in the self-refresh state by keeping this signal asserted. You can release the memory from the self-refresh state at any time by deasserting local_self_rfsh_req and the controller responds by deasserting local__self_rfsh_ack when it has successfully brought the memory out of the self-refresh state.
local_init_done	Output	When the memory initialization, training, and calibration are complete, the PHY sequencer asserts ctrl_usr_mode_rdy to the memory controller, which then asserts this signal to indicate that the memory interface is ready for use.
avl_rdata[] (9)	Output	Read data bus. The width of avl_rdata is twice that of the memory data bus for a full rate controller; four times the memory data bus for a half rate controller. If **Generate power-of-2 data bus widths for Qsys and SOPC Builder** is enabled, the width is rounded down to the nearest power of 2.

Table 5–15. Local Interface Signals (Part 4 of 4)

Signal Name	Direction	Description
avl_rdata_error (10)	Output	Asserted if the current read data has an error. This signal is only available if you turn on **Enable Error Detection and Correction Logic**. The controller asserts this signal with the avl_rdata_valid signal. If the controller encounters double-bit errors, no correction is made and the controller asserts this signal.
avl_rdata_valid (11)	Output	Read data valid signal. The avl_rdata_valid signal indicates that valid data is present on the read data bus.
avl_ready (12)	Output	The avl_ready signal indicates that the controller is ready to accept request signals. If controller asserts the avl_ready signal in the clock cycle that it asserts a read or write request, the controller accepts that request. The controller deasserts the avl_ready signal to indicate that it cannot accept any more requests. The controller can buffer eight read or write requests, after which the avl_ready signal goes low.
local_refresh_ack	Output	Refresh request acknowledge, which the controller asserts for one clock cycle every time it issues a refresh. Even if you do not turn on **Enable User Auto-Refresh Controls**, local_refresh_ack still indicates to the local interface that the controller has just issued a refresh command.
local_self_rfsh_ack	Output	Self refresh request acknowledge signal. The controller asserts and deasserts this signal in response to the local_self_rfsh_req signal.
local_power_down_ack	Output	Auto power-down acknowledge signal. The controller asserts this signal for one clock cycle every time auto power-down is issued.
ecc_interrupt (13)	Output	Interrupt signal from the ECC logic. The controller asserts this signal when the ECC feature is turned on, and the controller detects an error.

Notes for Table 5–15:

(1) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_addr becomes a per port value, avl_addr_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(2) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_be becomes a per port value, avl_be_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(3) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_burstbegin becomes a per port value, avl_burstbegin_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(4) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_read_req becomes a per port value, avl_read_req_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(5) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_size becomes a per port value, avl_size_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(6) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_wdata becomes a per port value, avl_wdata_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(7) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_write_req becomes a per port value, avl_write_req_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(8) This signal is not applicable to the hard memory controller.

(9) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_rdata becomes a per port value, avl_rdata_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(10) This signal is not applicable to the hard memory controller.

(11) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_rdata_valid becomes a per port value, avl_rdata_valid_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(12) For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl_ready becomes a per port value, avl_ready_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.

(13) This signal is not applicable to the hard memory controller.

Table 5–16 lists the controller interface signals.

Table 5–16. Interface Signals

Signal Name	Direction	Description
mem_dq[]	Bidirectional	Memory data bus. This bus is half the width of the local read and write data busses.
mem_dqs[]	Bidirectional	Memory data strobe signal, which writes data into the memory device and captures read data into the Altera device.
mem_dqs_n[]	Bidirectional	Inverted memory data strobe signal, which with the mem_dqs signal improves signal integrity.
mem_ck	Output	Clock for the memory device.
mem_ck_n	Output	Inverted clock for the memory device.
mem_addr[]	Output	Memory address bus.
mem_ac_parity (1)	Output	Address or command parity signal generated by the PHY and sent to the DIMM. DDR3 SDRAM only.
mem_ba[]	Output	Memory bank address bus.
mem_cas_n	Output	Memory column address strobe signal.
mem_cke[]	Output	Memory clock enable signals.
mem_cs_n[]	Output	Memory chip select signals.
mem_dm[]	Output	Memory data mask signal, which masks individual bytes during writes.
mem_odt	Output	Memory on-die termination control signal.
mem_ras_n	Output	Memory row address strobe signal.
mem_we_n	Output	Memory write enable signal.
parity_error_n (1)	Output	Active-low signal that is asserted when a parity error occurs and stays asserted until the PHY is reset. DDR3 SDRAM only
mem_err_out_n (1)	Input	Signal sent from the DIMM to the PHY to indicate that a parity error has occured for a particular cycle. DDR3 SDRAM only.

Notes to Table 5–16:

(1) This signal is for registered DIMMs only.

Table 5–17 lists the CSR interface signals.

Table 5–17. CSR Interface Signals (Part 1 of 2)

Signal Name	Direction	Description
csr_addr[]	Input	Register map address.The width of csr_addr is 16 bits.
csr_be[]	Input	Byte-enable signal, which you use to mask off individual bytes during writes. csr_be is active high.
csr_clk (1)	Output	Clock for the configuration and status register (CSR) interface, which is the same as afi_clk and is always synchronous relative to the main data slave interface.
csr_wdata[]	Input	Write data bus. The width of csr_wdata is 32 bits.
csr_write_req	Input	Write request signal. You cannot assert csr_write_req and csr_read_req signals at the same time.
csr_read_req	Input	Read request signal. You cannot assert csr_read_req and csr_write_req signals at the same time.

Table 5–17. CSR Interface Signals (Part 2 of 2)

Signal Name	Direction	Description
csr_rdata[]	Output	Read data bus. The width of csr_rdata is 32 bits.
csr_rdata_valid	Output	Read data valid signal. The csr_rdata_valid signal indicates that valid data is present on the read data bus.
csr_waitrequest	Output	The csr_waitrequest signal indicates that the HPC II is busy and not ready to accept request signals. If the csr_waitrequest signal goes high in the clock cycle when a read or write request is asserted, that request is not accepted. If the csr_waitrequest signal goes low, the HPC II is then ready to accept more requests.

Note for Table 5–17:

(1) Applies only to the hard memory controller with multiport front end available in Arria V and Cyclone V devices.

Controller Register Map

The controller register map allows you to control the memory controller settings. Table 5–18 lists the register map for the controller.

Table 5–18. Controller Register Map (Part 1 of 5)

Address	Bit	Name	Default	Access	Description
0x100	0	Reserved.	0	—	Reserved for future use.
	1	Reserved.	0	—	Reserved for future use.
	2	Reserved.	0	—	Reserved for future use.
	7:3	Reserved.	0	—	Reserved for future use.
	13:8	Reserved.	0	—	Reserved for future use.
	30:14	Reserved.	0	—	Reserved for future use.

Table 5–18. Controller Register Map (Part 2 of 5)

Address	Bit	Name	Default	Access	Description
0x110	15:0	AUTO_PD_CYCLES	0x0	Read write	The number of idle clock cycles after which the controller should place the memory into power-down mode. The controller is considered to be idle if there are no commands in the command queue. Setting this register to 0 disables the auto power-down mode. The default value of this register depends on the values set during the generation of the design.
	16	Reserved.	0	—	Reserved for future use.
	17	Reserved.	0	—	Reserved for future use.
	18	Reserved.	0	—	Reserved for future use.
	19	Reserved.	0	—	Reserved for future use.
	21:20	ADDR_ORDER	00	Read write	00 - Chip, row, bank, column. 01 - Chip, bank, row, column. 10 - reserved for future use. 11 - Reserved for future use.
	22	Reserved.	0	—	Reserved for future use.
	24:23	Reserved.	0	—	Reserved for future use.
	30:24	Reserved	0	—	Reserved for future use.
0x120	7:0	Column address width	—	Read write	The number of column address bits for the memory devices in your memory interface. The range of legal values is 7-12.
	15:8	Row address width	—	Read write	The number of row address bits for the memory devices in your memory interface. The range of legal values is 12-16.
	19:16	Bank address width	—	Read write	The number of bank address bits for the memory devices in your memory interface. The range of legal values is 2-3.
	23:20	Chip select address width	—	Read write	The number of chip select address bits for the memory devices in your memory interface. The range of legal values is 0-2. If there is only one single chip select in the memory interface, set this bit to 0.
	31:24	Reserved.	0	—	Reserved for future use.
0x121	31:0	Data width representation (word)	—	Read only	The number of DQS bits in the memory interface. This bit can be used to derive the width of the memory interface by multiplying this value by the number of DQ pins per DQS pin (typically 8).
0x122	7:0	Chip select representation	—	Read only	The number of chip select in binary representation. For example, a design with 2 chip selects has the value of 00000011.
	31:8	Reserved.	0	—	Reserved for future use.

Table 5–18. Controller Register Map (Part 3 of 5)

Address	Bit	Name	Default	Access	Description
0x123	3:0	t_{RCD}	—	Read write	The activate to read or write a timing parameter. The range of legal values is 2-11 cycles.
	7:4	t_{RRD}	—	Read write	The activate to activate a timing parameter. The range of legal values is 2-8 cycles.
	11:8	t_{RP}	—	Read write	The precharge to activate a timing parameter. The range of legal values is 2-11 cycles.
	15:12	t_{MRD}	—	Read write	The mode register load time parameter. This value is not used by the controller, as the controller derives the correct value from the memory type setting.
	23:16	t_{RAS}	—	Read write	The activate to precharge a timing parameter. The range of legal values is 4-29 cycles.
	31:24	t_{RC}	—	Read write	The activate to activate a timing parameter. The range of legal values is 8-40 cycles.
0x124	3:0	t_{WTR}	—	Read write	The write to read a timing parameter. The range of legal values is 1-10 cycles.
	7:4	t_{RTP}	—	Read write	The read to precharge a timing parameter. The range of legal values is 2-8 cycles.
	15:8	t_{FAW}	—	Read write	The four-activate window timing parameter. The range of legal values is 6-32 cycles.
	31:16	Reserved.	0	—	Reserved for future use.
0x125	15:0	t_{REFI}	—	Read write	The refresh interval timing parameter. The range of legal values is 780-6240 cycles.
	23:16	t_{RFC}	—	Read write	The refresh cycle timing parameter. The range of legal values is 12-88 cycles.
	31:24	Reserved.	0	—	Reserved for future use.
0x126	3:0	Reserved.	0	—	Reserved for future use.
	7:4	Reserved.	0	—	Reserved for future use.
	11:8	Reserved.	0	—	Reserved for future use.
	15:12	Reserved.	0	—	Reserved for future use.
	23:16	Burst Length	—	Read write	Value must match memory burst length.
	31:24	Reserved.	0	—	Reserved for future use.

Table 5–18. Controller Register Map (Part 4 of 5)

Address	Bit	Name	Default	Access	Description
0x130	0	ENABLE_ECC	1	Read write	When this bit equals 1, it enables the generation and checking of ECC. This bit is only active if ECC was enabled during IP parameterization.
	1	ENABLE_AUTO_CORR	—	Read write	When this bit equals 1, it enables auto-correction when a single-bit error is detected.
	2	GEN_SBE	0	Read write	When this bit equals 1, it enables the deliberate insertion of single-bit errors, bit 0, in the data written to memory. This bit is used only for testing purposes.
	3	GEN_DBE	0	Read write	When this bit equals 1, it enables the deliberate insertion of double-bit errors, bits 0 and 1, in the data written to memory. This bit is used only for testing purposes.
	4	ENABLE_INTR	1	Read write	When this bit equals 1, it enables the interrupt output.
	5	MASK_SBE_INTR	0	Read write	When this bit equals 1, it masks the single-bit error interrupt.
	6	MASK_DBE_INTR	0	Read write	When this bit equals 1, it masks the double-bit error interrupt
	7	CLEAR	0	Read write	When this bit equals 1, writing to this self-clearing bit clears the interrupt signal, and the error status and error address registers.
	8	MASK_CORDROP_INTR	0	Read write	When this bit equals 1, the dropped autocorrection error interrupt is dropped.
	9	Reserved.	0	—	Reserved for future use.
0x131	0	SBE_ERROR	0	Read only	Set to 1 when any single-bit errors occur.
	1	DBE_ERROR	0	Read only	Set to 1 when any double-bit errors occur.
	2	CORDROP_ERROR	0	Read only	Value is set to 1 when any controller-scheduled autocorrections are dropped.
	7:3	Reserved.	0	—	Reserved for future use.
	15:8	SBE_COUNT	0	Read only	Reports the number of single-bit errors that have occurred since the status register counters were last cleared.
	23:16	DBE_COUNT	0	Read only	Reports the number of double-bit errors that have occurred since the status register counters were last cleared.
	31:24	CORDROP_COUNT	0	Read only	Reports the number of controller-scheduled autocorrections dropped since the status register counters were last cleared.
0x132	31:0	ERR_ADDR	0	Read only	The address of the most recent ECC error. This address is a memory burst-aligned local address.

Table 5–18. Controller Register Map (Part 5 of 5)

Address	Bit	Name	Default	Access	Description
0x133	31:0	CORDROP_ADDR	0	Read only	The address of the most recent autocorrection that was dropped. This is a memory burst-aligned local address.
0x134	0	REORDER_DATA	—	Read write	
	15:1	Reserved.	0	—	Reserved for future use.
	23:16	STARVE_LIMIT	0	Read write	Number of commands that can be served before a starved command.
	31:24	Reserved.	0	—	Reserved for future use.

Sequence of Operations

This section explains how the various blocks pass information in common situations.

Write Command

When a requesting master issues a write command together with write data, the following events occur:

- The input interface accepts the write command and the write data.
- The input interface passes the write command to the command generator and the write data to the write data buffer.
- The command generator processes the command and sends it to the timing bank pool.
- Once all timing requirements are met and a write-data-ready notification has been received from the write data buffer, the timing bank pool sends the command to the arbiter.
- When rank timing requirements are met, the arbiter grants the command request from the timing bank pool and passes the write command to the AFI interface.
- The AFI interface receives the write command from the arbiter and requests the corresponding write data from the write data buffer.
- The PHY receives the write command and the write data, through the AFI interface.

Read Command

When a requesting master issues a read command, the following events occur:

- The input interface accepts the read command.
- The input interface passes the read command to the command generator.
- The command generator processes the command and sends it to the timing bank pool.
- Once all timing requirements are met, the timing bank pool sends the command to the arbiter.

- When rank timing requirements are met, the arbiter grants the command request from the timing bank pool and passes the read command to the AFI interface.
- The AFI interface receives the read command from the arbiter and passes the command to the PHY.
- The PHY receives the read command through the AFI interface, and returns read data through the AFI interface.
- The AFI interface passes the read data from the PHY to the read data buffer.
- The read data buffer sends the read data to the master through the input interface.

Read-Modify-Write Command

A read-modify-write command can occur when enabling ECC for partial write, and for ECC correction commands. When a read-modify-write command is issued, the following events occur:

- The command generator issues a read command to the timing bank pool.
- The timing bank pool and arbiter passes the read command to the PHY through the AFI interface.
- The PHY receives the read command, reads data from the memory device, and returns the read data through the AFI interface.
- The read data received from the PHY passes to the ECC block.
- The read data is processed by the write data buffer.
- When the write data buffer issues a read-modify-write data ready notification to the command generator, the command generator issues a write command to the timing bank pool. The arbiter can then issue the write request to the PHY through the AFI interface.
- When the PHY receives the write request, it passes the data to the memory device.

Document Revision History

Table 5–19 lists the revision history for this document.

Table 5–19. Document Revision History

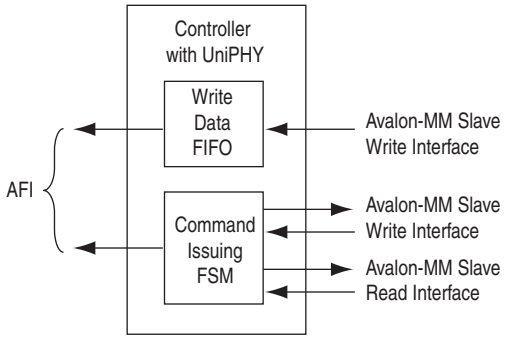
Date	Version	Changes
November 2012	2.1	■ Added Controller Register Map information. ■ Added Burst Merging information. ■ Updated User-Controlled Refresh Interface information. ■ Changed chapter number from 4 to 5.
June 2012	2.0	■ Added LPDDR2 support. ■ Added Feedback icon.
November 2011	1.1	■ Revised Figure 5–1. ■ Added AXI to Avalon-ST Converter information. ■ Added AXI Data Slave Interface information. ■ Added Half-Rate Bridge information.

The QDR II and QDR II+ controller translates memory requests from the Avalon Memory-Mapped (Avalon-MM) interface to AFI, while satisfying timing requirements imposed by the memory configurations. QDR II and QDR II+ SRAM has unidirectional data buses, therefore read and write operations are highly independent of each other and each has its own interface and state machine.

Block Description

This topic describes the blocks in the IP. Figure 6–1 shows a block diagram of the QDR II and QDR II+ SRAM controller architecture.

Figure 6–1. QDR II and QDR II+ SRAM Controller Architecture Block Diagram



Avalon-MM Slave Read and Write Interfaces

The read and write blocks accept read and write requests, respectively, from the Avalon-MM interface . Each block has a simple state machine that represents the state of the command and address registers, which stores the command and address when a request arrives.

The read data passes through without the controller registering it, as the PHY takes care of read latency. The write data goes through a pipeline stage to delay for a fixed number of cycles as specified by the write latency. In the full-rate burst length of four controller, the write data is also multiplexed into a burst of 2, which is then multiplexed again in the PHY to become a burst of 4 in DDR.

The user interface to the controller has separate read and write Avalon-MM interfaces because reads and writes are independent of each other in the memory device. The separate channels give efficient use of available bandwidth.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Command Issuing FSM

The command-issuing full-state machine (FSM) has two states: `INIT` and `INIT_COMPLETE`. In the `INIT_COMPLETE` state, commands are issued immediately as requests arrive using combinational logic and do not require state transitions.

AFI

In the full-rate burst length of two configuration, the controller can issue both read and write commands in the same clock cycle. In the memory device, both commands are clocked on the positive edge, but the read address is clocked on the positive edge, while the write address is clocked on the negative edge. Care must be taken on how these signals are ordered in the AFI.

For the half-rate burst length of four configuration the controller also issues both read and write commands, but the AFI width is doubled to fill two memory clocks per controller clock. As the controller only issues one write command and one read command per controller clock, the AFI read and write signals corresponding to the other memory cycle are tied to no operation (NOP).

For information on the AFI, refer to AFI 3.0 Specification.

Avalon-MM and Memory Data Width

Table 6–1 lists the data width ratio between the memory interface and the Avalon-MM interface. The half-rate controller does not support burst-of-2 devices because it under-uses the available memory bandwidth. Regardless of full or half-rate decision and the device burst length, the Avalon-MM interface must supply all the data for the entire memory burst in a single clock cycle. Therefore the Avalon-MM data width of the full-rate controller with burst-of-4 devices is four times the memory data width. For width-expanded configurations, the data width is further multiplied by the expansion factor (not shown in table 5-1 and 5-2).

Table 6–1. Data Width Ratio

Memory Burst Length	Half-Rate Designs	Full-Rate Designs
QDR II 2-word burst	No Support	2:1
QDR II and II+ 4-word burst	4:1	

Signal Description

This topic discusses the signals for each interface.

For information on the AFI signals, refer to AFI 3.0 Specification.

Avalon-MM Slave Read Interface

Table 6–2 lists the signals of the controller's Avalon-MM slave read interface.

Table 6–2. Avalon-MM Slave Read Signals

Signal	Width	Direction	Avalon-MM Signal Type
avl_r_ready	1	Out	waitrequest_n
avl_r_read_req	1	In	read
avl_r_addr	≤20	In	address
avl_r_rdata_valid	1	Out	readdatavalid
avl_r_rdata	16, 18, 36, 72, 144	Out	readdata
avl_r_size	log_2(MAX_BURST_SIZE) + 1	In	burstcount

The data width of the Avalon-MM interface is restricted to powers of two when using SOPC Builder or Qsys. Non-power-of-two data widths are supported when using the MegaWizard Plug-In Manager.

For more information about the Avalon interface, refer to *Avalon Interface Specifications*.

Avalon-MM Slave Write Interface

Table 6–3 lists the signals of the controller's Avalon-MM slave write interface.

Table 6–3. Avalon-MM Slave Write Signals

Signal	Width	Direction	Avalon-MM Signal Type
avl_w_ready	1	Out	waitrequest_n
avl_w_write_req	1	In	write
avl_w_addr	≤20	In	address
avl_w_wdata	18, 36, 72, 144	In	writedata
avl_w_be	2,4,8,16	In	byteenable
avl_w_size	log_2(MAX_BURST_SIZE) + 1	In	burstcount

For more information about the Avalon interface, refer to *Avalon Interface Specifications*.

Document Revision History

Table 6–4 lists the revision history for this document.

Table 6–4. Document Revision History

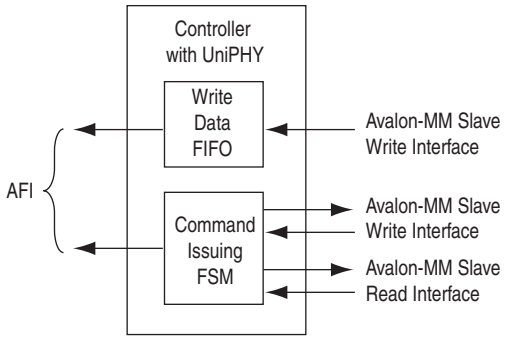
Date	Version	Changes
November 2012	3.3	Changed chapter number from 5 to 6.
June 2012	3.2	Added Feedback icon.
November 2011	3.1	Harvested Controller chapter from 11.0 **QDR II and QDR II+ SRAM Controller with UniPHY User Guide**.

The RLDRAM II controller translates memory requests from the Avalon Memory-Mapped (Avalon-MM) interface to AFI, while satisfying timing requirements imposed by the memory configurations.

Block Description

This topic describes the blocks in the IP. Figure 7–1 shows a block diagram of the RLDRAM II controller architecture.

Figure 7–1. RLDRAM II Controller Architecture Block Diagram



Avalon-MM Slave Interface

This Avalon-MM slave interface accepts read and write requests. A simple state machine represents the state of the command and address registers, which stores the command and address when a request arrives.

The Avalon-MM slave interface decomposes the Avalon-MM address to the memory bank, column, and row addresses. The IP automatically maps the bank address to the LSB of the Avalon address vector.

The Avalon-MM slave interface includes a burst adaptor, which has the following two parts:

- The first part is a read and write request combiner that groups requests to sequential addresses into the native memory burst. Given that the second request arrives within the read and write latency window of the first request, the controller can combine and satisfy both requests with a single memory transaction.
- The second part is the burst divider in the front end of the Avalon-MM interface, which breaks long Avalon bursts into individual requests of sequential addresses, which then pass to the controller state machine.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Write Data FIFO Buffer

The write data FIFO buffer accepts write data from the Avalon-MM interface. The AFI controls the subsequent consumption of the FIFO buffer write data.

Command Issuing FSM

The command issuing finite-state machine (FSM) has three states. The controller is in the `INIT` state when the PHY initializes the memory. Upon receiving the `afi_cal_success` signal, the state transitions to `INIT_COMPLETE`. If the calibration fails, `afi_cal_fail` is asserted and the state transitions to `INIT_FAIL`. The PHY receives commands only in the `INIT_COMPLETE` state.

When a refresh request arrives at the state machine at the same time as a read or write request, the refresh request takes precedence. The read or write request waits until there are no more refresh requests, and is issued immediately if timing requirements are met.

Refresh Timer

With automatic refresh, the refresh timer periodically issues refresh requests to the command issuing FSM. The refresh interval can be set at generation.

Timer Module

The timer module contains one DQ timer and eight bank timers (one per bank). The DQ timer tracks how often read and write requests can be issued, to avoid bus contention. The bank timers track the cycle time (t_{RC}).

The 8-bit wide output bus of the bank timer indicates to the command issuing FSM whether each bank can be issued a read, write, or refresh command.

AFI

For information on the AFI, refer to AFI 3.0 Specification.

User-Controlled Features

The following features are available on the **General Settings** tab of the parameter editor. These features are disabled by default.

Error Detection Parity

The error detection parity protection feature creates a simple parity encoder block which processes all read and write data. The error detection feature asserts an error signal if it detects any corrupted data during the read process. For every 8 bits of write data, a parity bit is generated and concatenated to the data before it is written to the memory. During the subsequent read operation, the parity bit is checked against the data bits to ensure data integrity.

Enabling the error detection parity protection feature reduces the local data width by one. For example, a nine-bit memory interface will present eight bits of data to the controller interface.

You can enable error detection parity protection in the **Controller Settings** section of the **General Settings** tab of the parameter editor.

User-Controlled Refresh

The user-controlled refresh feature allows you to take control of the refresh process that the controller normally performs automatically. You can control when refresh requests occur, and, if there are multiple memory devices, you control which bank receives the refresh signal. When you enable this feature, you disable auto-refresh, and assume responsibility for maintaining the necessary average periodic refresh rate.

You can enable user-controlled refresh in the **Controller Settings** section of the **General Settings** tab of the parameter editor.

Avalon-MM and Memory Data Width

Table 7–1 lists the data width ratio between the memory interface and the Avalon-MM interface. The half-rate controller does not support burst-of-2 devices because it under-uses the available memory bandwidth.

Table 7–1. Data Width Ratio

Memory Burst Length	Half-Rate Designs	Full-Rate Designs
2-word	No Support	2:1
4-word	4:1	
8-word		

Signal Description

This topic discusses the signals for each interface.

For information on the AFI signals, refer to AFI 3.0 Specification.

Avalon-MM Slave Interface

Table 7–2 lists the signals of the controller's Avalon-MM slave interface.

Table 7–2. Avalon-MM Slave Signals

Signal	Width	Direction	Avalon-MM Signal Type	Description
avl_size	1 to 11	In	burstcount	—
avl_ready	1	Out	waitrequest_n	—
avl_read_req	1	In	read	—
avl_write_req	1	In	write	—
avl_addr	≤25	In	address	—
avl_rdata_valid	1	Out	readdatavalid	—

Table 7–2. Avalon-MM Slave Signals

Signal	Width	Direction	Avalon-MM Signal Type	Description
avl_rdata	18, 36, 72, 144	Out	readdata	—
avl_wdata	18, 36, 72, 144	In	writedata	—

The data width of the Avalon-MM interface is restricted to powers of two when using SOPC Builder or Qsys. Non-power-of-two data widths are supported when using the MegaWizard Plug-In Manager.

Document Revision History

Table 7–3 lists the revision history for this document.

Table 7–3. Document Revision History

Date	Version	Changes
November 2012	3.3	Changed chapter number from 6 to 7.
June 2012	3.2	Added Feedback icon.
November 2011	3.1	Harvested Controller chapter from 11.0 **RLDRAM II Controller with UniPHY IP User Guide**.

8. Functional Description—RLDRAM 3 PHY-Only IP

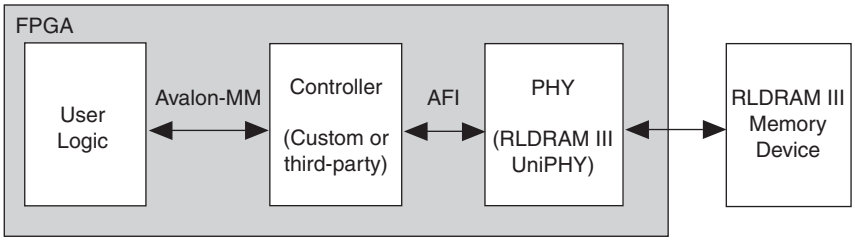
EMI_RM_018-1.0

The RLDRAM 3 PHY-only IP works with a customer-supplied memory controller to translate memory requests from user logic to RLDRAM 3 memory devices, while satisfying timing requirements imposed by the memory configurations.

Block Description

The RLDRAM 3 UniPHY-based IP is a PHY-only offering which you can use with a third-party controller or a controller that you develop yourself. Figure 8–1 shows a block diagram of the RLDRAM 3 system architecture.

Figure 8–1. RLDRAM 3 System Architecture



Features

The RLDRAM 3 UniPHY-based IP supports features available from major RLDRAM 3 device vendors at speeds of up to 800 MHz. The following list summarizes key features of the RLDRAM 3 UniPHY-based IP:

- support for Arria V GZ and Stratix V devices
- standard AFI interface between the PHY and the memory controller
- quarter-rate and half-rate AFI interface
- maximum frequency of 533 MHz for half-rate operation and 800 MHz for quarter-rate operation
- burst length of 2, 4, or 8
- x18 and x36 memory organization
- common I/O device support
- nonmultiplexed addressing
- multibank write and refresh protocol (programmable through mode register)
- optional use of data mask pins

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

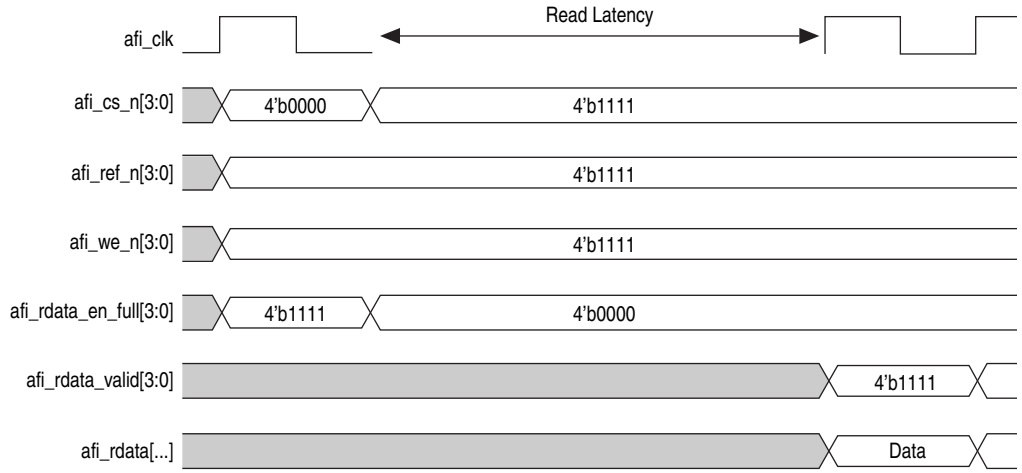
ISO 9001:2008 Registered

RLDRAM III AFI Protocol

The RLDRAM 3 UniPHY-based IP communicates with the memory controller using an AFI interface that follows the AFI 3.0 specification. To maximize bus utilization efficiency, the RLDRAM 3 UniPHY-based IP can issue multiple memory read/write operations within a single AFI cycle.

Figure 8–2 illustrates AFI bus activity when a quarter-rate controller issues four consecutive burst-length 2 read requests.

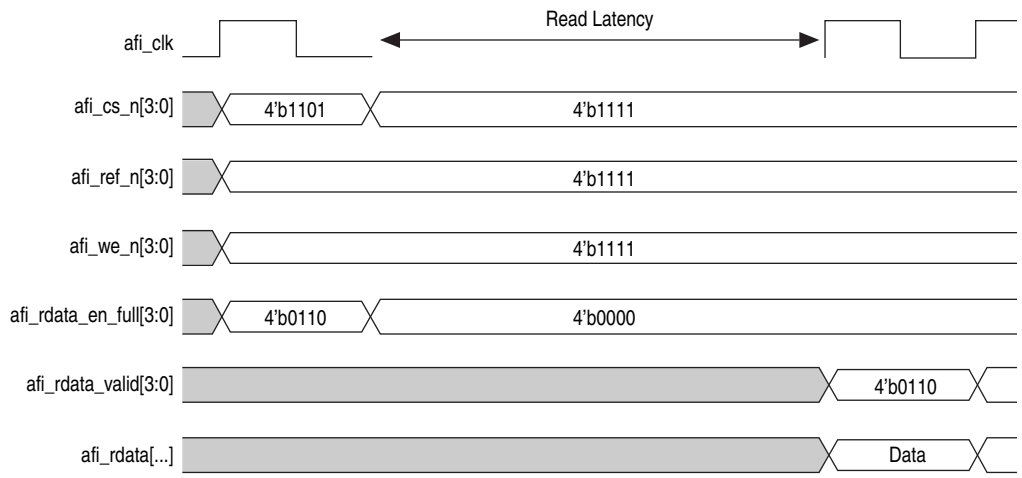
Figure 8–2. AFI Bus Activity for Quarter-Rate Controller Issuing Four Burst-Length 2 Read Requests



The controller does not have to begin a read or write command using channel 0 of the AFI bus. The flexibility afforded by being able to begin a command on any bus channel can facilitate command scheduling in the memory controller.

Figure 8–3 illustrates the AFI bus activity when a quarter-rate controller issues a single burst-length 4 read command to the memory on channel 1 of the AFI bus.

Figure 8–3. AFI Bus Activity for Quarter-Rate Controller Issuing One Burst-Length 4 Read Request



For information on the AFI, refer to AFI 3.0 Specification.

Document Revision History

Table 8–1 lists the revision history for this document.

Table 8–1. Document Revision History

Date	Version	Changes
November 2012	1.0	Initial release.

9. Functional Description—Example Designs

This chapter describes the example designs and the traffic generator.

Two independent example designs are created during generation with the MegaWizard Plug-In Manager. These example designs illustrate how to instantiate and connect the memory interface for both synthesis and simulation flows.

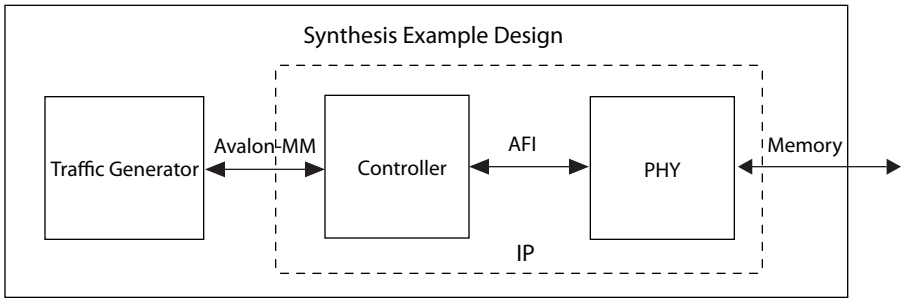
The two example designs are completely independent, and contain independent RTL files and other project files; they should be compiled or simulated separately, and the files should not be mixed. Nonetheless, the designs are related, as the simulation example design builds upon the design of the synthesis example design.

Synthesis Example Design

The synthesis example design contains the following major blocks, as shown in Figure 9–1:

- A traffic generator, which is a synthesizable Avalon-MM example driver that implements a pseudo-random pattern of reads and writes to a parameterized number of addresses. The traffic generator also monitors the data read from the memory to ensure it matches the written data and asserts a failure otherwise.
- An instance of the UniPHY memory interface, which includes a memory controller that moderates between the Avalon-MM interface and the AFI interface, and the UniPHY, which serves as an interface between the memory controller and external memory devices to perform read and write operations.

Figure 9–1. Synthesis Example Design

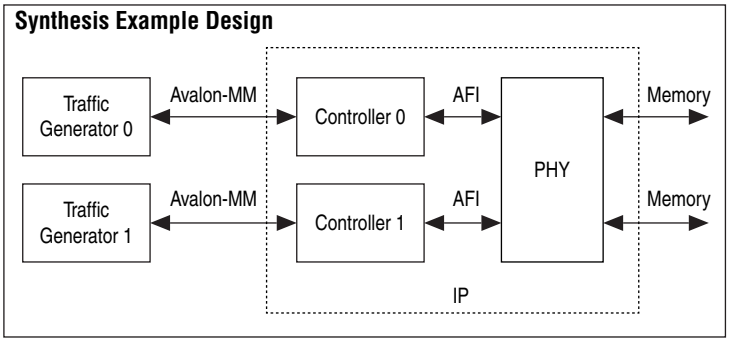


© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

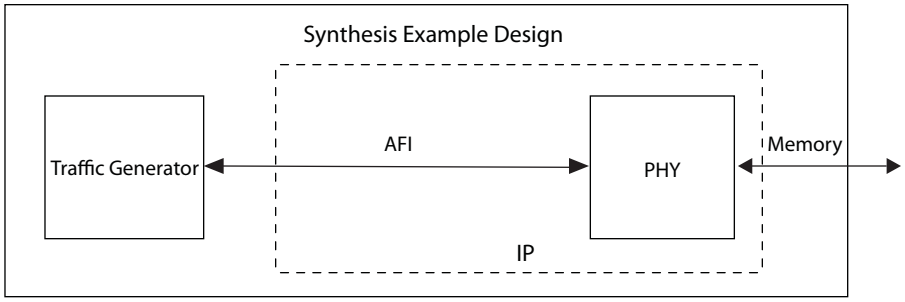
If you are using the Ping Pong PHY feature, the synthesis example design includes two traffic generators issuing commands to two independent memory devices through two independent controllers and a common PHY, as shown in Figure 9–2.

Figure 9–2. Synthesis Example Design for Ping Pong PHY



If you are using RLDRAM 3, the traffic generator in the synthesis example design communicates directly with the PHY using AFI, as shown in Figure 9–3.

Figure 9–3. Synthesis Example Design for RLDRAM 3 Interfaces



You can obtain the synthesis example design by generating your IP core using the MegaWizard Plug-In Manager flow. The files related to the synthesis example design reside at **<*variation_name*>_example_design/example_project**. The synthesis example design includes a Quartus II project file (**<*variation_name*>_example_design/example_project/<*variation_name*>_example.qpf**). The Quartus II project file can be compiled in the Quartus II software, and can be run on hardware.

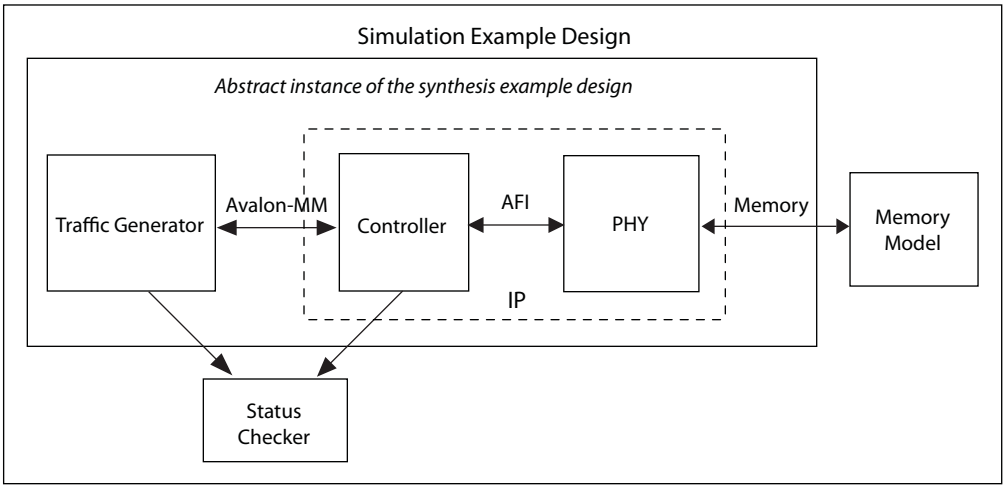
If one or more of the **PLL Sharing Mode**, **DLL Sharing Mode**, or **OCT Sharing Mode** parameters are set to any value other than **No Sharing**, the synthesis example design will contain two traffic generator/memory interface instances. The two traffic generator/memory interface instances are related only by shared PLL/DLL/OCT connections as defined by the parameter settings. The traffic generator/memory interface instances demonstrate how you can make such connections in your own designs.

Simulation Example Design

The simulation example design contains the following major blocks, as shown in Figure 9–4:

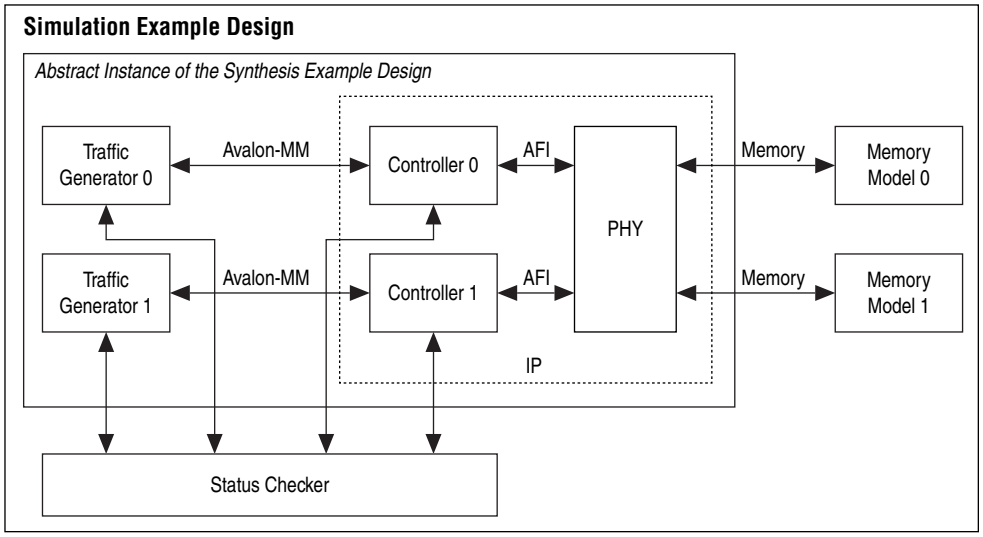
- An instance of the synthesis example design. As described in the previous section, the synthesis example design contains a traffic generator and an instance of the UniPHY memory interface. These blocks default to abstract simulation models where appropriate for rapid simulation.
- A memory model, which acts as a generic model that adheres to the memory protocol specifications. Frequently, memory vendors provide simulation models for specific memory components that you can download from their websites.
- A status checker, which monitors the status signals from the UniPHY IP and the traffic generator, to signal an overall pass or fail condition.

Figure 9–4. Simulation Example Design



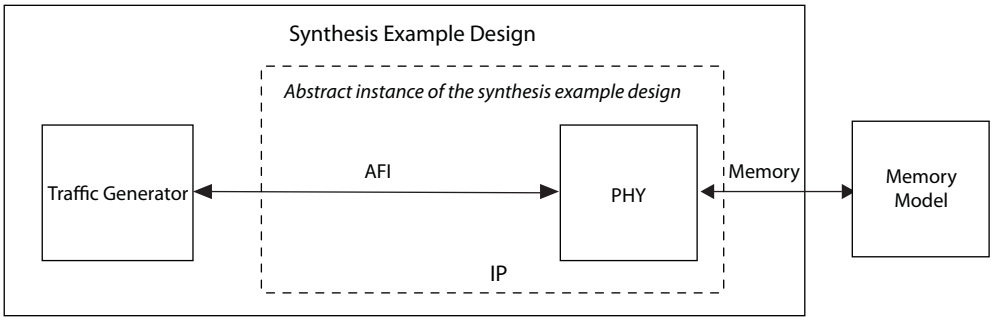
If you are using the Ping Pong PHY feature, the simulation example design includes two traffic generators issuing commands to two independent memory devices through two independent controllers and a common PHY, as shown in Figure 9–5.

Figure 9–5. Simulation Example Design for Ping Pong PHY



If you are using RLDRAM 3, the traffic generator in the simulation example design communicates directly with the PHY using AFI, as shown in Figure 9–6.

Figure 9–6. Simulation Example Design for RLDRAM 3 Interface



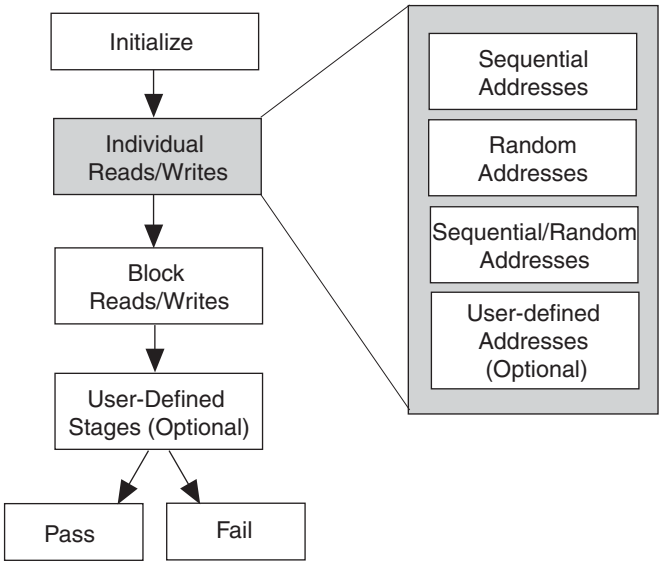
You can obtain the simulation example design by generating your IP core with the MegaWizard Plug-In Manager. The files related to the simulation example design reside at <*variation_name*>**_example_design/simulation**. After obtaining the files generated by the MegaWizard Plug-In Manager, you must still generate the simulation example design RTL for your desired HDL language. The file <*variation_name*>**_example_design/simulation/README.txt** contains details about how to generate the IP and to run the simulation in ModelSim-AE/SE.

Traffic Generator and BIST Engine

The traffic generator and built-in self test (BIST) engine for Avalon-MM memory interfaces generates Avalon-MM traffic on an Avalon-MM master interface. The traffic generator creates read and write traffic, stores the expected read responses internally, and compares the expected responses to the read responses as they arrive. If all reads report their expected response, the pass signal is asserted; however, if any read responds with unexpected data a fail signal occurs.

Each operation generated by the traffic generator is a single write or block of writes followed by a single read or block of reads to the same addresses, which allows the driver to precisely determine the data that should be expected when the read data is returned by the memory interface. The traffic generator comprises a traffic generation block, the Avalon-MM interface and a read comparison block. The traffic generation block generates addresses and write data, which are then sent out over the Avalon-MM interface. The read comparison block compares the read data received from the Avalon-MM interface to the write data from the traffic generator. If at any time the data received is not the expected data, the read comparison block records the failure, finishes reading all the data, and then signals that there is a failure and the traffic generator enters a fail state. If all patterns have been generated and compared successfully, the traffic generator enters a pass state.

Figure 9–7. Example Driver Operations



Within the traffic generator, there are the following main states:

- Generation of individual read and writes
- Generation of block read and writes
- The pass state
- The fail state

Within each of the generation states there are the following substates:

- Sequential address generation
- Random address generation
- Mixed sequential and random address generation

For each of the states and substates, the order and number of operations generated for each substate is parameterizable—you can decide how many of each address pattern to generate, or can disable certain patterns entirely if you want. The sequential and random interleave substate takes in additions to the number of operations to generate. An additional parameter specifies the ratio of sequential to random addresses to generate randomly.

Read and Write Generation

The traffic generator block can generate individual or block reads and writes.

Individual Read and Write Generation

During the traffic generator's individual read and write generation state, the traffic generation block generates individual write followed by individual read Avalon-MM transactions, where the address for the transactions is chosen according to the specific substate. The width of the Avalon-MM interface is a global parameter for the driver, but each substate can have a parameterizable range of burst lengths for each operation.

Block Read and Write Generation

During the traffic generator's block read and write generation state, the traffic generator block generates a parameterizable number of write operations followed by the same number of read operations. The specific addresses generated for the blocks are chosen by the specific substates. The burst length of each block operation can be parameterized by a range of acceptable burst lengths.

Address and Burst Length Generation

The traffic generator block can perform sequential or random addressing.

Sequential Addressing

The sequential addressing substate defines a traffic pattern where addresses are chosen in sequential order starting from a user definable address. The number of operations in this substate is parameterizable.

Random Addressing

The random addressing substate defines a traffic pattern where addresses are chosen randomly over a parameterizable range. The number of operations in this substate is parameterizable.

Sequential and Random Interleaved Addressing

The sequential and random interleaved addressing substate defines a traffic pattern where addresses are chosen to be either sequential or random based on a parameterizable ratio. The acceptable address range is parameterizable as is the number of operations to perform in this substate.

Traffic Generator Signals

Table 9–1 lists the signals used by the traffic generator.

Table 9–1. Traffic Generator Signals

Signal	Width	Signal Type
clk		
reset_n		
avl_ready		avl_ready
avl_write_req		avl_write_req
avl_read_req		avl_read_req
avl_addr	24	avl_addr
avl_size	3	avl_size
avl_wdata	72	avl_wdata
avl_rdata	72	avl_rdata
avl_rdata_valid		avl_rdata_valid
pnf_per_bit		pnf_per_bit
pnf_per_bit_persist		pnf_per_bit_persist
pass		pass
fail		fail
test_complete		test_complete

For information about the Avalon signals and the Avalon interface, refer to *Avalon Interface Specifications*.

Traffic Generator Add-Ons

Some optional components that can be useful for verifying aspects of the controller and PHY operation are generated in conjunction with certain user-specified options. These add-on components are self-contained, and are not part of the controller or PHY, nor the traffic generator.

User Refresh Generator

The user refresh generator sends refresh requests to the memory controller when user refresh is enabled. The memory controller returns an acknowledgement signal and then issues the refresh command to the memory device.

The user refresh generator is created when you turn on **Enable User Refresh** on the **Controller Settings** tab of the parameter editor.

Traffic Generator Timeout Counter

The traffic generator timeout counter uses the Avalon interface clock.

When a test fails due to driver failure or timeout, the `fail` signal is asserted. When a test has failed, the traffic generator must be reset with the `reset_n` signal.

Creating and Connecting the UniPHY Memory Interface and the Traffic Generator in Qsys

The traffic generator can be used in Qsys as a stand-alone component for use within a larger system. This section explains how to instantiate and configure the example driver, and includes tips on configuring a UniPHY memory interface which can apply in many general situations.

If you are using RLDRAM 3, you cannot use the example driver because the RLDRAM 3 IP is PHY-only, and does not include a controller.

Creating the Qsys System

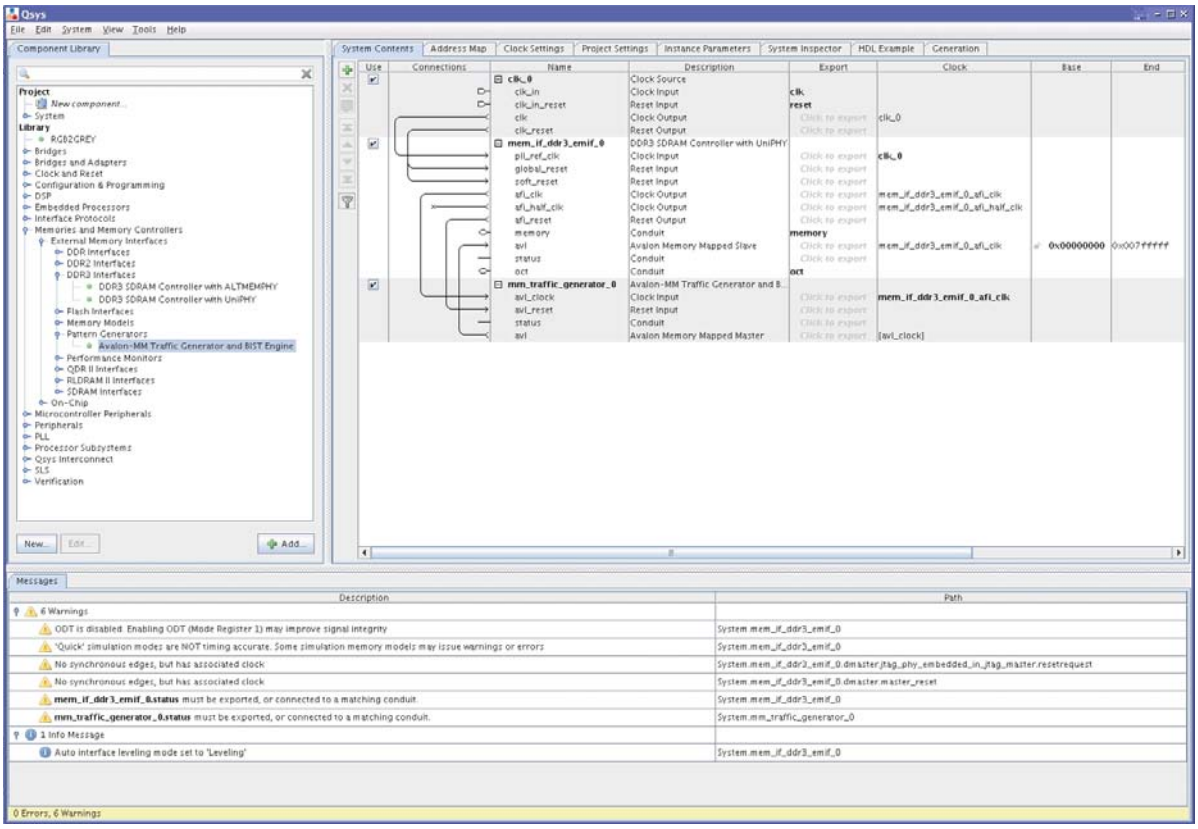
To create the system in Qsys, perform the following steps:

1. Start Qsys.
2. On the **Project Settings** tab, select the required device from the **Device Family** list.
3. In the **Component Library**, choose a UniPHY memory interface to instantiate. For example, under **Library** > **Memories and Memory Controllers** > **External Memory Interfaces**, select **DDR3 SDRAM Controller with UniPHY**.
4. Configure the parameters for your instantiation of the memory interface.
5. In the Component Library, find the example driver and instantiate it in the system. For example, under **Library** > **Memories and Memory Controllers** > **Pattern Generators**, select **Avalon-MM Traffic Generator and BIST Engine**.
6. Configure the parameters for your instantiation of the example driver.

The Avalon specification stipulates that Avalon-MM master interfaces issue byte addresses, while Avalon-MM slave interfaces accept word addresses. The default for the Avalon-MM Traffic Generator and BIST Engine is to issue word addresses. When using Qsys or SOPC Builder, you must enable the **Generate per byte address** setting in the traffic generator.

7. Connect the interfaces as illustrated in Figure 9–8. At this point, you can generate synthesis RTL, Verilog or VHDL simulation RTL, or a simulation testbench system.

Figure 9–8. Qsys System



Notes on Configuring UniPHY IP in Qsys

This section includes notes and tips on configuring the UniPHY IP in Qsys.

- The address ranges shown for the Avalon-MM slave interface on the UniPHY component should be interpreted as byte addresses that an Avalon-MM master would address, despite the fact that this range is modified by configuring the word addresses width of the Avalon-MM slave interface on the UniPHY controller.
- The `afi_clk` clock source is the associated clock to the Avalon-MM slave interface on the memory controller. This is the ideal clock source to use for all IP components connected on the same Avalon network. Using another clock would cause Qsys to automatically instantiate clock-crossing logic, potentially degrading performance.
- The `afi_clk` clock rate is determined by the **Rate on Avalon-MM interface** setting on the UniPHY **PHY Settings** tab. The `afi_half_clk` clock interface has a rate which is further halved. For example, if **Rate on Avalon-MM interface** is set to **Half**, the `afi_clk` rate is half of the memory clock frequency, and the `afi_half_clk` is one quarter of the memory clock frequency.

- The `global_reset` input interface can be used to reset the UniPHY memory interface and the PLL contained therein. The `soft_reset` input interface can be used to reset the UniPHY memory interface but allow the PLL to remain locked. You can use the `soft_reset` input to reset the memory but to maintain the AFI clock output to other components in the system.
- Do not connect a reset request from a system component (such as a Nios II processor) to the UniPHY `global_reset_n` port. Doing so would reset the UniPHY PLL, which would propagate as a reset condition on `afi_reset` back to the requester; the resulting reset loop could freeze the system.
- Qsys generates an interconnect fabric for each Avalon network. The interconnect fabric is capable of burst and width adaptation. If your UniPHY memory controller is configured with an Avalon interface data width which is wider than an Avalon-MM master interface connected to it, you must enable the byte enable signal on the Avalon-MM slave interface, by checking the **Enable Avalon-MM byte-enable signal** checkbox on the **Controller Settings** tab in the parameter editor.
- If you have a point-to-point connection from an Avalon-MM master to the Avalon-MM slave interface on the memory controller, and if the Avalon data width and burst length settings match, then the Avalon interface data widths may be multiples of either a power of two or nine. Otherwise, you must enable **Generate power-of-2 data bus widths for Qsys or SOPC Builder** on the **Controller Settings** tab of the parameter editor.

Document Revision History

Table 9–2 lists the revision history for this document.

Table 9–2. Document Revision History

Date	Version	Changes
November 2012	1.3	■ Added block diagrams of simulation and synthesis example designs for RLDRAM 3 and Ping Pong PHY. ■ Changed chapter number from 7 to 9.
June 2012	1.2	■ Added Feedback icon.
November 2011	1.1	■ Added Synthesis Example Design and Simulation Example Design sections. ■ Added Creating and Connecting the UniPHY Memory Interface and the Traffic Generator in Qsys. ■ Revised Example Driver section as Traffic Generator and BIST Engine.

Section II. UniPHY Reference

This section provides reference information about the UniPHY-based external memory interface IP.

This section includes the following chapters:

- Chapter 10, Introduction to UniPHY IP
- Chapter 11, Latency for UniPHY IP
- Chapter 12, Timing Diagrams for UniPHY IP
- Chapter 13, UniPHY External Memory Interface Debug Toolkit
- Chapter 14, Upgrading to UniPHY-based Controllers from ALTMEMPHY-based Controllers

For information about the revision history for chapters in this section, refer to "Document Revision History" in each individual chapter.

10. Introduction to UniPHY IP

EMI_RM_008-2.1

The Altera® DDR2, DDR3, and LPDDR2 SDRAM controllers with UniPHY, QDR II and QDR II+ SRAM controllers with UniPHY, RLDRAM II controller with UniPHY, and RLDRAM 3 PHY-only IP provide low latency, high-performance, feature-rich interfaces to industry-standard memory devices. The DDR2, QDR II and QDR II+, and RLDRAM II controllers with UniPHY offer full-rate and half-rate interfaces, while the DDR3 controller with UniPHY and the RLDRAM 3 PHY-only IP offer half-rate and quarter-rate interfaces, and the LPDDR2 controller with UniPHY offers a half-rate interface.

The UniPHY IP is an interface between a memory controller and memory devices and performs read and write operations to the memory. The UniPHY IP creates the datapath between the memory device and the memory controller and user logic in various Altera devices.

The MegaWizard™ interface generates an example top-level project, consisting of an example driver, and your controller custom variation. The controller instantiates an instance of the UniPHY datapath.

The example top-level project is a fully-functional design that you can simulate, synthesize, and use in hardware. The example driver is a self-test module that issues read and write commands to the controller and checks the read data to produce the pass, fail, and test-complete signals.

☞ For device families not supported by the UniPHY-based designs, use the Altera ALTMEMPHY-based High Performance SDRAM Controller IP core.

If the UniPHY datapath does not match your requirements, you can create your own memory interface datapath using the ALTDLL, ALTDQ_DQS, ALTDQ_DQS2, ALTDQ, or ALTDQS megafunctions, available in the Quartus® II software, but you are then responsible for all aspects of the design including timing analysis and design constraints.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Release Information

Table 10–1 provides information about this release of the DDR2 and DDR3 SDRAM, QDR II and QDR II+ SRAM, and RLDRAM II controllers with UniPHY, and the RLDRAM 3 PHY-only IP.

Table 10–1. Release Information

Item	Protocol			
	DDR2, DDR3, LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Version	12.1	12.1	12.1	12.1
Release Date	November 2012	November 2012	November 2012	November 2012
Ordering Code	IP-DDR2/UNI IP-DDR3/UNI IP-SDRAM/LPDDR2	IP-QDRII/UNI	IP-RLDII/UNI	—

Altera verifies that the current version of the Quartus II software compiles the previous version of each MegaCore function. The *MegaCore IP Library Release Notes and Errata* report any exceptions to this verification. Altera does not verify compilation with MegaCore function versions older than one release.

Device Family Support

Table 10–2 defines the device support levels for Altera IP cores.

Table 10–2. Altera IP Core Device Support Levels

FPGA Device Families	HardCopy Device Families
Preliminary support—The IP core is verified with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.	**HardCopy Companion**—The IP core is verified with preliminary timing models for the HardCopy companion device. The IP core meets all functional requirements, but might still be undergoing timing analysis for the HardCopy device family. It can be used in production designs with caution.
Final support—The IP core is verified with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.	**HardCopy Compilation**—The IP core is verified with final timing models for the HardCopy device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.

Table 10–3 shows the level of support offered by each of the UniPHY-based external memory interface protocols for Altera device families.

Table 10–3. Device Family Support (Part 1 of 2)

Device Family	Support Level					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Arria® II GX	No support	No support	No support	Final	No support	No support
Arria II GZ	Final	Final	No support	Final	Final	No support
Arria V	Refer to the What's New in Altera IP page of the Altera website.					No support

Table 10–3. Device Family Support (Part 2 of 2)

Device Family	Support Level					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Arria V GZ	Refer to the What's New in Altera IP page of the Altera website.		No support	Refer to the What's New in Altera IP page of the Altera website.		
Cyclone V	Refer to the What's New in Altera IP page of the Altera website.					No support
HardCopy® III	Refer to the What's New in Altera IP page of the Altera website.		No support	Refer to the What's New in Altera IP page of the Altera website.		No support
HardCopy IV	Refer to the What's New in Altera IP page of the Altera website.		No support	Refer to the What's New in Altera IP page of the Altera website.		No support
Stratix® III	Final	Final (Only V_{CC} = 1.1V supported)	No support	Final	Final (Only V_{CC} = 1.1V supported)	No support
Stratix IV	Final	Final	No support	Final	Final	No support
Stratix V	Refer to the What's New in Altera IP page of the Altera website.		No support	Refer to the What's New in Altera IP page of the Altera website.		
Other device families	No support	No support	No support	No support	No support	No support

For information about features and supported clock rates for external memory interfaces, refer to the *External Memory Specifiation Estimator*.

Features

Table 10–4 summarizes key feature support for Altera's UniPHY-based external memory interfaces.

Table 10–4. Feature Support (Part 1 of 2)

Key Feature	Protocol					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
High-performance controller II (HPC II)	✓	✓	✓	—	—	—
Half-rate core logic and user interface	✓	✓	✓	✓	✓	✓
Full-rate core logic and user interface	✓	—	—	✓	✓	—
Quarter-rate core logic and user interface	—	✓ (1)	—	—	—	✓
Dynamically generated Nios II-based sequencer	✓	✓	✓	✓	✓	✓
Choice of RTL-based or dynamically generated Nios® II-based sequencer	—	—	—	✓ (2) (3)	✓	—
Available Efficiency Monitor and Protocol Checker	✓	✓	✓	—	✓	✓
DDR3L support	—	✓ (1)	—	—	—	—
UDIMM and RDIMM in any form factor	✓	✓ (4) (5)	—	—	—	—

Table 10–4. Feature Support (Part 2 of 2)

Key Feature	Protocol					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Multiple components in a single-rank UDIMM or RDIMM layout	✓	✓	—	—	—	—
LRDIMM	—	✓	—	—	—	—
Burst length (half-rate)	8	—	8 or 16	4	4 or 8	2, 4, or 8
Burst length (full-rate)	4	—	—	2 or 4	2, 4, or 8	—
Burst length (quarter-rate)	—	8	—	—	—	2, 4, or 8
Burst length of 8 and burst chop of 4 (on the fly)	—	✓	—	—	—	—
With leveling	✓ (240 MHz and above) (10)	✓ (9) (10)	—	—	—	✓
Without leveling	✓ (below 240 MHz)	—	✓	—	—	—
Maximum data width	144 bits (6)	144 bits (6)	32 bits	72 bits	72 bits	72 bits
Reduced controller latency	—	—	—	✓ (2) (7)	✓ (2) (7)	—
Read latency	—	—	—	1.5 (QDR II) 2 or 2.5 (QDR II+)	—	—
ODT (in memory device)	—	✓	—	✓ (QDR II + only)	✓	✓
x36 emulation mode	—	—	—	✓ (8) (11)	—	—

Notes:

(1) For Arria V GZ and Stratix V devices only.

(2) Not available in Arria II GX devices.

(3) Nios II-based sequencer not available for full-rate interfaces.

(4) For DDR3, the DIMM form is not supported in Arria II GX, Arria II GZ, Arria V, or Cyclone V devices.

(5) Arria II GZ uses leveling logic for discrete devices in DDR3 interfaces to achieve high speeds, but that leveling cannot be used to implement the DIMM form in DDR3 interfaces.

(6) For any interface with data width above 72 bits, you must use Quartus II software timing analysis of your complete design to determine the maximum clock rate.

(7) The maximum achievable clock rate when reduced controller latency is selected must be attained through Quatrus II software timing analysis of your complete design.

(8) Emulation mode allows emulation of a larger memory-width interface using multiple smaller memory-width interfaces. For example, an x36 QDR II or QDR II+ interface can be emulated using two x18 interfaces.

(9) The leveling delay on the board between first and last DDR3 SDRAM component laid out as a DIMM must be less than 0.69 t_{CK}.

(10) Leveling is not available for Arria V or Cyclone V devices.

(11) x36 emulation mode is not supported in Arria V, Arria V GZ, Cyclone V, or Stratix V devices.

Unsupported Features

Table 10–4 summarizes key feature support. Device, protocol, and architecture support is summarized in the Protocol Support Matrix in chapter 1 of volume 1 of this handbook.

System Requirements

The DDR2, DDR3, and LPDDR2 SDRAM controllers with UniPHY, QDR II and QDR II+ SRAM controllers with UniPHY, RLDRAM II controller with UniPHY, and RLDRAM 3 PHY-only IP are part of the MegaCore IP Library, which Altera distributes with the Quartus II software.

For system requirements and installation instructions, refer to *Altera Software Installation and Licensing*.

MegaCore Verification

Altera has carried out extensive random, directed tests with functional test coverage using industry-standard models to ensure the functionality of the external memory controllers with UniPHY. Altera's functional verification of the external memory controllers with UniPHY use modified Denali models, with certain assertions disabled.

Resource Utilization

This section lists resource utilization information for the external memory controllers with UniPHY for supported device families.

DDR2, DDR3, and LPDDR2 SDRAM Controllers with UniPHY

Table 10–5 shows typical resource usage of the DDR2, DDR3, and LPDDR2 SDRAM controllers with UniPHY in the current version of Quartus II software for Arria V devices.

Table 10–5. Resource Utilization in Arria V Devices (Part 1 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controiler
Controller						
DDR2 (Half rate)	8	2286	1404	4	6560	0
	64	2304	1379	17	51360	0
DDR2 (Fullrate)	32	0	0	0	0	1
DDR3 (Half rate)	8	2355	1412	4	6560	0
	64	2372	1440	17	51360	0
DDR3 (Full rate)	32	0	0	0	0	1

Table 10–5. Resource Utilization in Arria V Devices (Part 2 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controiler
LPDDR2 (Half rate)	8	2230	1617	4	6560	0
	32	2239	1600	10	25760	0
PHY						
DDR2 (Half rate)	8	1652	2015	34	141312	0
	64	1819	2089	34	174080	0
DDR2 (Fullrate)	32	1222	1415	34	157696	1
DDR3 (Half rate)	8	1653	1977	34	141312	0
	64	1822	2090	34	174080	0
DDR3 (Full rate)	32	1220	1428	34	157696	0
LPDDR2 (Half rate)	8	2998	3187	35	150016	0
	32	3289	3306	35	174592	0
Total						
DDR2 (Half rate)	8	4555	3959	39	148384	0
	64	4991	4002	52	225952	0
DDR2 (Fullrate)	32	1776	1890	35	158208	1
DDR3 (Half rate)	8	4640	3934	39	148384	0
	64	5078	4072	52	225952	0
DDR3 (Full rate)	32	1774	1917	35	158208	1
LPDDR2 (Half rate)	8	5228	4804	39	156576	0
	32	5528	4906	45	200352	0

Table 10–6 shows typical resource usage of the DDR2 and DDR3 SDRAM controllers with UniPHY in the current version of Quartus II software for Arria II GZ devices.

Table 10–6. Resource Utilization in Arria II GZ Devices (Part 1 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
Controller							
DDR2 (Half rate)	8	1,781	1,092	10	2	0	4,352
	16	1,784	1,092	10	4	0	8,704
	64	1,818	1,108	10	15	0	34,560
	72	1,872	1,092	10	17	0	39,168

Table 10–6. Resource Utilization in Arria II GZ Devices (Part 2 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
DDR2 (Full rate)	8	1,851	1,124	10	2	0	2,176
	16	1,847	1,124	10	2	0	4,352
	64	1,848	1,124	10	8	0	17,408
	72	1,852	1,124	10	9	0	19,574
DDR3 (Half rate)	8	1,869	1,115	10	2	0	4,352
	16	1,868	1,115	10	4	0	8,704
	64	1,882	1,131	10	15	0	34,560
	72	1,888	1,115	10	17	0	39,168
PHY							
DDR2 (Half rate)	8	2,560	2,042	183	22	0	157,696
	16	2,730	2,262	183	22	0	157,696
	64	3,606	3,581	183	22	0	157,696
	72	3,743	3,796	183	22	0	157,696
DDR2 (Full rate)	8	2,494	1,934	169	22	0	157,696
	16	2,652	2,149	169	22	0	157,696
	64	3,519	3,428	169	22	0	157,696
	72	3,646	3,642	169	22	0	157,696
DDR3 (Half rate)	8	2,555	2,032	187	22	0	157,696
	16	3,731	2,251	187	22	0	157,696
	64	3,607	3,572	187	22	0	157,696
	72	3,749	3,788	187	22	0	157,696
Total							
DDR2 (Half rate)	8	4,341	3,134	193	24	0	4,374
	16	4,514	3,354	193	26	0	166,400
	64	5,424	4,689	193	37	0	192,256
	72	5,615	4,888	193	39	0	196,864
DDR2 (Full rate)	8	4,345	3,058	179	24	0	159,872
	16	4,499	3,273	179	24	0	162,048
	64	5,367	4,552	179	30	0	175,104
	72	5,498	4,766	179	31	0	177,280
DDR3 (Half rate)	8	4,424	3,147	197	24	0	162,048
	16	5,599	3,366	197	26	0	166,400
	64	5,489	4,703	197	37	0	192,256
	72	5,637	4,903	197	39	0	196,864

Table 10–7 shows typical resource usage of the DDR2 and DDR3 SDRAM controllers with UniPHY in the current version of Quartus II software for Stratix III devices.

Table 10–7. Resource Utilization in Stratix III Devices (Part 1 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
Controller							
DDR2 (Half rate)	8	1,807	1,058	0	4	0	4,464
	16	1,809	1,058	0	6	0	8,816
	64	1,810	1,272	10	14	0	32,256
	72	1,842	1,090	10	17	0	39,168
DDR2 (Full rate)	8	1,856	1,093	0	4	0	2,288
	16	1,855	1,092	0	4	0	4,464
	64	1,841	1,092	0	10	0	17,520
	72	1,834	1,092	0	11	0	19,696
DDR3 (Half rate)	8	1,861	1,083	0	4	0	4,464
	16	1,863	1,083	0	6	0	8,816
	64	1,878	1,295	10	14	0	32,256
	72	1,895	1,115	10	17	0	39,168
PHY							
DDR2 (Half rate)	8	2,591	2,100	218	6	1	157,696
	16	2,762	2,320	218	6	1	157,696
	64	3,672	3,658	242	6	1	157,696
	72	3,814	3,877	242	6	1	157,696
DDR2 (Full rate)	8	2,510	1,986	200	6	1	157,696
	16	2,666	2,200	200	6	1	157,696
	64	3,571	3,504	224	6	1	157,696
	72	3,731	3,715	224	6	1	157,696
DDR3 (Half rate)	8	2,591	2,094	224	6	1	157,696
	16	2,765	2,314	224	6	1	157,696
	64	3,680	3,653	248	6	1	157,696
	72	3,819	3,871	248	6	1	157,696
Total							
DDR2 (Half rate)	8	4,398	3,158	218	10	1	162,160
	16	4,571	3,378	218	12	1	166,512
	64	5,482	4,930	252	20	1	189,952
	72	5,656	4,967	252	23	1	196,864
DDR2 (Full rate)	8	4,366	3,079	200	10	1	159,984
	16	4,521	3,292	200	10	1	162,160
	64	5,412	4,596	224	16	1	175,216
	72	5,565	4,807	224	17	1	177,392

Table 10–7. Resource Utilization in Stratix III Devices (Part 2 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
DDR3 (Half rate)	8	4,452	3,177	224	10	1	162,160
	16	4,628	3,397	224	12	1	166,512
	64	5,558	4,948	258	20	1	189,952
	72	5,714	4,986	258	23	1	196,864

Table 10–8 shows typical resource usage of the DDR2 and DDR3 SDRAM controllers with UniPHY in the current version of Quartus II software for Stratix IV devices.

Table 10–8. Resource Utilization in Stratix IV Devices (Part 1 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
Controller							
DDR2 (Half rate)	8	1,785	1,090	10	2	0	4,352
	16	1,785	1,090	10	4	0	8,704
	64	1,796	1,106	10	15	0	34,560
	72	1,798	1,090	10	17	0	39,168
DDR2 (Full rate)	8	1,843	1,124	10	2	0	2,176
	16	1,845	1,124	10	2	0	4,352
	64	1,832	1,124	10	8	0	17,408
	72	1,834	1,124	10	9	0	19,584
DDR3 (Half rate)	8	1,862	1,115	10	2	0	4,352
	16	1,874	1,115	10	4	0	8,704
	64	1,880	1,131	10	15	0	34,560
	72	1,886	1,115	10	17	0	39,168
PHY							
DDR2 (Half rate)	8	2,558	2,041	183	6	1	157,696
	16	2,728	2,262	183	6	1	157,696
	64	3,606	3,581	183	6	1	157,696
	72	3,748	3,800	183	6	1	157,696
DDR2 (Full rate)	8	2,492	1,934	169	6	1	157,696
	16	2,652	2,148	169	6	1	157,696
	64	3,522	3,428	169	6	1	157,696
	72	3,646	3,641	169	6	1	157,696
DDR3 (Half rate)	8	2,575	2,031	187	6	1	157,696
	16	2,732	2,251	187	6	1	157,696
	64	3,602	3,568	187	6	1	157,696
	72	3,750	3,791	187	6	1	157,696
Total							

Table 10–8. Resource Utilization in Stratix IV Devices (Part 2 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
DDR2 (Half rate)	8	4,343	3,131	193	8	1	162,048
	16	4,513	3,352	193	10	1	166,400
	64	5,402	4,687	193	21	1	192,256
	72	5,546	4,890	193	23	1	196,864
DDR2 (Full rate)	8	4,335	3,058	179	8	1	159,872
	16	4,497	3,272	179	8	1	162,048
	64	5,354	4,552	179	14	1	175,104
	72	5,480	4,765	179	15	1	177,280
DDR3 (Half rate)	8	4,437	3,146	197	8	1	162,048
	16	4,606	3,366	197	10	1	166,400
	64	5,482	4,699	197	21	1	192,256
	72	5,636	4,906	197	23	1	196,864

Table 10–9 shows typical resource usage of the DDR2 and DDR3 SDRAM controllers with UniPHY in the current version of Quartus II software for Arria V GZ and Stratix V devices.

Table 10–9. Resource Utilization in Arria V GZ and Stratix V Devices (Part 1 of 2)

Protocol	Memory Width (Bits)	Combinational LCs	Logic Registers	M20K Blocks	Memory (Bits)
Controller					
DDR2 (Half rate)	8	1,787	1,064	2	4,352
	16	1,794	1,064	4	8,704
	64	1,830	1,070	14	34,304
	72	1,828	1,076	15	38,400
DDR2 (Full rate)	8	2,099	1,290	2	2,176
	16	2,099	1,290	2	4,352
	64	2,126	1,296	7	16,896
	72	2,117	1,296	8	19,456
DDR3 (Quarter rate)	8	2,101	1,370	4	8,704
	16	2,123	1,440	7	16,896
	64	2,236	1,885	28	69,632
	72	2,102	1,870	30	74,880
DDR3 (Half rate)	8	1,849	1,104	2	4,352
	16	1,851	1,104	4	8,704
	64	1,853	1,112	14	34,304
	72	1,889	1,116	15	38,400

Table 10–9. Resource Utilization in Arria V GZ and Stratix V Devices (Part 2 of 2)

Protocol	Memory Width (Bits)	Combinational LCs	Logic Registers	M20K Blocks	Memory (Bits)
PHY					
DDR2 (Half rate)	8	2,567	1,757	13	157,696
	16	2,688	1,809	13	157,696
	64	3,273	2,115	13	157,696
	72	3,377	2,166	13	157,696
DDR2 (Full rate)	8	2,491	1,695	13	157,696
	16	2,578	1,759	13	157,696
	64	3,062	2,137	13	157,696
	72	3,114	2,200	13	157,696
DDR3 (Quarter rate)	8	2,209	2,918	18	149,504
	16	2,355	3,327	18	157,696
	64	3,358	5,228	18	182,272
	72	4,016	6,318	18	198,656
DDR3 (Half rate)	8	2,573	1,791	13	157,696
	16	2,691	1,843	13	157,696
	64	3,284	2,149	13	157,696
	72	3,378	2,200	13	157,696
Total					
DDR2 (Half rate)	8	4,354	2,821	15	162,048
	16	4,482	2,873	17	166,400
	64	5,103	3,185	27	192,000
	72	5,205	3,242	28	196,096
DDR2 (Full rate)	8	4,590	2,985	15	159,872
	16	4,677	3,049	15	162,048
	64	5,188	3,433	20	174,592
	72	5,231	3,496	21	177,152
DDR3 (Quarter rate)	8	4,897	4,844	23	158,720
	16	5,065	5,318	26	175,104
	64	6,183	7,669	47	252,416
	72	6,705	8,744	49	274,048
DDR3 (Half rate)	8	4,422	2,895	15	162,048
	16	4,542	2,947	17	166,400
	64	5,137	3,261	27	192,000
	72	5,267	3,316	28	196,096

QDR II and QDR II+ SRAM Controllers with UniPHY

Table 10–10 shows typical resource usage of the QDR II and QDR II+ SRAM controllers with UniPHY in the current version of Quartus II software for Arria V devices.

[8]

Table 10–10. Resource Utilization in Arria V Devices

PHY Rate	Memory Width (Bits)	Combinational ALUTs	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controiler
Controller						
Half	9	98	120	0	0	0
	18	96	156	0	0	0
	36	94	224	0	0	0
PHY						
Half	9	234	257	0	0	0
	18	328	370	0	0	0
	36	522	579	0	0	0
Total						
Half	9	416	377	0	0	0
	18	542	526	0	0	0
	36	804	803	0	0	0

Table 10–11 shows typical resource usage of the QDR II and QDR II+ SRAM controllers with UniPHY in the current version of Quartus II software for Arria II GX devices.

Table 10–11. Resource Utilization in Arria II GX Devices

PHY Rate	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Memory (Bits)	M9K Blocks
Half	9	620	701	0	0
	18	921	1122	0	0
	36	1534	1964	0	0
Full	9	584	708	0	0
	18	850	1126	0	0
	36	1387	1962	0	0

Table 10–12 shows typical resource usage of the QDR II and QDR II+ SRAM controllers with UniPHY in the current version of Quartus II software for Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V devices.

Table 10–12. Resource Utilization in Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices

PHY Rate	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Memory (Bits)	M9K Blocks
Half	9	602	641	0	0
	18	883	1002	0	0
	36	1457	1724	0	0
Full	9	586	708	0	0
	18	851	1126	0	0
	36	1392	1962	0	0

RLDRAM II Controller with UniPHY

Table 10–13 shows typical resource usage of the RLDRAM II controller with UniPHY in the current version of Quartus II software for Arria V devices.

Table 10–13. Resource Utilization in Arria V Devices

PHY Rate	Memory Width (Bits)	Combinational ALUTs	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controller
Controller						
Half	9	353	303	1	288	0
	18	350	324	2	576	0
	36	350	402	4	1152	0
PHY						
Half	9	295	474	0	0	0
	18	428	719	0	0	0
	36	681	1229	0	0	0
Total						
Half	9	705	777	1	288	0
	18	871	1043	2	576	0
	36	1198	1631	4	1152	0

Table 10–14 shows typical resource usage of the RLDRAM II controller with UniPHY in the current version of Quartus II software for Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V devices.

Table 10–14. Resource Utilization in Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices *(1)*

PHY Rate	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Memory (Bits)	M9K Blocks
Half	9	829	763	288	1
	18	1145	1147	576	2
	36	1713	1861	1152	4
Full	9	892	839	288	1
	18	1182	1197	576	1
	36	1678	1874	1152	2

Note to Table 10–14:

(1) Half-rate designs use the same amount of memory as full-rate designs, but the data is organized in a different way (half the width, double the depth) and the design may need more M9K resources.

Document Revision History

Table 10–15 lists the revision history for this document.

Table 10–15. Document Revision History

Date	Version	Changes
November 2012	2.1	■ Added RLDRAM 3 support ■ Added LRDIMM support ■ Added Arria V GZ support ■ Changed chapter number from 8 to 10.
June 2012	2.0	■ Added LPDDR2 support. ■ Moved Protocol Support Matrix to Volume 1. ■ Added Feedback icon.
November 2011	1.1	■ Combined Release Information, Device Family Support, Features list, and Unsupported Features list for DDR2, DDR3, QDR II, and RLDRAM II. ■ Added Protocol Support Matrix. ■ Combined Resource Utilization information for DDR2, DDR3, QDR II, and RLDRAM II. Updated data for 11.1.

11. Latency for UniPHY IP

EMI_RM_009-2.1

Altera defines read and write latencies in terms of memory clock cycles. There are two types of latencies that exists while designing with memory controllers—read and write latencies, which have the following definitions:

- Read latency—the amount of time it takes for the read data to appear at the local interface after initiating the read request.
- Write latency—the amount of time it takes for the write data to appear at the memory interface after initiating the write request.

Latency of the memory interface depends on its configuration and traffic patterns, therefore you should simulate your system to determine precise latency values. The numbers presented in this chapter are typical values meant only as guidelines.

Latency found in simulation may differ from latency found on the board, because functional simulation does not consider board trace delays and differences in process, voltage, and temperature. For a given design on a given board, the latency found may differ by one clock cycle (for full-rate designs), or two clock cycles (for quarter-rate or half-rate designs) upon resetting the board. The same design can yield different latencies on different boards.

For a half-rate controller, the local side frequency is half of the memory interface frequency. For a full-rate controller, the local side frequency is equal to the memory interface frequency.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

DDR2, DDR3, and LPDDR2

Table 11–1 shows the DDR2 SDRAM latency in full-rate memory clock cycles.

Table 11–1. DDR2 SDRAM Controller Latency (In Full-Rate Memory Clock Cycles) *(1) (2)*

Latency in Full-Rate Memory Clock Cycles							
Rate	**Controller Address & Command**	**PHY Address & Command**	**Memory Maximum Read**	**PHY Read Return**	**Controller Read Return**	**Round Trip**	**Round Trip Without Memory**
Half	10	EWL: 3	3–7	6	4	EWL: 26–30	EWL: 23
		OWL: 4				OWL: 27–31	OWL: 24
Full	5	0	3–7	4	10	22–26	19

Notes:

(1) EWL = Even write latency

(2) OWL = Odd write latency

Table 11–2 shows the DDR3 SDRAM latency in full-rate memory clock cycles.

Table 11–2. DDR3 SDRAM Controller Latency (In Full-Rate Memory Clock Cycles) *(1) (2) (3) (4)*

Latency in Full-Rate Memory Clock Cycles							
Rate	**Controller Address & Command**	**PHY Address & Command**	**Memory Maximum Read**	**PHY Read Return**	**Controller Read Return**	**Round Trip**	**Round Trip Without Memory**
Quarter	20	EWER : 8	5–11	EWER: 16	8	EWER: 57–63	EWER: 52
		EWOR: 8		EWOR: 17		EWOR: 58–64	EWOR: 53
		OWER: 11		OWER: 17		OWER: 61–67	OWER: 56
		OWOR: 11		OWOR: 14		OWOR: 58–64	OWOR: 53
Half	10	EWER: 3	5–11	EWER: 7	4	EWER: 29–35	EWER: 24
		EWOR: 3		EWOR: 6		EWOR: 28–34	EWOR: 23
		OWER: 4		OWER: 6		OWER: 29–35	OWER: 24
		OWOR: 4		OWOR: 7		OWOR: 30–36	OWOR: 25
Full	5	0	5–11	4	10	24–30	19

Notes:

(1) EWER = Even write latency and even read latency

(2) EWOR = Even write latency and odd read latency

(3) OWER = Odd write latency and even read latency

(4) OWOR = Odd write latency and odd read latency

Table 11–3 shows the LPDDR2 SDRAM latency in full-rate memory clock cycles.

Table 11–3. LPDDR2 SDRAM Controller Latency (In Full-Rate Memory Clock Cycles) *(1) (2) (3) (4)*

Latency in Full-Rate Memory Clock Cycles							
Rate	**Controller Address & Command**	**PHY Address & Command**	**Memory Maximum Read**	**PHY Read Return**	**Controller Read Return**	**Round Trip**	**Round Trip Without Memory**
Half	10	EWER: 3	5–11	EWER: 7	4	EWER: 29–35	EWER: 24
		EWOR: 3		EWOR: 6		EWOR: 28–34	EWOR: 23
		OWER: 4		OWER: 6		OWER: 29–35	OWER: 24
		OWOR: 4		OWOR: 7		OWOR: 30–36	OWOR: 25
Full	5	0	5–11	4	10	24–30	19

Notes:

(1) EWER = Even write latency and even read latency
(2) EWOR = Even write latency and odd read latency
(3) OWER = Odd write latency and even read latency
(4) OWOR = Odd write latency and odd read latency

QDR II and QDR II+

Table 11–4 shows the latency in full-rate memory clock cycles.

Table 11–4. QDR II Latency (In Full-Rate Memory Clock Cycles) *(1)*

Latency in Full-Rate Memory Clock Cycles							
Rate	**Controller Address & Command**	**PHY Address & Command**	**Memory Maximum Read**	**PHY Read Return**	**Controller Read Return**	**Round Trip**	**Round Trip Without Memory**
Half	2	1	1.5, 2.0, 2.5	RL 1.5: 5.5	0	RL 1.5: 10	RL 1.5: 8.5
				RL 2.0: 5.0		RL 2.0: 10	RL 2.0: 8
				RL 2.5 :4.5		RL 2.5: 10	RL 2.5: 7.5
Full	1	1	1.5, 2.0, 2.5	RL 1.5: 4.5	0	RL 1.5: 8	RL 1.5: 6.5
				RL 2.0: 4.0		RL 2.0: 8	RL 2.0: 6.0
				RL 2.5 :4.5		RL 2.5: 9	RL 2.5: 6.5

Note:

(1) RL = Read latency

RLDRAM II

Table 11–5 shows the latency in full-rate memory clock cycles.

Table 11–5. RLDRAM II Latency (In Full-Rate Memory Clock Cycles) (1) (2)

Latency in Full-Rate Memory Clock Cycles							
Rate	**Controller Address & Command**	**PHY Address & Command**	**Memory Maximum Read**	**PHY Read Return**	**Controller Read Return**	**Round Trip**	**Round Trip Without Memory**
Half	4	EWL: 1	3–8	EWL: 4	0	EWL: 12–17	EWL: 9
		OWL: 2		OWL: 4		OWL: 13–18	OWL: 10
Full	2	1	3–8	4	0	10–15	7

Notes:

(1) EWL = Even write latency

(2) OWL = Odd write latency

RLDRAM 3

Table 11–6 shows the latency in full-rate memory clock cycles.

Table 11–6. RLDRAM 3 Latency (In Full-Rate Memory Clock Cycles)

Latency in Full-Rate Memory Clock Cycles						
Rate	**PHY Address & Command**	**Memory Maximum Read**	**PHY Read Return**	**Controller Read Return**	**Round Trip**	**Round Trip Without Memory**
Quarter	7	3–16	18	0	28–41	25
Half	4	3–16	6	0	13–26	10

Variable Controller Latency

The variable controller latency feature allows you to take advantage of lower latency for variations designed to run at lower frequency. When deciding whether to vary the controller latency from the default value of 1, be aware of the following considerations:

- Reduced latency can help achieve a reduction in resource usage and clock cycles in the controller, but might result in lower f_{MAX}.
- Increased latency can help acheive greater f_{MAX}, but might consume more clock cycles in the controller and result in increased resource usage.

If you select a latency value that is inappropriate for the target frequency, the system displays a warning message in the text area at the bottom of the parameter editor.

You can change the controller latency by altering the value of the **Controller Latency** setting in the **Controller Settings** section of the **General Settings** tab of the QDR II and QDR II+ SRAM controller with UniPHY parameter editor.

Document Revision History

Table 11–7 lists the revision history for this document.

Table 11–7. Document Revision History

Date	Version	Changes
November 2012	2.1	■ Added latency information for RLDRAM 3. ■ Changed chapter number from 9 to 11.
June 2012	2.0	■ Added latency information for LPDDR2. ■ Added Feedback icon.
November 2011	1.0	■ Consolidated latency information from 11.0 **DDR2 and DDR3 SDRAM Controller with UniPHY User Guide**, **QDR II and QDR II+ SRAM Controller with UniPHY User Guide**, and **RLDRAM II Controller with UniPHY IP User Guide**. ■ Updated data for 11.1.

12. Timing Diagrams for UniPHY IP

EMI_RM_010-2.1

This chapter contains timing diagrams for the UniPHY-based external memory interface IP.

DDR2 and DDR3 Timing Diagrams

This section contains timing diagrams for DDR2 and DDR3 protocols.

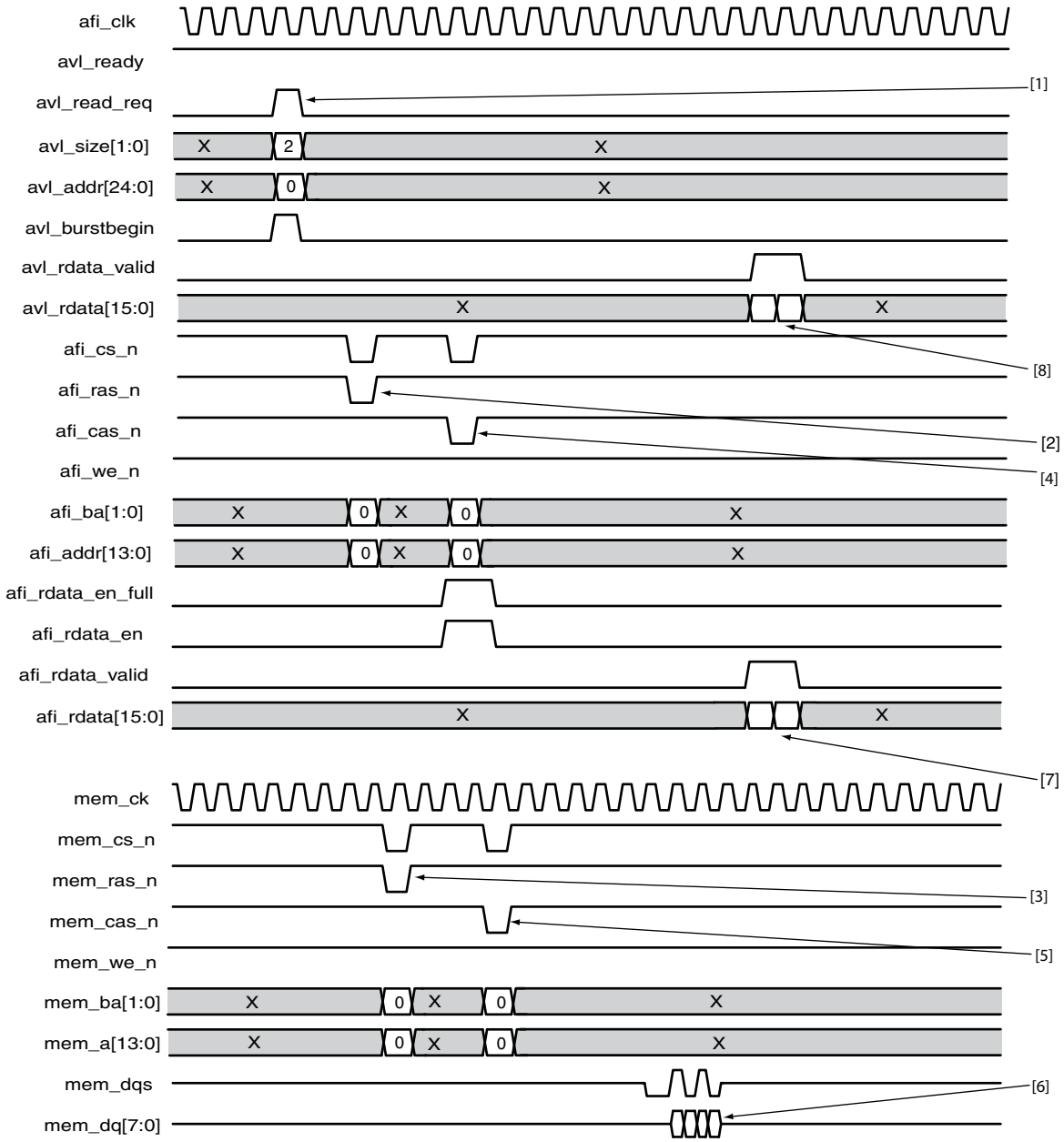
Figure 12–1 through Figure 12–8 present the following timing diagrams, based on a Stratix III device:

- Full-Rate DDR2 SDRAM Read
- Full-Rate DDR2 SDRAM Write
- Half-Rate DDR2 SDRAM Read
- Half-Rate DDR2 SDRAM Write
- Half-Rate DDR3 SDRAM Read
- Half-Rate DDR3 SDRAM Writes
- Quarter-Rate DDR3 SDRAM Reads
- Quarter-Rate DDR3 SDRAM Writes

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

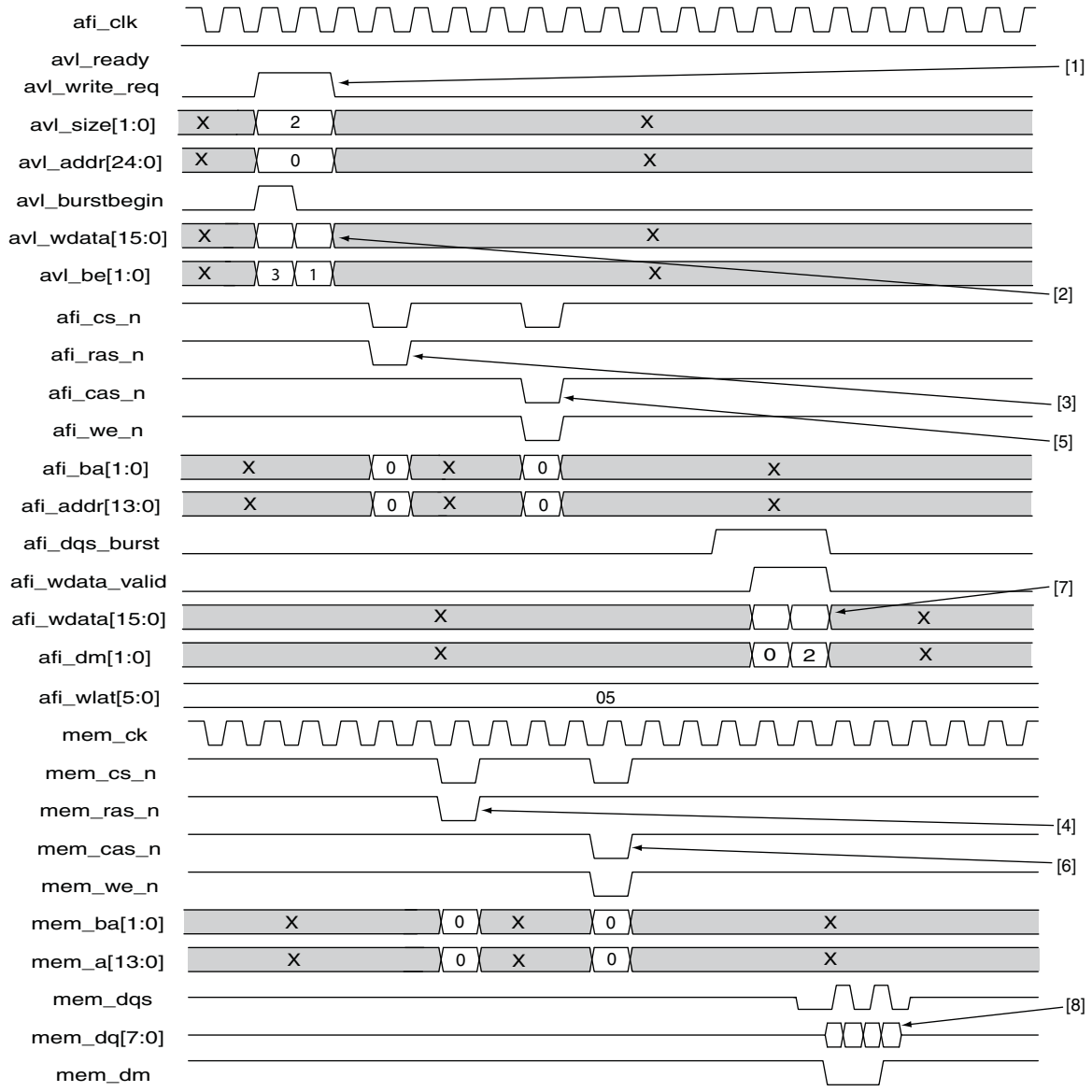
Figure 12–1. Full-Rate DDR2 SDRAM Read



Notes for Figure 12–1

(1) Controller receives read command.
(2) Controller issues activate command to PHY.
(3) PHY issues activate command to memory.
(4) Controller issues read command to PHY.
(5) PHY issues read command to memory.
(6) PHY receives read data from memory.
(7) Controller receives read data from PHY.
(8) User logic receives read data from controller.

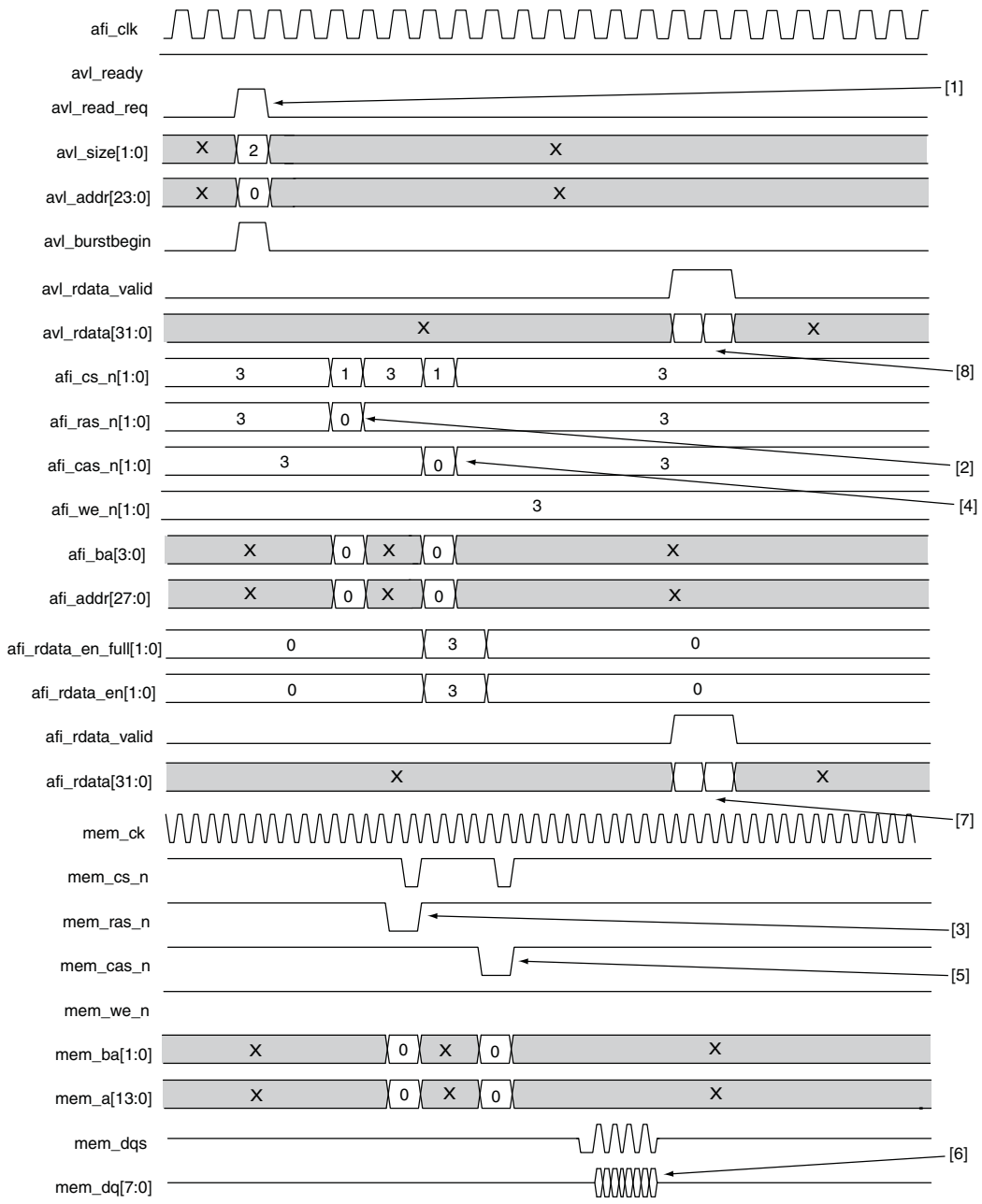
Figure 12–2. Full-Rate DDR2 SDRAM Write



Notes for Figure 12–2:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues activate command to PHY.
(4) PHY issues activate command to memory.
(5) Controller issues write command to PHY.
(6) PHY issues write command to memory.
(7) Controller sends write data to PHY.
(8) PHY sends write data to memory.

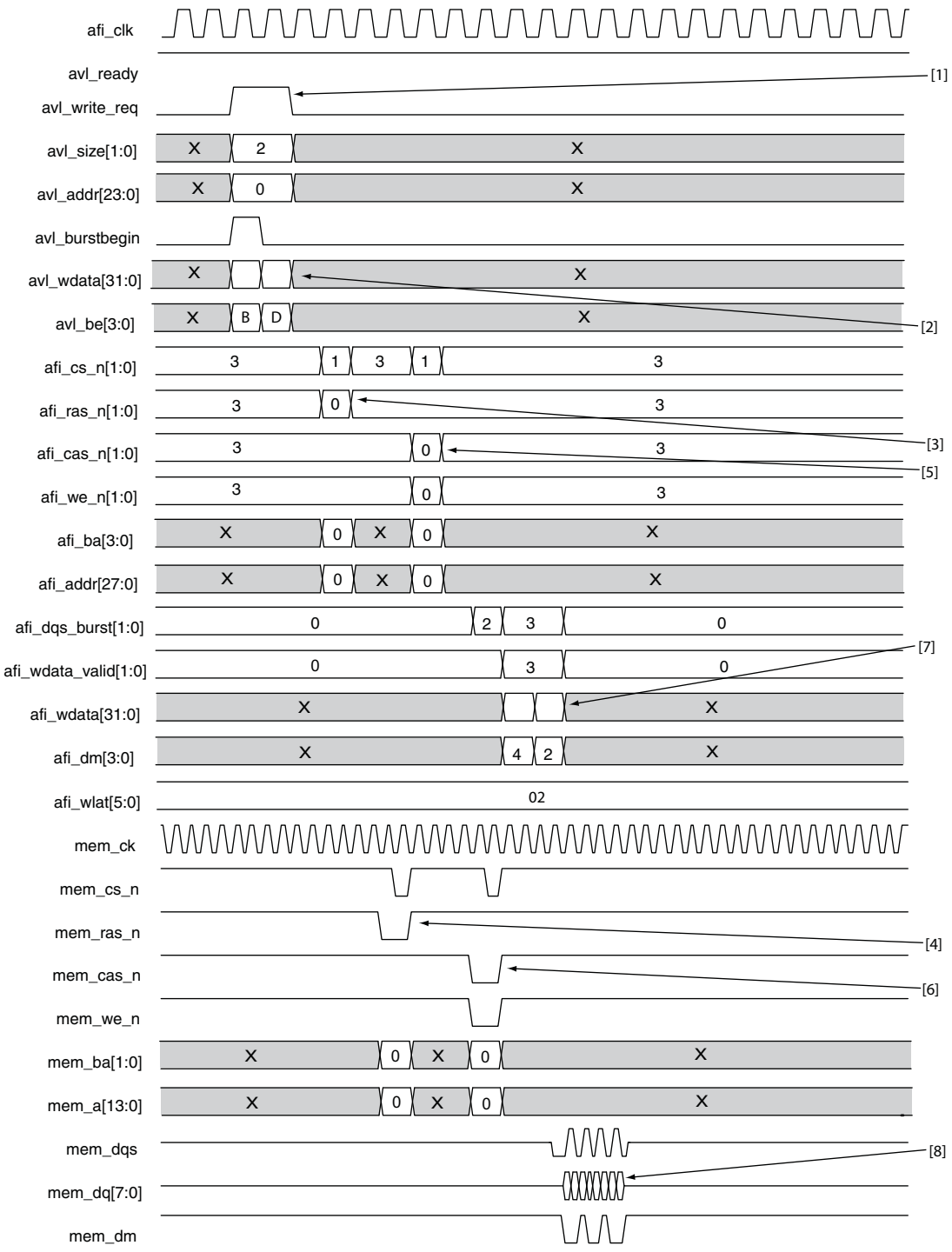
Figure 12–3. Half-Rate DDR2 SDRAM Read



Notes for Figure 12–3:

(1) Controller receives read command.
(2) Controller issues activate command to PHY.
(3) PHY issues activate command to memory.
(4) Controller issues read command to PHY.
(5) PHY issues read command to memory.
(6) PHY receives read data from memory.
(7) Controller receives read data from PHY.
(8) User logic receives read data from controller.

Figure 12–4. Half-Rate DDR2 SDRAM Write

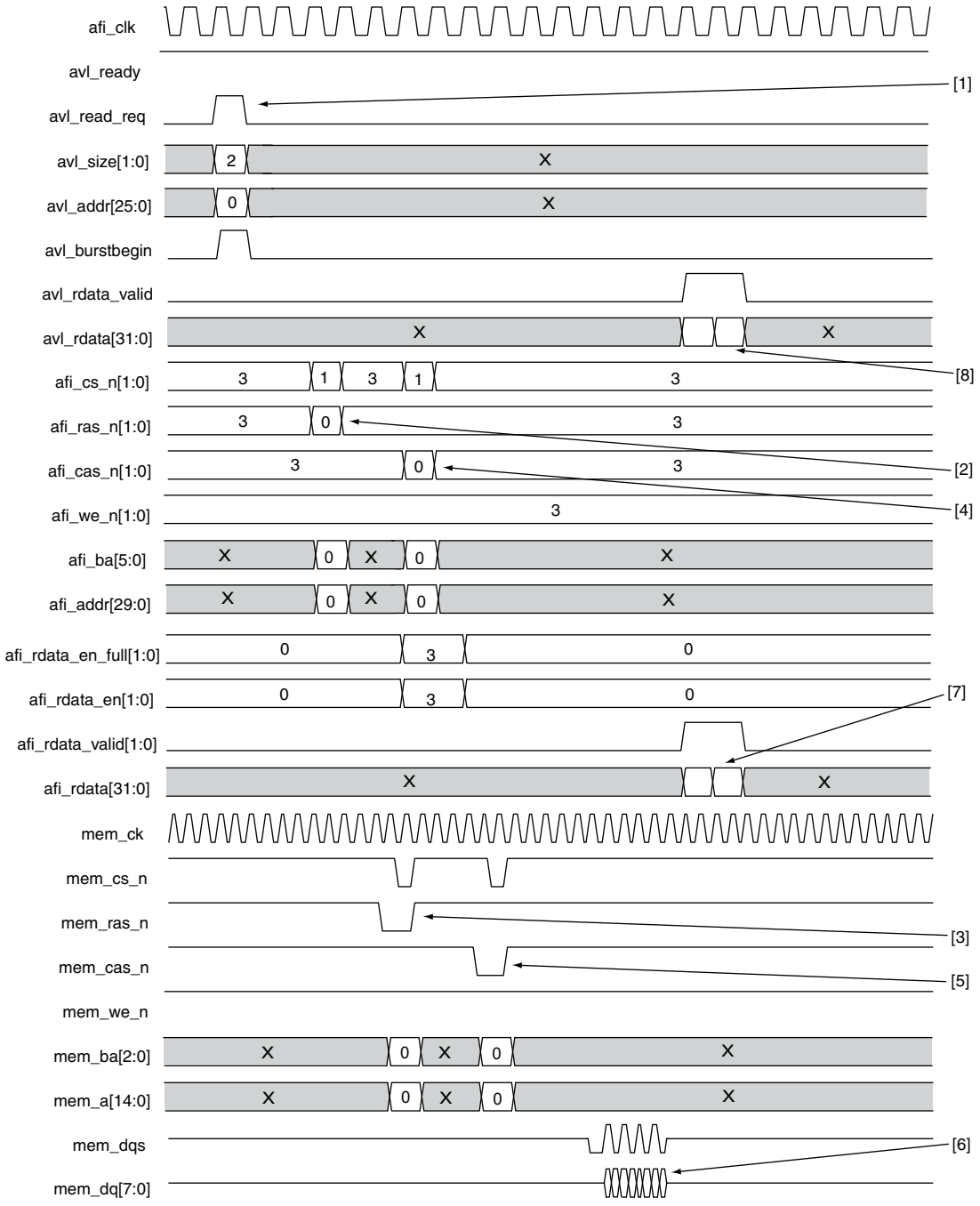


Notes for Figure 12–4:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues activate command to PHY.
(4) PHY issues activate command to memory.

(5) Controller issues write command to PHY.
(6) PHY issues write command to memory.
(7) Controller sends write data to PHY.
(8) PHY sends write data to memory.

Figure 12–5. Half-Rate DDR3 SDRAM Read

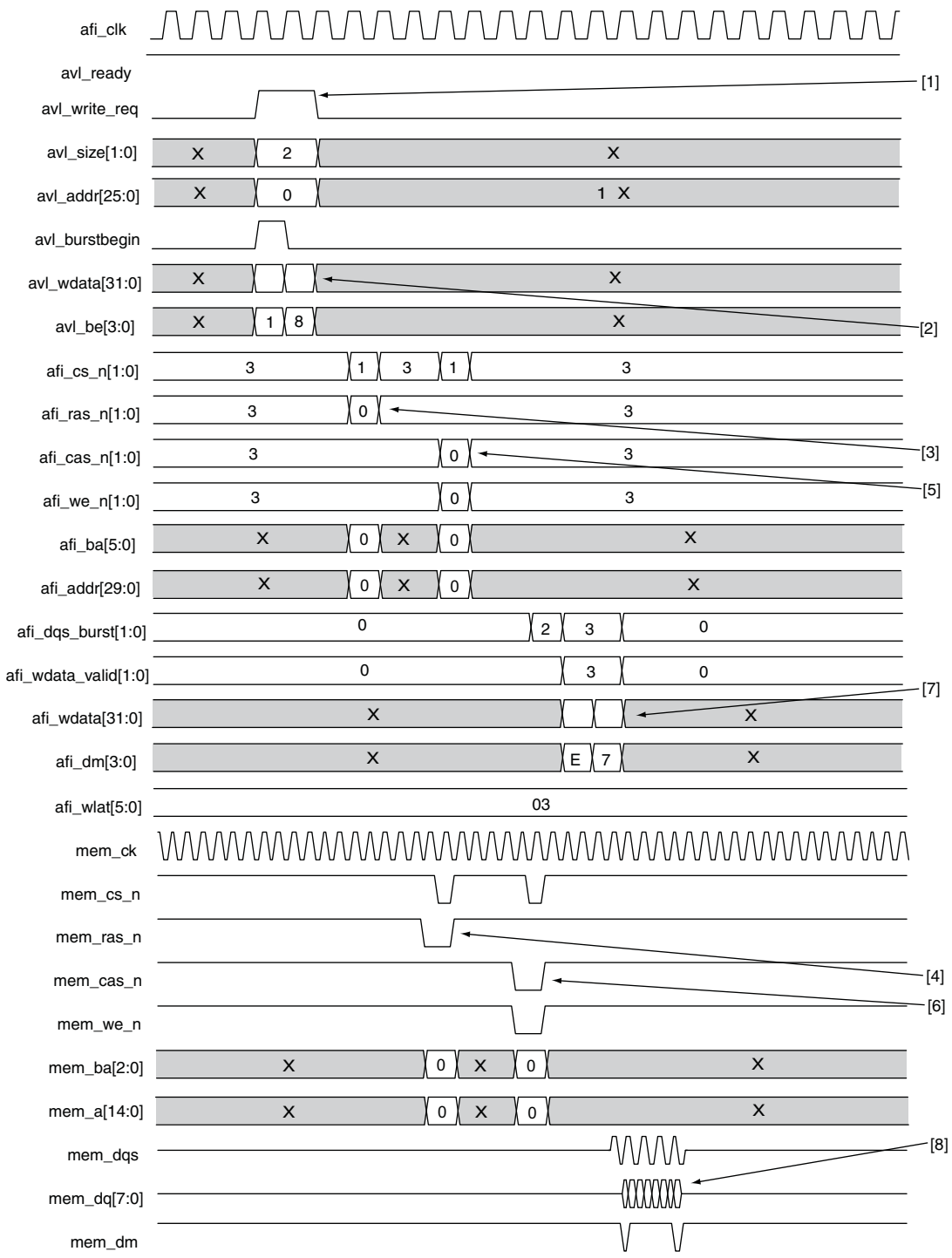


Notes for Figure 12–5:

(1) Controller receives read command.
(2) Controller issues activate command to PHY.
(3) PHY issues activate command to memory.

(4) Controller issues read command to PHY.
(5) PHY issues read command to memory.
(6) PHY receives read data from memory.
(7) Controller receives read data from PHY.
(8) User logic receives read data from controller.

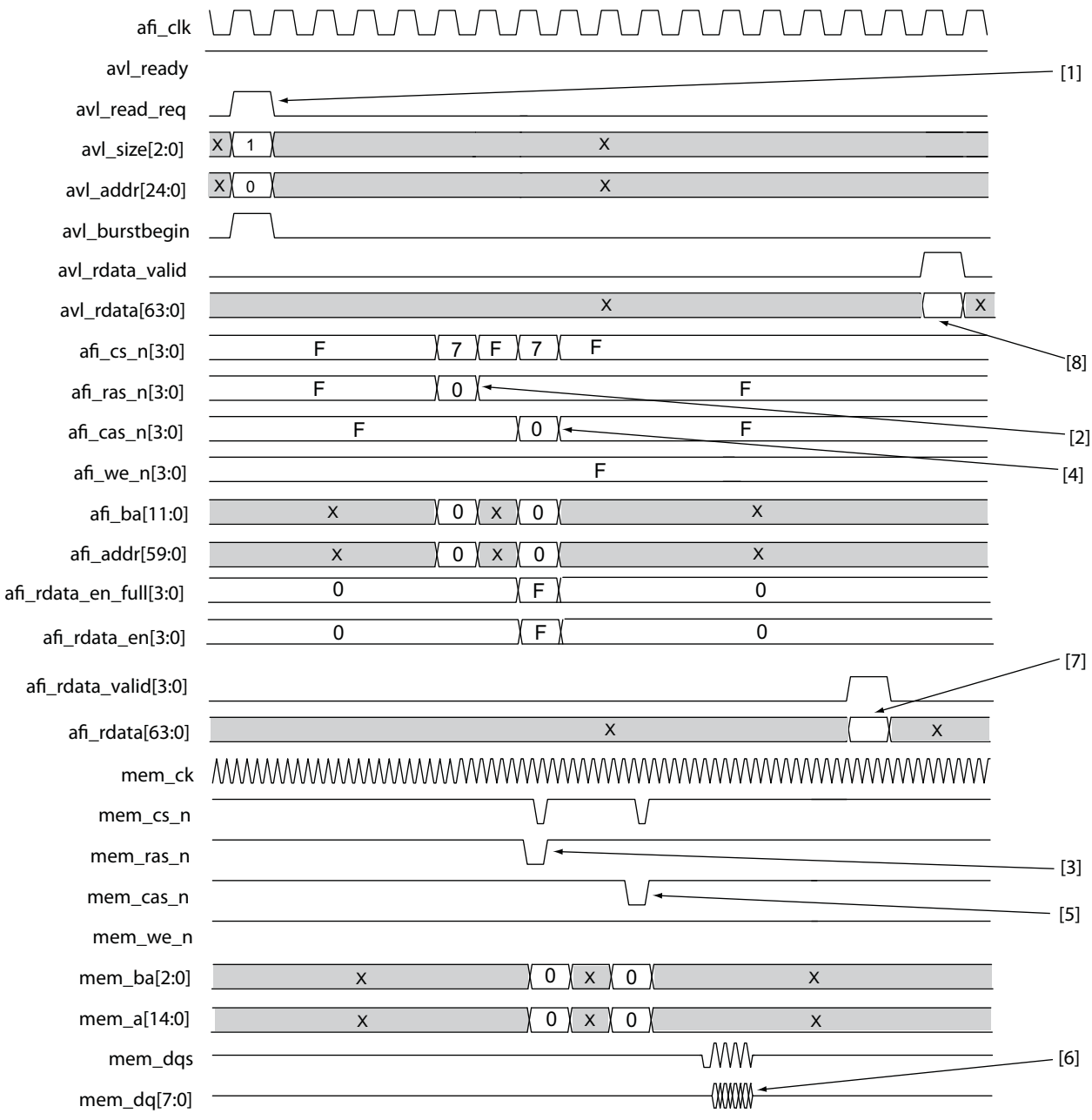
Figure 12–6. Half-Rate DDR3 SDRAM Writes



Notes for Figure 12–6:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues activate command to PHY.
(4) PHY issues activate command to memory.
(5) Controller issues write command to PHY.
(6) PHY issues write command to memory.
(7) Controller sends write data to PHY.
(8) PHY sends write data to memory.

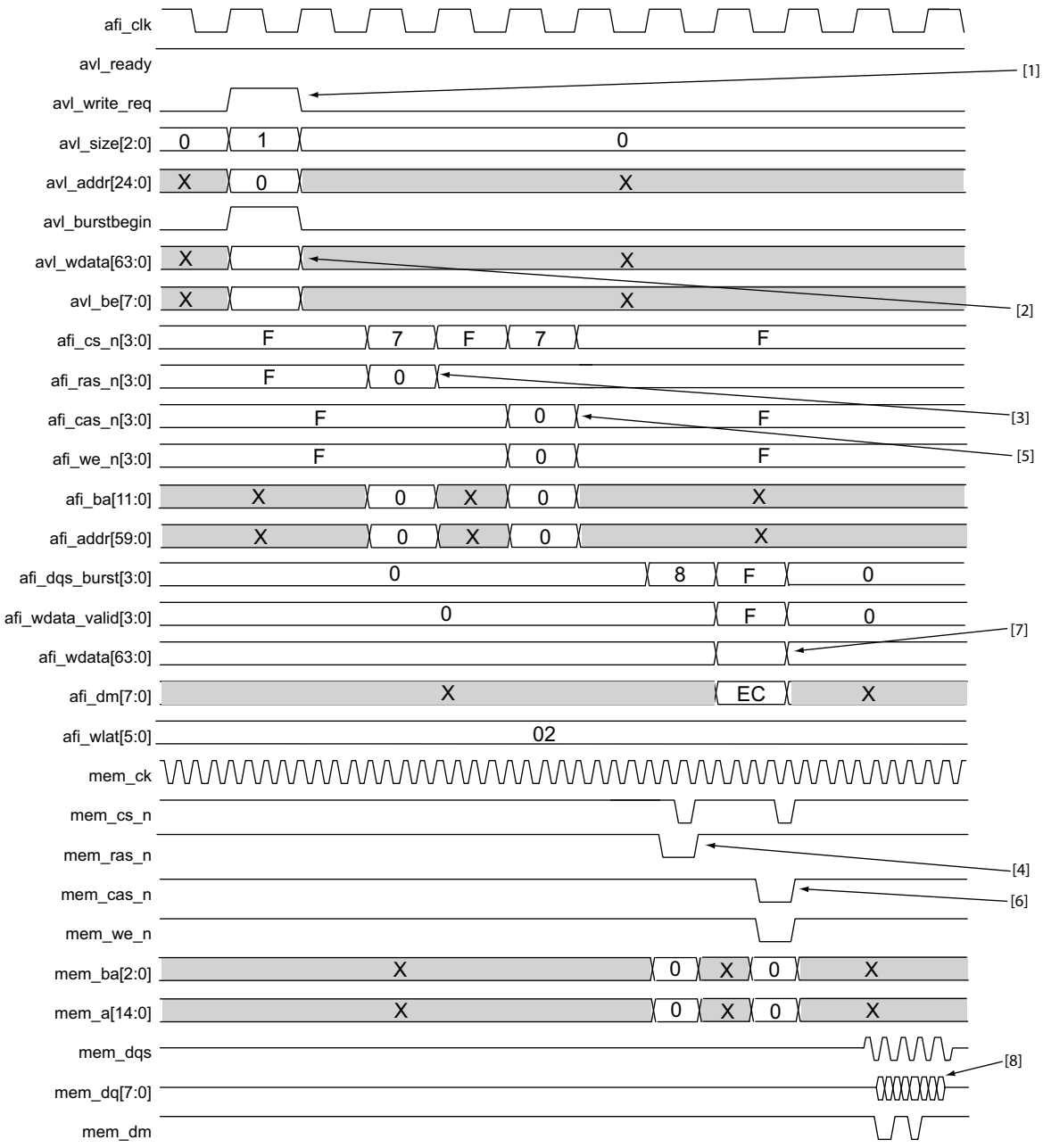
Figure 12–7. Quarter-Rate DDR3 SDRAM Reads



Notes for Figure 12–7:

(1) Controller receives read command.
(2) Controller issues activate command to PHY.
(3) PHY issues activate command to memory.
(4) Controller issues read command to PHY.
(5) PHY issues read command to memory.
(6) PHY receives read data from memory
(7) Controller receives read data from PHY
(8) User logic receives read data from controller.

Figure 12–8. Quarter-Rate DDR3 SDRAM Writes



Notes for Figure 12–8:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues activate command to PHY
(4) PHY issues activate command to memory.
(5) Controller issues write command to PHY
(6) PHY issues write command to memory
(7) Controller sends write data to PHY
(8) PHY sends write data to memory.

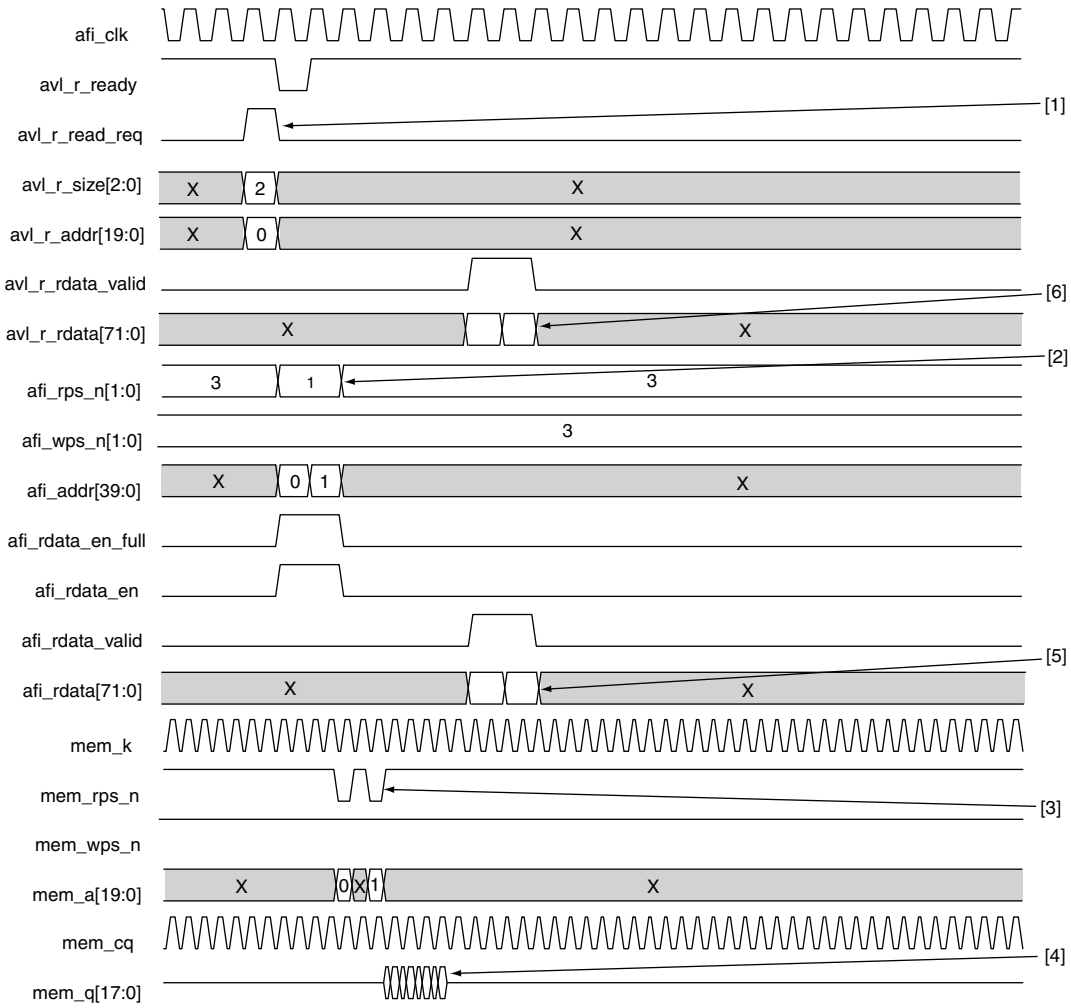
QDR II and QDR II+ Timing Diagrams

This section contains timing diagrams for QDR II and QDR II+ protocols.

Figure 12–9 through Figure 12–12 present the following timing diagrams, based on a Stratix III device:

- Half-Rate QDR II and QDR II+ SRAM Read
- Half-Rate QDR II and QDR II+ SRAM Write
- Full-Rate QDR II and QDR II+ SRAM Read
- Full-Rate QDR II and QDR II+ SRAM Write

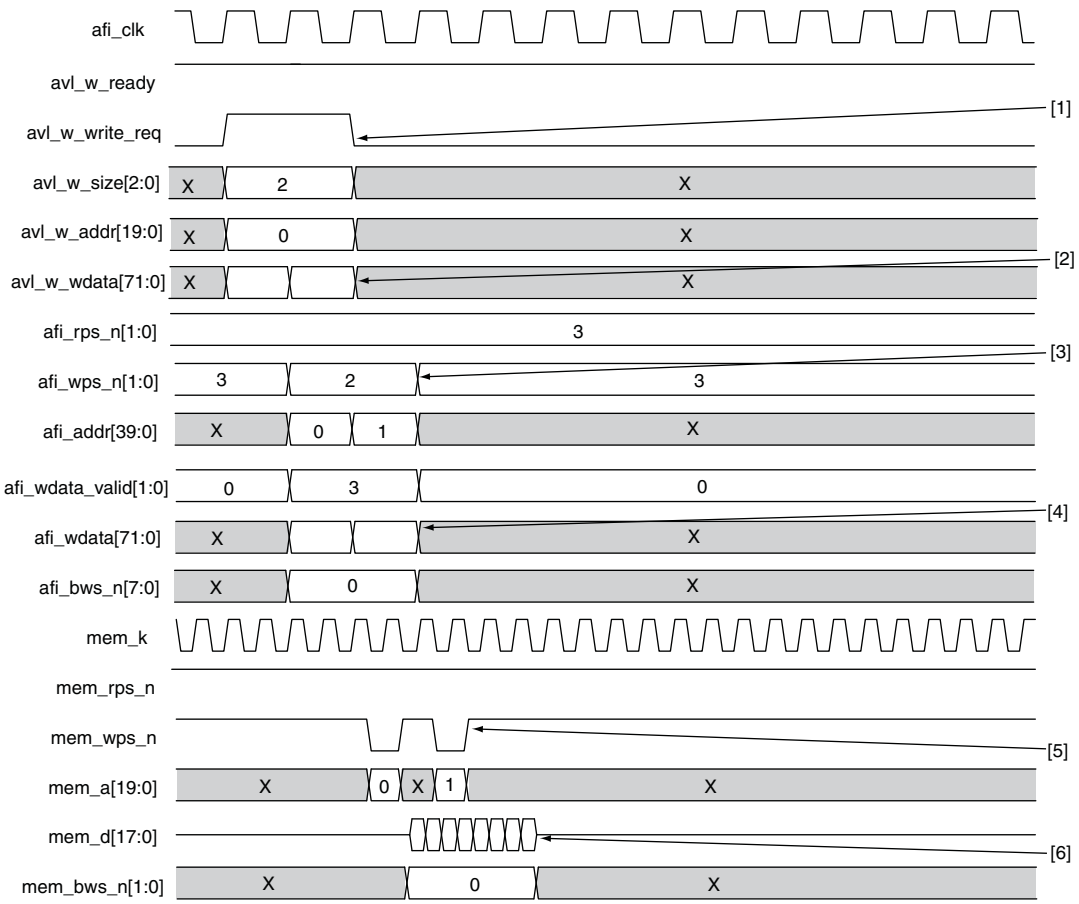
Figure 12–9. Half-Rate QDR II and QDR II+ SRAM Read



Notes for Figure 12–9:

(1) Controller receives read command.
(2) Controller issues two read commands to PHY.
(3) PHY issues two read commands to memory.
(4) PHY receives read data from memory.
(5) Controller receives read data from PHY.
(6) User logic receives read data from controller.

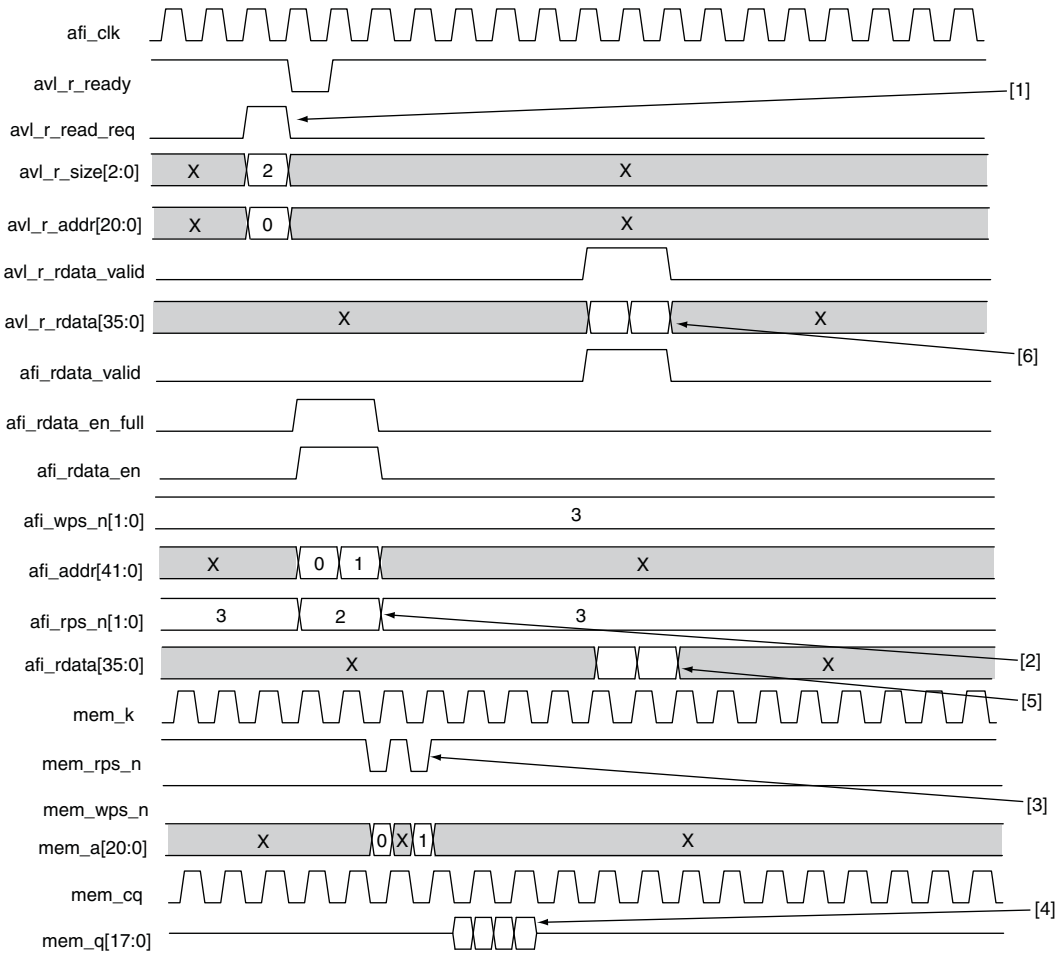
Figure 12–10. Half-Rate QDR II and QDR II+ SRAM Write



Notes for Figure 12–10:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues two write commands to PHY.
(4) Controller sends write data to PHY.
(5) PHY issues two write commands to memory.
(6) PHY sends write data to memory.

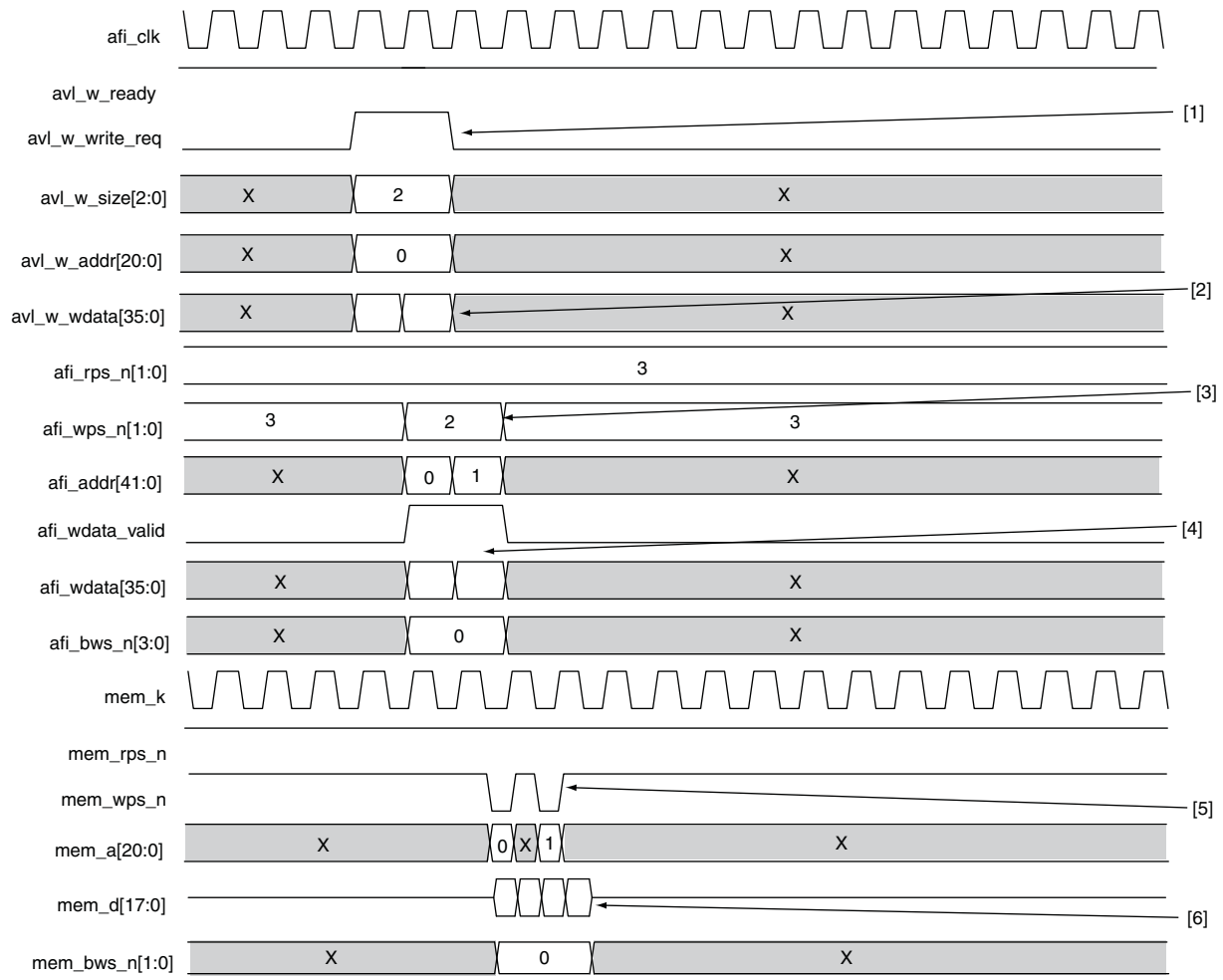
Figure 12–11. Full-Rate QDR II and QDR II+ SRAM Read



Notes for Figure 12–11:

(1) Controller receives read command.
(2) Controller issues two read commands to PHY.
(3) PHY issues two read commands to memory.
(4) PHY receives read data from memory.
(5) Controller receives read data from PHY.
(6) User logic receives read data from controller.

Figure 12–12. Full-Rate QDR II and QDR II+ SRAM Write



Notes for Figure 12–12:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues two write commands to PHY.
(4) Controller sends write data to PHY.
(5) PHY issues two write commands to memory.
(6) PHY sends write data to memory.

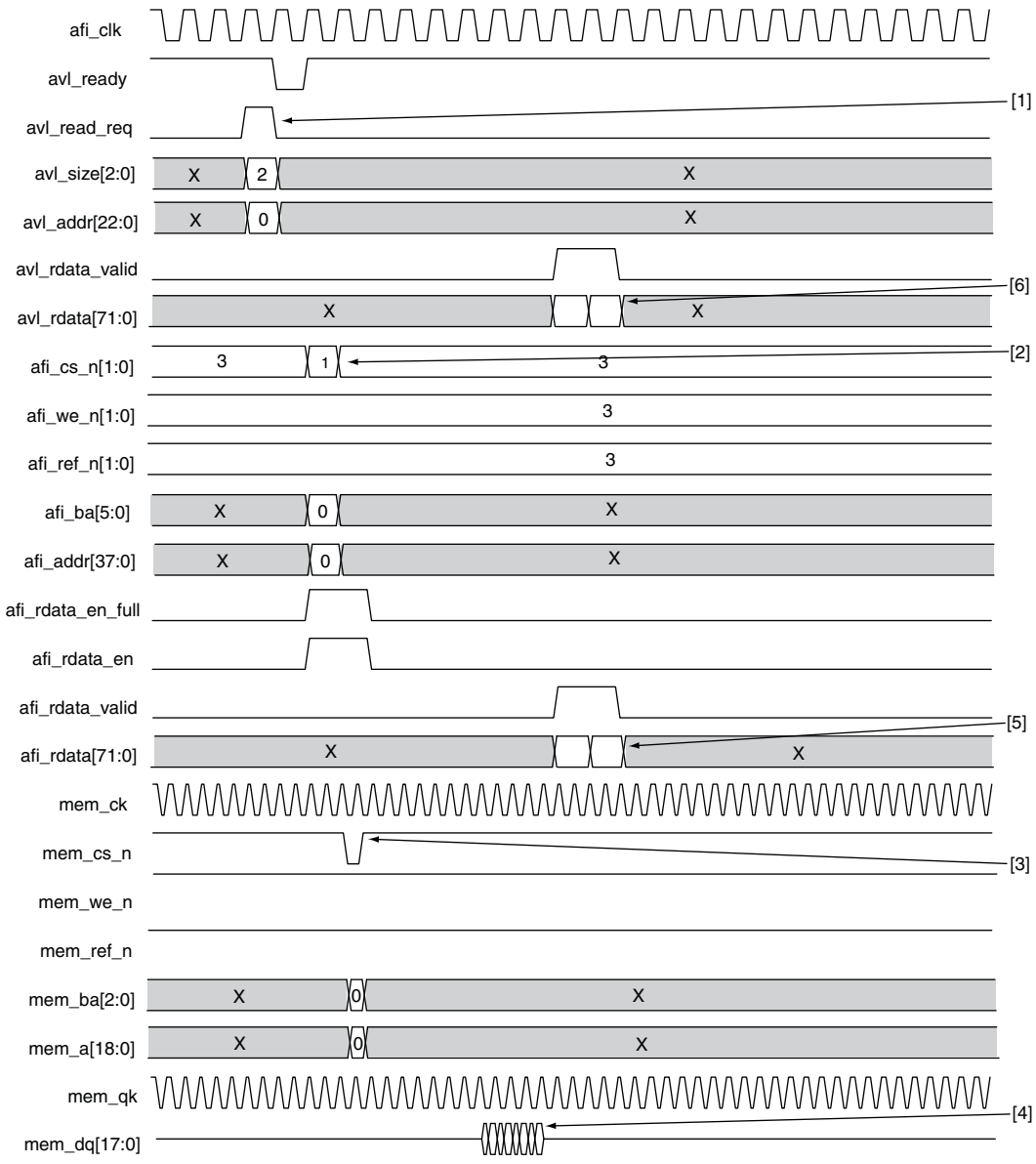
RLDRAM II Timing Diagrams

This section contains timing diagrams for RLDRAM protocols.

Figure 12–13 through Figure 12–16 present the following timing diagrams, based on a Stratix III device:

- Half-Rate RLDRAM II Read
- Half-Rate RLDRAM II Write
- Full-Rate RLDRAM II Read
- Full-Rate RLDRAM II Write

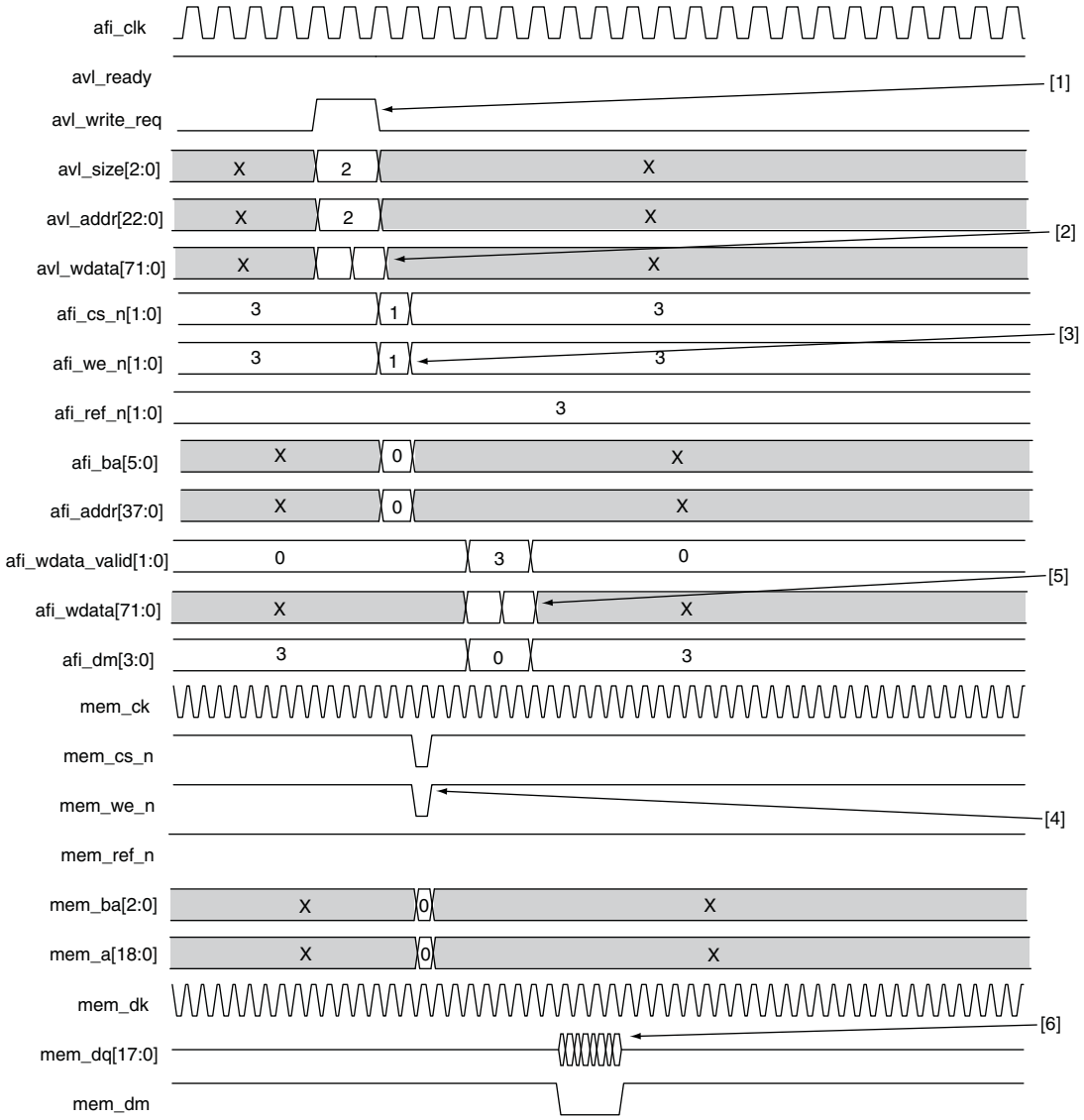
Figure 12–13. Half-Rate RLDRAM II Read



Notes for Figure 12–13:

(1) Controller receives read command.
(2) Controller issues read command to PHY.
(3) PHY issues read command to memory.
(4) PHY receives read data from memory.
(5) Controller receives read data from PHY.
(6) User logic receives read data from controller.

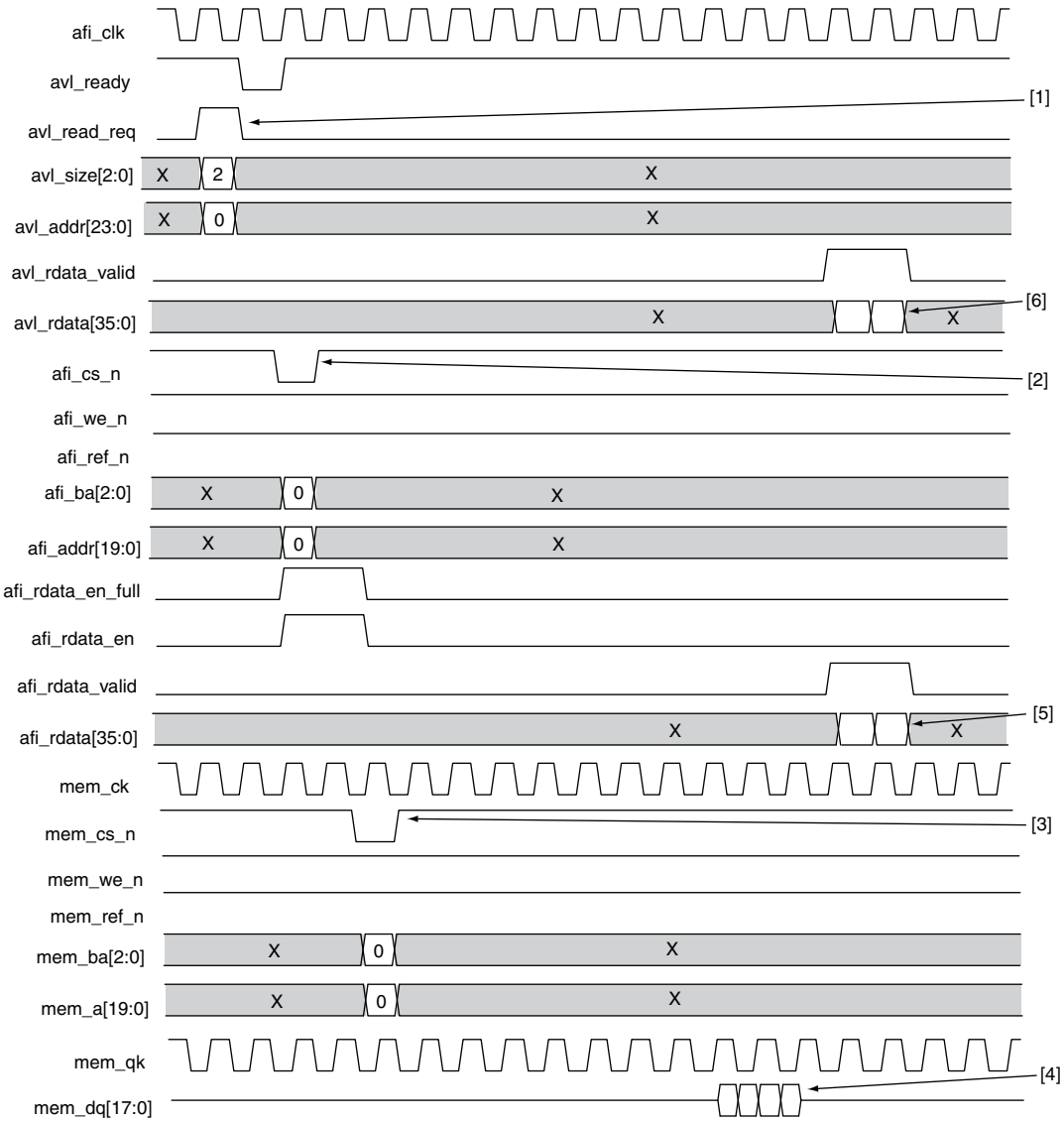
Figure 12–14. Half-Rate RLDRAM II Write



Notes for Figure 12–14:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues write command to PHY.
(4) PHY issues write command to memory.
(5) Controller sends write data to PHY.
(6) PHY sends write data to memory.

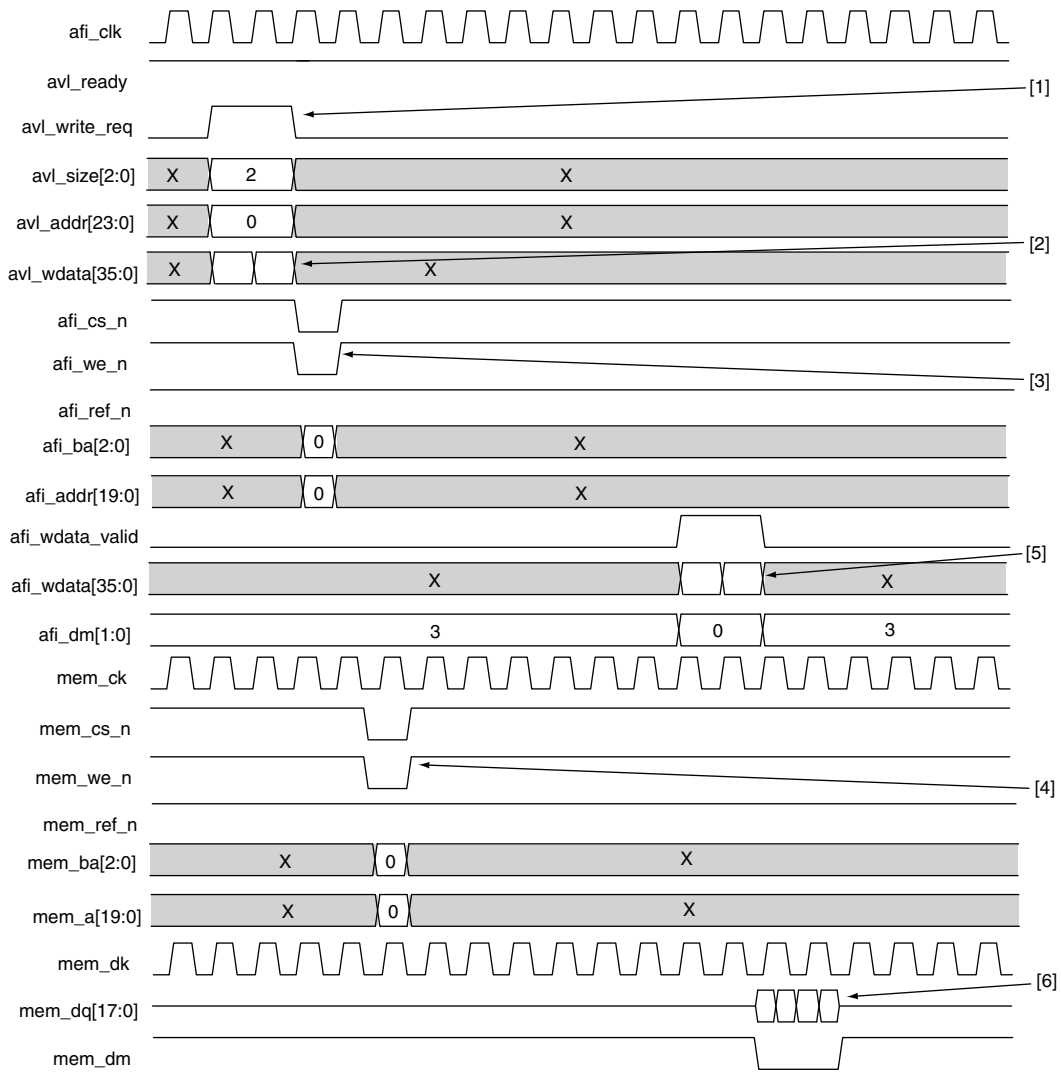
Figure 12–15. Full-Rate RLDRAM II Read



Notes for Figure 12–15:

(1) Controller receives read command.
(2) Controller issues read command to PHY.
(3) PHY issues read command to memory.
(4) PHY receives read data from memory.
(5) Controller receives read data from PHY.
(6) User logic receives read data from controller.

Figure 12–16. Full-Rate RLDRAM II Write



Notes for Figure 12–16:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues write command to PHY.
(4) PHY issues write command to memory.
(5) Controller sends write data to PHY.
(6) PHY sends write data to memory.

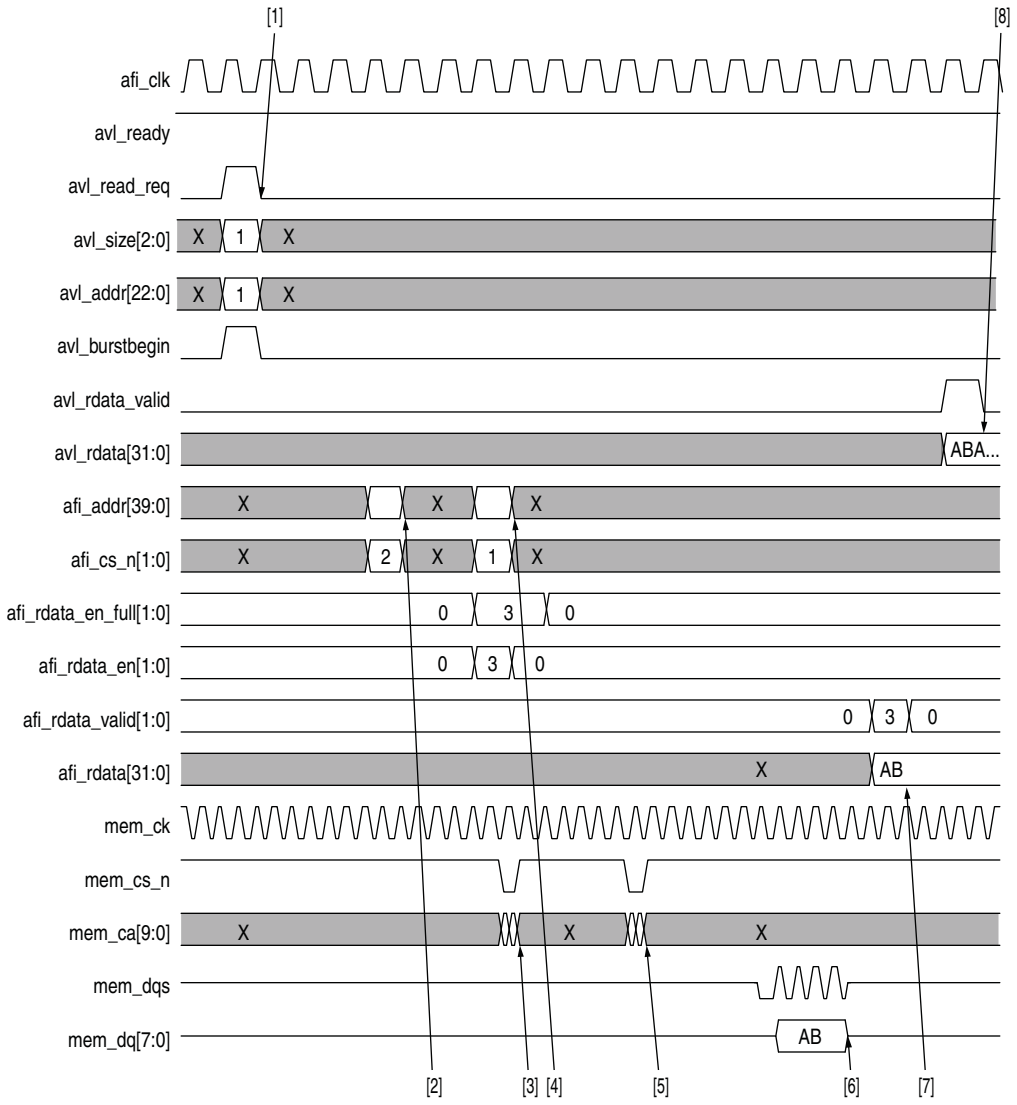
LPDDR2 Timing Diagrams

This section contains timing diagrams for LPDDR2 protocols.

Figure 12–17 through Figure 12–20 present the following timing diagrams:

- Half-Rate LPDDR2 Read
- Half-Rate LPDDR2 Write
- Full-Rate LPDDR2 Read
- Full-Rate LPDDR2 Write

Figure 12–17. Half-Rate LPDDR2 Read

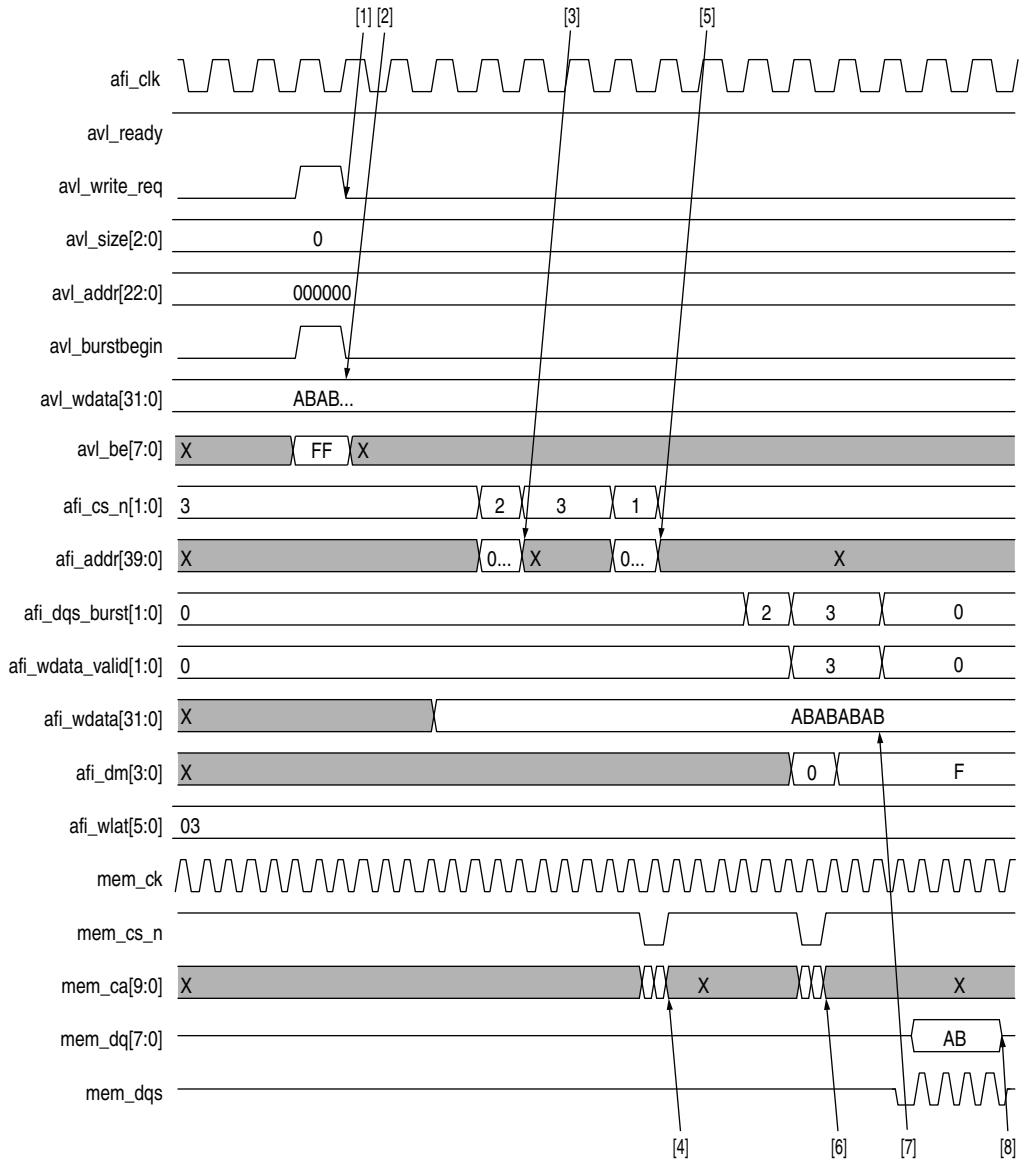


Notes for Figure 12–17:

(1) Controller receives read command.
(2) Controller issues activate command to PHY.
(3) PHY issues activate command to memory.

(4) Controller issues read command to PHY.
(5) PHY issues read command to memory.
(6) PHY receives read data from memory.
(7) Controller receives read data from PHY.
(8) User logic receives read data from controller.

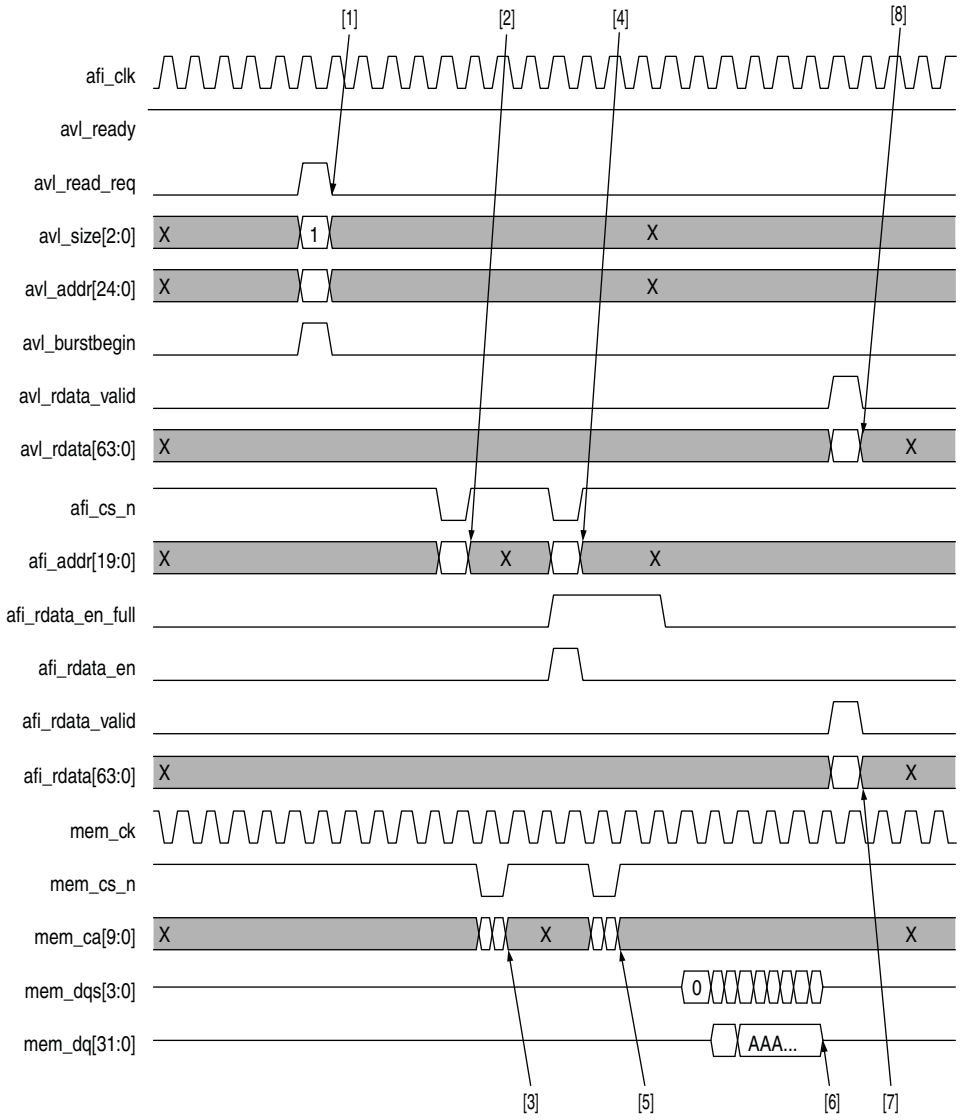
Figure 12–18. Half-Rate LPDDR2 Write



Notes for Figure 12–18:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues activate command to PHY.
(4) PHY issues activate command to memory.
(5) Controller issues write command to PHY.
(6) PHY issues write command to memory.
(7) Controller sends write data to PHY.
(8) PHY sends write data to memory.

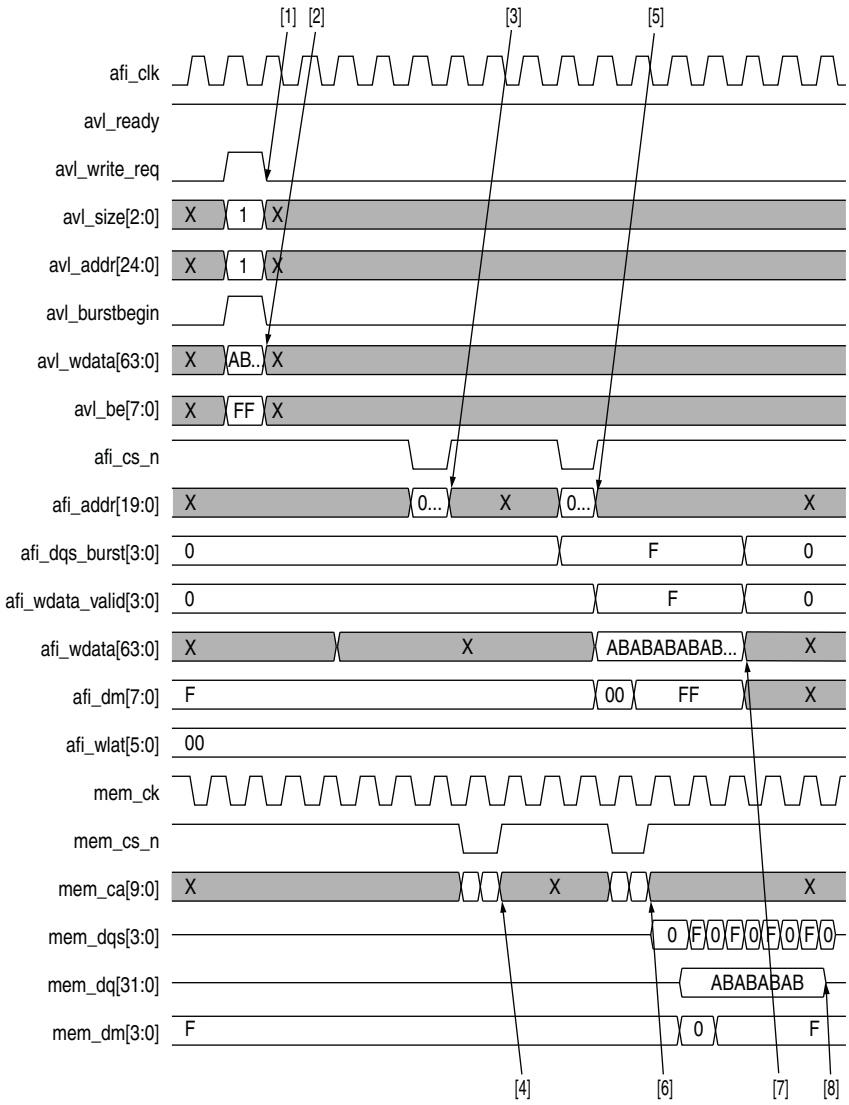
Figure 12–19. Full-Rate LPDDR2 Read



Notes for Figure 12–19:

(1) Controller receives read command.
(2) Controller issues activate command to PHY.
(3) PHY issues activate command to memory.
(4) Controller issues read command to PHY.
(5) PHY issues read command to memory.
(6) PHY receives read data from memory.
(7) Controller receives read data from PHY.
(8) User logic receives read data from controller.

Figure 12–20. Full-Rate LPDDR2 Write



Notes for Figure 12–20:

(1) Controller receives write command.
(2) Controller receives write data.
(3) Controller issues activate command to PHY.
(4) PHY issues activate command to memory.
(5) Controller issues write command to PHY.
(6) PHY issues write command to memory.
(7) Controller sends write data to PHY.
(8) PHY sends write data to memory.

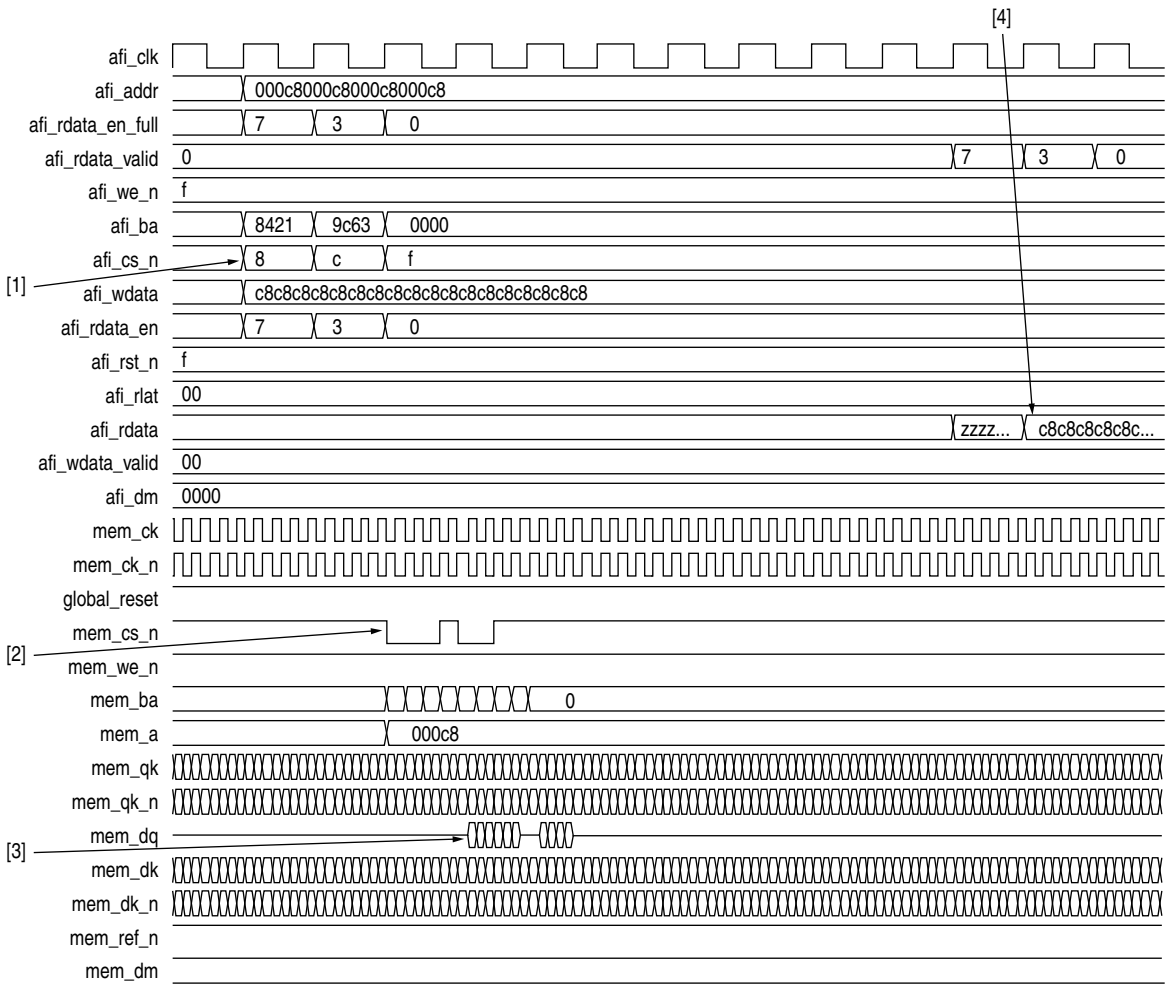
RLDRAM 3 Timing Diagrams

This section contains timing diagrams for RLDRAM 3 protocols.

Figure 12–21 through Figure 12–22 present the following timing diagrams:

- Quarter-Rate RLDRAM 3 Read
- Quarter-Rate RLDRAM 3 Write
- Half-rate RLDRAM 3 Read
- Half-Rate RLDRAM 3 Write

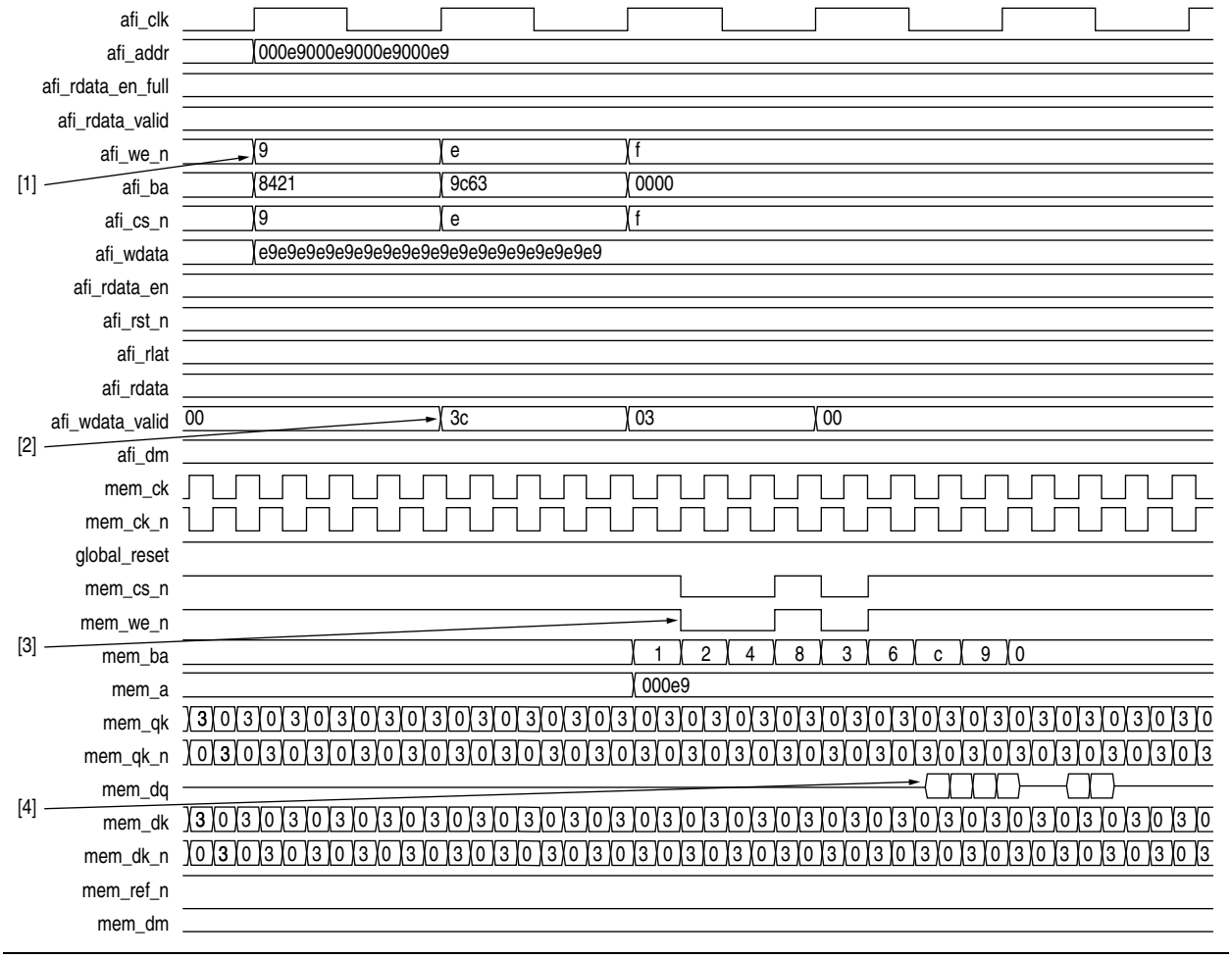
Figure 12–21. Quarter-Rate RLDRAM 3 Read



Notes for Figure 12–21:

(1) Controller issues read command to PHY.
(2) PHY issues read command to memory.
(3) PHY receives data from memory.
(4) Controller receives read data from PHY.

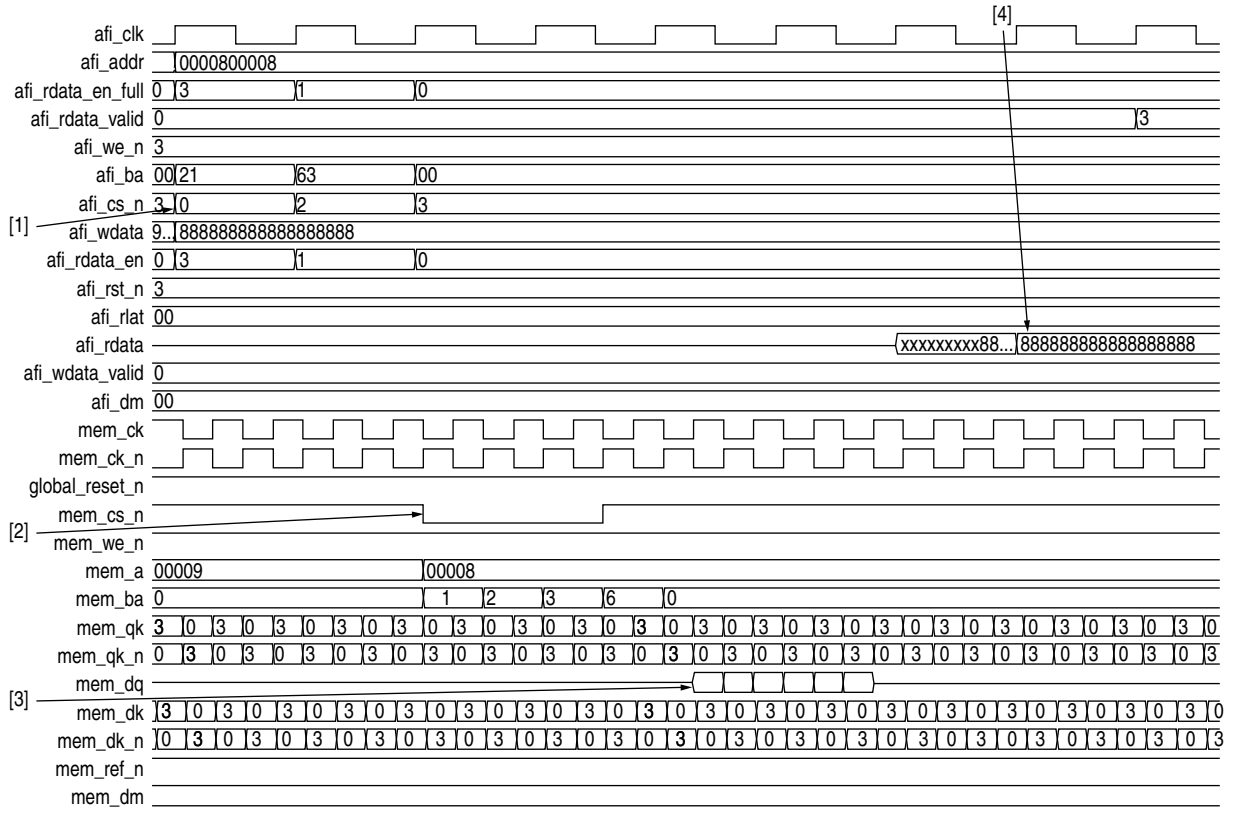
Figure 12–22. Quarter-Rate RLDRAM 3 Write



Notes for Figure 12–22:

(1) Controller issues write command to PHY.
(2) Data ready from controller for PHY.
(3) PHY issues write command to memory.
(4) PHY sends read data to memory.

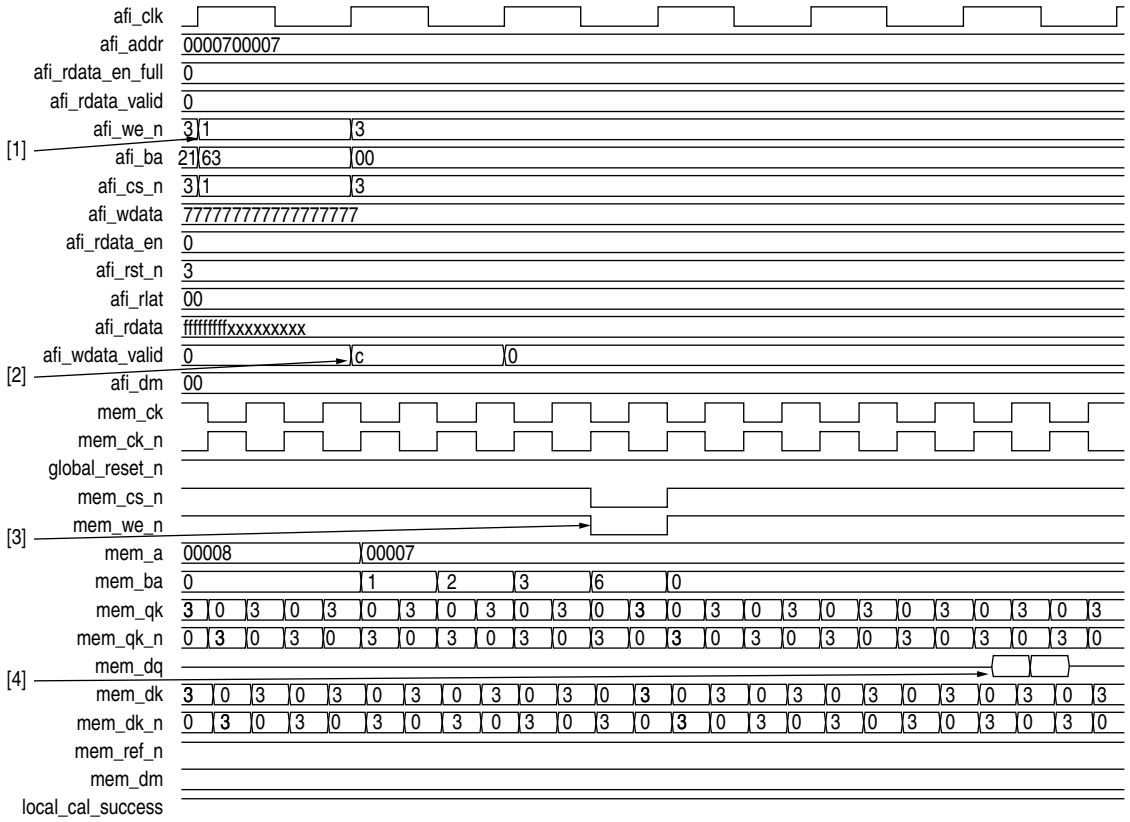
Figure 12–23. Half-rate RLDRAM 3 Read



Notes for Figure 12–23:

(1) Controller issues read command to PHY.
(2) PHY issues read command to memory.
(3) PHY receives data from memory.
(4) Controller receives read data from PHY.

Figure 12–24. Half-Rate RLDRAM 3 Write



Notes for Figure 12–24:

(1) Controller issues write command to PHY.
(2) Data ready from controller for PHY.
(3) PHY issues write command to memory.
(4) PHY sends read data to memory.

Document Revision History

Table 12–1 lists the revision history for this document.

Table 12–1. Document Revision History

Date	Version	Changes
November 2012	2.1	■ Added timing diagrams for RLDRAM 3. ■ Changed chapter number from 10 to 12.
June 2012	2.0	■ Added timing diagrams for LPDDR2. ■ Added Feedback icon.
November 2011	1.1	■ Consolidated timing diagrams from 11.0 **DDR2 and DDR3 SDRAM Controller with UniPHY User Guide**, **QDR II and QDR II+ SRAM Controller with UniPHY User Guide**, and **RLDRAM II Controller with UniPHY IP User Guide**. ■ Added Read and Write diagrams for DDR3 quarter-rate.

13. UniPHY External Memory Interface Debug Toolkit

EMI_RM_011-2.2

This chapter describes the UniPHY External Memory Interface Toolkit. It explains the architecture and workflow, as well as how to launch the toolkit, link your project to a device, and establish communications over a selected connection. The chapter also discussses several operational considerations to enhance your productivity with the toolkit.

Introduction

The EMIF Toolkit lets you diagnose and debug calibration problems and produce margining reports for your external memory interface. The toolkit is compatible with UniPHY-based external memory interfaces that use the Nios II-based sequencer, with toolkit communication enabled. Toolkit communication is on by default in versions 10.1 and 11.0 of UniPHY IP; for version 11.1 and later, toolkit communication is on whenever debugging is enabled on the Diagnostics tab of the IP core interface.

The EMIF Toolkit can communicate with several different memory interfaces on the same device, but can communicate with only one memory device at a time.

Architecture

The EMIF toolkit provides a graphical user interface for communication with connections, however all functions provided in the toolkit are also available directly from the `quartus_sh` TCL shell, through the `external_memif_toolkit` TCL package. The availablity of TCL support allows you to create scripts to run automatically from TCL. You can find information about specific TCL commands by running `help -pkg external_memif_toolkit` from the `quartus_sh` TCL shell.

If you want, you can begin interacting with the toolkit through the GUI, and later automate your workflow by creating TCL scripts. The toolkit GUI records a history of the commands that you run. You can see the command history on the History tab in the toolkit GUI.

Communication

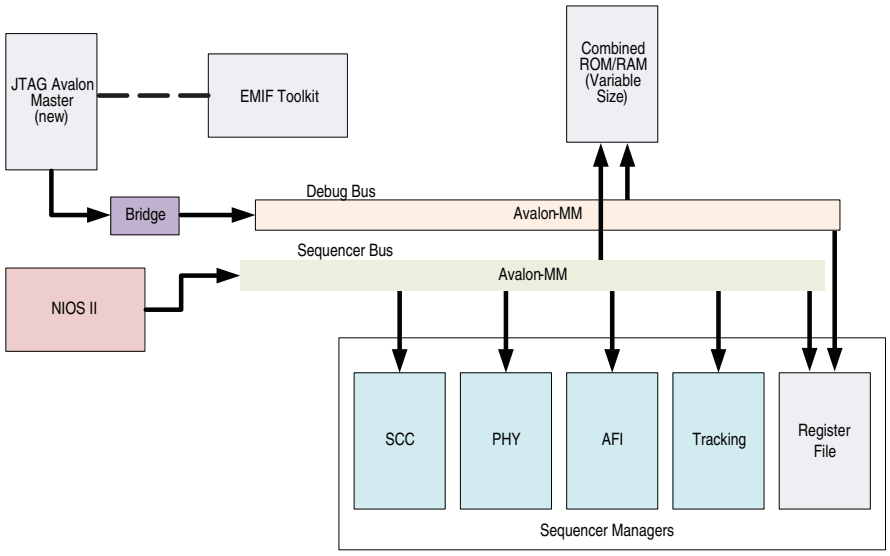
Communication between the EMIF Toolkit and external memory interface connections varies, depending on the connection type and version. In versions 10.1 and 11.0 of the EMIF IP, communication is achieved using direct communication to the Nios II-based sequencer. In version 11.1 and later, communication is achieved using a JTAG Avalon-MM master attached to the sequencer bus.

Figure 13–1 shows the structure of UniPHY-based IP version 11.1 and later, with JTAG Avalon-MM master attached to sequencer bus masters.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Figure 13–1. UniPHY IP Version 11.1 and Later, with JTAG Avalon-MM Master



Calibration and Report Generation

For versions 10.1 and 11.0 UniPHY-based interfaces, the EMIF Toolkit causes the memory interface to calibrate several times, to produce the data from which the toolkit generates its reports. In version 11.1 and later, report data is generated during calibration, without need to repeat calibration. For version 11.1 and later, generated reports reflect the result of the previous calibration, without need to recalibrate unless you choose to do so.

Setup and Use

Before using the EMIF Toolkit, you should compile your design and program the target device with the resulting SRAM Object File (**.sof**). For designs compiled in the Quartus II software version 12.0 or earlier, debugging information is contained in the JTAG Debugging Information file (**.jdi**); however, for designs compiled in the Quartus II software version 12.1 or later, all debugging information resides in the **.sof** file.

You can run the toolkit using all your project files, or using only the Quartus II Project File (**.qpf**), Quartus II Settings File (**.qsf**), and **.sof** file; the **.jdi** file is also required for designs compiled prior to version 12.1. To ensure that all debugging information is correctly synchronized for designs compiled prior to version 12.1, ensure that the **.sof** and **.jdi** files that you used are generated during the same run of the Quartus II Assembler.

After you have programmed the target device, you can run the EMIF Toolkit and open your project. You can then use the toolkit to create connections to the external memory interface.

General Workflow

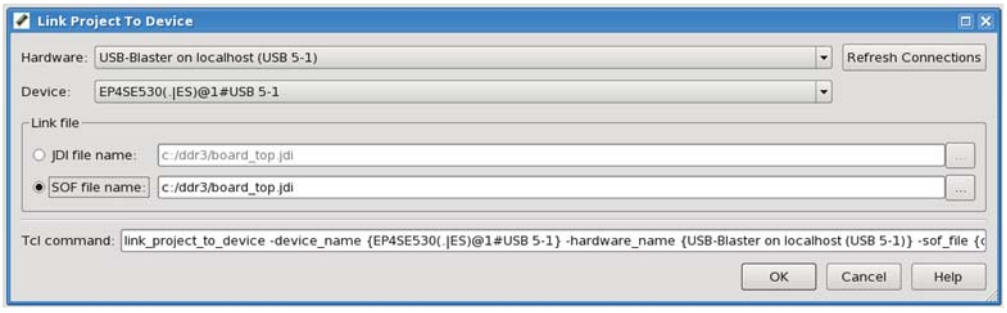
To use the EMIF Toolkit, you must link your compiled project to a device, and create a communication channel to the connection that you want to examine. This section explains how to perform these major steps.

Linking the Project to a Device

1. To launch the toolkit, select External Memory Interface Toolkit from the Tools menu in the Quartus II software.
2. After you have launched the toolkit, open your project and click the **Initialize connections** task in the **Tasks** window, to initialize a list of all known connections.
3. To link your project to a specific device on specific hardware, perform the following steps:
   a. Click the **Link Project to Device** task in the **Tasks** window.
   b. Select the desired hardware from the **Hardware** dropdown menu in the **Link Project to Device** dialog box.
   c. Select the desired device on the hardware from the **Device** dropdown menu in the **Link Project to Device** dialog box.
   d. Select the correct **Link file type**, depending on the version of Quartus II software in which your design was compiled:
      - If your design was compiled in the Quartus II software version 12.0 or earlier, select **JDI** as the **Link file type**, verify that the **.jdi** file is correct for your **.sof** file, and click **Ok**.
      - If your design was compiled in the Quartus II software version 12.1 or later, select **SOF** as the **Link file type**, verify that the **.sof** file is correct for your programmed device, and click **Ok.**

Figure 13–2 illustrates the **Link Project to Device** dialog box.

Figure 13–2. Link Project to Device Dialog Box



When you link your project to the device, the toolkit verifies all connections on the device against the information in the JDI or SOF file, as appropriate. If the toolkit detects any mismatch between the JDI file and the device connections, an error message is displayed.

For designs compiled using the Quartus II software version 12.1 or later, the SOF file contains a design hash to ensure the SOF file used to program the device matches the SOF file specified for linking to a project. If the hash does not match, an error message appears.

If the toolkit successfully verifies all connections, it then attempts to determine the connection type for each connection. Connections of a known type are listed in the Linked Connections report, and are available for the toolkit to use.

Establishing Communication to Connections

After you have completed linking the project, you can establish communication to the connections

1. In the Tasks window,
   - Click **Establish Memory Interface Connection** to create a connection to the external memory interface.
   - Click **Establish Efficiency Monitor Connection** to create a connection to the efficiency monitor.
2. To create a communication channel to a connection, select the desired connection from the displayed pulldown menu of connections, and click **Ok**.

   The toolkit establishes a communication channel to the connection, creates a report folder for the connection, and creates a folder of tasks for the connection.

   By default, the connection and the reports and tasks folders are named according to the hierarchy path of the connection. If you want, you can specify a different name for the connection and its folders.
3. You can run any of the tasks in the folder for the connection; any resulting reports appear in the reports folder for the connection.

Reports

The toolkit can generate a variety of reports, including summary, calibration, and margining reports for external memory interface connections. To generate a supported type of report for a connection, you run the associated task in the tasks folder for that connection.

Summary Report

The Summary Report provides an overview of the memory interface; it consists of the following tables:

- Summary table. Provides a high-level summary of calibration results. This table lists details about the connection, IP version, IP protocol, and basic calibration results, including calibration failures. This table also lists the estimated average read and write data valid windows, and the calibrated read and write latencies.
- Interface Details table. Provides details about the parameterization of the memory IP. This table allows you to verify that the parameters in use match the actual memory device in use.

- Groups Masked from Calibration table. Lists any groups that were masked from calibration when calibration occurred. Masked groups are ignored during calibration.
- Ranks Masked from Calibration tables (DDR2 and DDR3 only). Lists any ranks that were masked from calibration when calibration occurred. Masked ranks are ignored during calibration.

Calibration Report

The Calibration Report provides detailed information about the margins observed before and after calibration, and the settings applied to the memory interface during calibration; it consists of the following tables:

- Per DQS Group Calibration table. Lists calibration results for each group. If a group fails calibration, this table also lists the reason for the failure.

  ☞ If a group fails calibration, the calibration routine skips all remaining groups. You can deactivate this behaviour by running the **Enable Calibration for All Groups On Failure** command in the toolkit.

- DQ Pin Margins Observed Before Calibration table. Lists the DQ pin margins observed before calibration occurs. You can refer to this table to see the per-bit skews resulting from the specific silicon and board that you are using.
- DQS Group Margins Observed During Calibration table. Lists the DQS group margins observed during calibration.
- DQ Pin Settings After Calibration and DQS Group Settings After Calibration table. Lists the settings made to all dynamically controllable parts of the memory interface as a result of calibration. You can refer to this table to see the modifications made by the calibration algorithm.

Margin Report

The Margin Report lists the post-calibration margins for each DQ and data mask pin, keeping all other pin settings constant; it consists of the following tables:

- DQ Pin Post Calibration Margins table. Lists the margin data in tabular format.
- Read Data Valid Windows report. Shows read data valid windows in graphical format.
- Write Data Valid Windows report. Shows write data valid windows in graphical format.

Operational Considerations

This section provides additional information about specific features and considerations.

Specifying a Particular JDI File

Correct operation of the EMIF Toolkit depends on the correct JDI being used when linking the project to the device. The JDI file is produced by the Quartus II Assembler, and contains a list of all system level debug nodes and their heirarchy path names. If the default **.jdi** file name is incorrect for your project, you must specify the correct **.jdi** file. The **.jdi** file is supplied during the link-project-to-device step, where the ***revision_name*.jdi** file in the project directory is used by default. To supply an alternative **.jdi** file, click on the ellipse then select the correct **.jdi** file.

PLL Status

When connecting to DDR-based external memory interface connections, the PLL status appears in the Establish Connection dialog box when the IP is generated to use the CSR controller port, allowing you to immediately see whether the PLL status is locked. If the PLL is not locked, no communication can occur until the PLL becomes locked and the memory interface reset is deasserted.

When you are linking your project to a device, an error message will occur if the toolkit detects that a JTAG Avalon-MM master has no clock running. You can run the **Reindex Connections** task to have the toolkit rescan for connections and update the status and type of found connections in the Linked Connections report.

Margining Reports

The EMIF Toolkit can display margining information showing the post-calibration data-valid windows for reads and writes. Margining information is determined by individually modifying the input and output delay chains for each data and strobe/clock pin to determine the working region. The toolkit can display margining data in both tabular and hierarchial formats.

Group Masks

To aid in debugging your external memory interface, the EMIF Toolkit allows you to mask individual groups and ranks from calibration. Masked groups and ranks are skipped during the calibration process, meaning that only unmasked groups and ranks are included in calibration. Subsequent mask operations overwrite any previous masks.

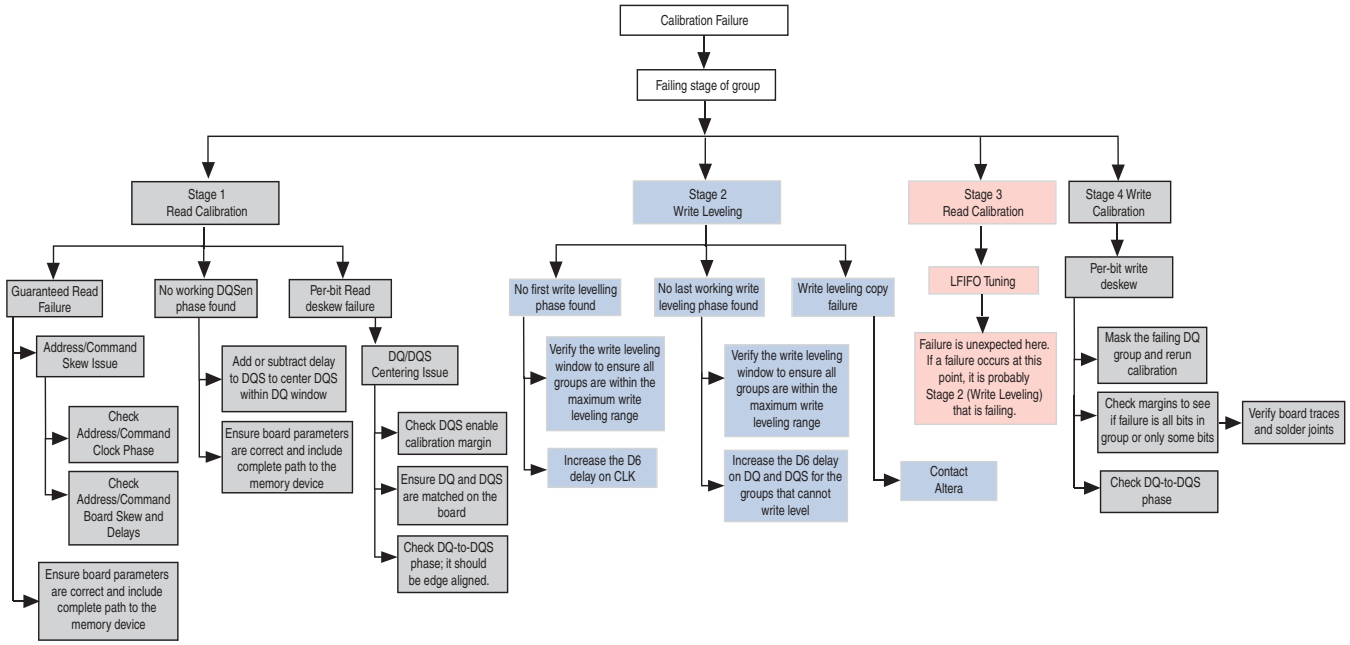
For information about calibration stages, refer to UniPHY Calibration Stages in section 1 of this volume.

Troubleshooting

In the event of calibration failure, refer to Figure 13–3 to assist in troubleshooting your design. Calibration results and failing stages are available through the external memory interface toolkit.

Figure 13–3 shows the recommended flow for troubleshooting calibration failure.

Figure 13–3. Debugging Tips



EMIF On-Chip Debug Toolkit

The EMIF On-Chip Debug Toolkit allows user logic to access the same calibration data used by the EMIF Toolkit, and allows user logic to send commands to the sequencer.

Introduction

You can use the EMIF On-Chip Debug Toolkit to access calibration data for your design and to send commands to the sequencer just as the EMIF Toolkit would. The following information is available:

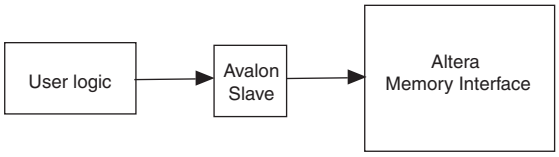
- Pass/fail status for each DQS group
- Read and write data valid windows for each group

In addition, user logic can request the following commands from the sequencer:

- Destructive recalibration of all groups
- Masking of groups and ranks
- Generation of per-DQ pin margining data as part of calibration

The user logic communicates through an Avalon-MM slave interface as shown in Figure 13–4.

Figure 13–4. User Logic Access



Access Protocol

The EMIF On-Chip Debug Toolkit provides access to calibration data through an Avalon-MM slave interface. To send a command to the sequencer, user logic sends a command code to the command space in sequencer memory. The sequencer polls the command space for new commands after each group completes calibration, and continuously after overall calibration has completed.

The communication protocol to send commands from user logic to the sequencer uses a multistep handshake with a data structure as shown below, and an algorithm as shown in Figure 13–5.

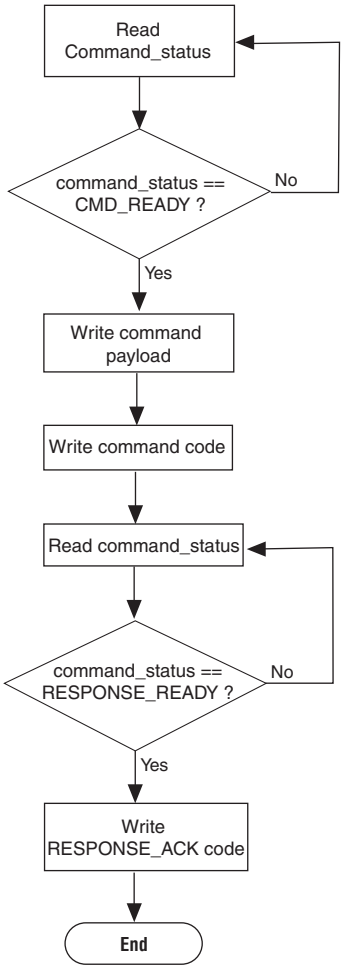
```
typedef struct_debug_data_struct {
    ...
    // Command interaction
    alt_u32 requested_command;
    alt_u32 command_status;
    alt_u32 command_parameters[COMMAND_PARAM_WORDS];
    ...
}
```

To send a command to the sequencer, user logic must first poll the `command_status` word for a value of `TCLDBG_TX_STATUS_CMD_READY`, which indicates that the sequencer is ready to accept commands. When the sequencer is ready to accept commands, user logic must write the command parameters into `command_parameters`, and then write the command code into `requested_command`.

The sequencer detects the command code and replaces `command_status` with `TCLDBG_TX_STATUS_CMD_EXE`, to indicate that it is processing the command. When the sequencer has finished running the command, it sets `command_status` to `TCLDBG_TX_STATUS_RESPONSE_READY` to indicate that the result of the command is available to be read. (If the sequencer rejects the requested command as illegal, it sets `command_status` to `TCLDBG_TX_STATUS_ILLEGAL_CMD`.)

User logic acknowledges completion of the command by writing `TCLDBG_CMD_RESPONSE_ACK` to `requested_command`. The sequencer responds by setting `command_status` back to `STATUS_CMD_READY`. (If an illegal command is received, it must be cleared using `CMD_RESPONSE_ACK`.)

Figure 13–5. Debugging Algorithm Flowchart



Command Codes Reference

Table 13–1 lists the supported command codes for the On-Chip Debug Toolkit.

Table 13–1. Supported Command Codes

Command	Parameters	Description
TCLDBG_RUN_MEM_CALIBRATE	None	Runs the calibration routine.
TCLDBG_MARK_ALL_DQS_GROUPS_AS_VALID	None	Marks all groups as valid for calibration.
TCLDBG_MARK_GROUP_AS_SKIP	Group to skip	Mark the specified group to be skipped by calibration.
TCLDBG_MARK_ALL_RANKS_AS_VALID	None	Mark all ranks as valid for calibration
TCLDBG_MARK_RANK_AS_SKIP	Rank to skip	Mark the specified rank to be skipped by calibration.
TCLDBG_ENABLE_MARGIN_REPORT	None	Enables generation of the margin report.

Header Files

The UniPHY-based external memory interface IP generates header files which identify the debug data structures and memory locations used with the EMIF On-Chip Debug Toolkit. You should refer to these header files for information required for use with your core user logic. It is highly recommended to use a software component (such as a Nios II processor) to access the calibration debug data.

The header files are unique to your IP parameterization and version, therefore you must ensure that you are referring to the correct version of header for your design. The names of the header files are: **core_debug.h** and **core_debug_defines.h**.

Generating UniPHY IP With the Debug Port

The following steps summarize the procedure for implementing your IP with the EMIF On-Chip Debug Toolkit enabled:

1. Add the variable `alt_mem_if_enable_core_debug_data=1` to the **quartus.ini** file.
2. Start the Quartus II software and generate a new DDR3 external memory interface with UniPHY.
3. On the **Diagnostics** tab of the parameter editor, turn on **Enable core sequencer debug access**.
4. Ensure that the **Core sequencer debug interface type** is set to **Avalon-MM Slave**.
5. Click **Finish** to generate your IP.
6. Find the Avalon interface in the top-level generated file. Connect this interface to your debug component.

```
input  wire [19:0] seq_debug_addr,        //    seq_debug.address
input  wire        seq_debug_read_req,    //             .read
output wire [31:0] seq_debug_rdata,       //             .readdata
input  wire        seq_debug_write_req,   //             .write
input  wire [31:0] seq_debug_wdata,       //             .writedata
output wire        seq_debug_waitrequest, //             .waitrequest
input  wire [3:0]  seq_debug_be,          //             .byteenable
output wire        seq_debug_rdata_valid  //             .readdatavalid
```

7. Find the **core_debug.h** and **core_debug_defines.h** header files in *<design_name>*/*<design_name>*_s0_software/sequencer and include these files in your debug component code.
8. Write your debug component using the supported command codes, to read and write to the Avalon-MM interface.

The debug data structure resides at the memory address SEQ_CORE_DEBUG_BASE, which is defined in the **core_debug_defines.h** header file.

Example C Code for Accessing Debug Data

A typical use of the EMIF On-Chip Debug Toolkit might be to recalibrate the external memory interface, and then access the reports directly using the summary_report_ptr, cal_report_ptr, and margin_report_ptr pointers, which are part of the debug data structure.

The following code sample illustrates:

```
/*
 * DDR3 UniPHY sequencer core access example
 */

#include <stdio.h>
#include <unistd.h>
#include <io.h>
#include "core_debug_defines.h"

int send_command(volatile debug_data_t* debug_data_ptr, int command, int
args[], int num_args)
{
       volatile int i, response;

       // Wait until command_status is ready
       do {
         response = IORD_32DIRECT(&(debug_data_ptr->command_status), 0);
       } while(response != TCLDBG_TX_STATUS_CMD_READY);

       // Load arguments
       if(num_args > COMMAND_PARAM_WORDS)
       {
```

```
                // Too many arguments
                return 0;
        }
        for(i = 0; i < num_args; i++)
        {
                IOWR_32DIRECT(&(debug_data_ptr->command_parameters[i]), 0,
args[i]);
        }

        // Send command code
        IOWR_32DIRECT(&(debug_data_ptr->requested_command), 0, command);

        // Wait for acknowledgment
        do {
                  response = IORD_32DIRECT(&(debug_data_ptr->
command_status), 0);
        } while(response != TCLDBG_TX_STATUS_RESPOSE_READY && response !=
TCLDBG_TX_STATUS_ILLEGAL_CMD);

        // Acknowledge response
        IOWR_32DIRECT(&(debug_data_ptr->requested_command), 0,
TCLDBG_CMD_RESPONSE_ACK);

        // Return 1 on success, 0 on illegal command
        return (response != TCLDBG_TX_STATUS_ILLEGAL_CMD);
}

int main()
{
          volatile debug_data_t* my_debug_data_ptr;
          volatile debug_summary_report_t* my_summary_report_ptr;
          volatile debug_cal_report_t* my_cal_report_ptr;
          volatile debug_margin_report_t* my_margin_report_ptr;

          volatile debug_cal_observed_dq_margins_t*
cal_observed_dq_margins_ptr;

          int i, j, size;
          int args[COMMAND_PARAM_WORDS];

          // Initialize pointers to the debug reports
          my_debug_data_ptr = (debug_data_t*)SEQ_CORE_DEBUG_BASE;
          my_summary_report_ptr =
(debug_summary_report_t*)(IORD_32DIRECT(&(my_debug_data_ptr->
summary_report_ptr), 0));
          my_cal_report_ptr =
(debug_cal_report_t*)(IORD_32DIRECT(&(my_debug_data_ptr->
cal_report_ptr), 0));
```

```
        my_margin_report_ptr =
(debug_margin_report_t*)(IORD_32DIRECT(&(my_debug_data_ptr->
margin_report_ptr), 0));

        // Activate all groups and ranks
        send_command(my_debug_data_ptr,
TCLDBG_MARK_ALL_DQS_GROUPS_AS_VALID, 0, 0);
        send_command(my_debug_data_ptr, TCLDBG_MARK_ALL_RANKS_AS_VALID,
0, 0);
        send_command(my_debug_data_ptr, TCLDBG_ENABLE_MARGIN_REPORT, 0,
0);

        // Mask group 4
        args[0] = 4;
        send_command(my_debug_data_ptr, TCLDBG_MARK_GROUP_AS_SKIP,
args, 1);

        send_command(my_debug_data_ptr, TCLDBG_RUN_MEM_CALIBRATE, 0, 0);

        // SUMMARY
        printf("SUMMARY REPORT\n");
        printf("mem_address_width: %u\n",
IORD_32DIRECT(&(my_summary_report_ptr->mem_address_width), 0));
        printf("mem_bank_width: %u\n",
IORD_32DIRECT(&(my_summary_report_ptr->mem_bank_width), 0));
        // etc...

        // CAL REPORT
        printf("CALIBRATION REPORT\n");
        // DQ read margins
        for(i = 0; i < RW_MGR_MEM_DATA_WIDTH; i++)
        {
              cal_observed_dq_margins_ptr = &(my_cal_report_ptr->
cal_dq_in_margins[i]);

              printf("0x%x DQ %d Read Margin (taps): -%d : %d\n",
(unsigned int)cal_observed_dq_margins_ptr, i,
                    IORD_32DIRECT(&(cal_observed_dq_margins_ptr->
left_edge), 0),
                    IORD_32DIRECT(&(cal_observed_dq_margins_ptr->
right_edge), 0));
        }
        // etc...

        return 0;
}
```

Document Revision History

Table 13–2 lists the revision history for this document.

Table 13–2. Document Revision History

Date	Version	Changes
November 2012	2.2	■ Changes to Setup and Use and General Workflow sections. ■ Added EMIF On-Chip Debug Toolkit section ■ Changed chapter number from 11 to 13.
August 2012	2.1	■ Added table of debugging tips.
June 2012	2.0	■ Revised content for new UnIPHY EMIF Toolkit. ■ Added Feedback icon.
November 2011	1.0	Harvested 11.0 **DDR2 and DDR3 SDRAM Controller with UniPHY EMIF Toolkit** content.

14. Upgrading to UniPHY-based Controllers from ALTMEMPHY-based Controllers

EMI_RM_012-2.3

This chapter describes upgrading the following ALTMEMPHY-based controller designs to UniPHY-based controllers:

- DDR2 or DDR3 SDRAM High-Performance Controller II with ALTMEMPHY designs
- DDR2 or DDR3 SDRAM High-Performance Controller with ALTMEMPHY designs

Altera does not support upgrading designs that do not use the AFI.

If your design uses non-AFI IP cores, Altera recommends that you start a new design with the UniPHY IP core . In addition, Altera recommends that any new designs targeting Stratix III, Stratix IV, or Stratix V use the UniPHY datapath.

Upgrading from DDR2 or DDR3 SDRAM High-Performance Controller II with ALTMEMPHY Designs

To upgrade to the DDR2 or DDR3 SDRAM controller with UniPHY IP core from DDR2 or DDR3 SDRAM High-Performance Controller II with ALTMEMPHY designs, follow these steps:

1. Generating Equivalent Design
2. Replacing the ALTMEMPHY Datapath with UniPHY Datapath
3. Resolving Port Name Differences
4. Creating OCT Signals
5. Running Pin Assignments Script
6. Removing Obsolete Files
7. Simulating your Design

The following sections describes these steps.

Generating Equivalent Design

Create a new DDR2 or DDR3 SDRAM controller with UniPHY IP core, by following the steps in volume 2, section 1, chapter 8, *Implementing and Parameterizing Memory IP* and use the following guidelines:

- Specify the same variation name as the ALTMEMPHY variation.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

- Specify a directory different than the ALTMEMPHY design directory to prevent files from overwriting each other during generation.

To ease the migration process, ensure the UniPHY-based design you create is as similar as possible to the existing ALTMEMPHY-based design. In particular, you should ensure the following settings are the same in your UniPHY-based design:

- **PHY settings** tab
  - FPGA speed grade
  - PLL reference clock
  - Memory clock frequency
  - There is no need to change the default **Address and command clock phase settings**; however, if you have board skew effects in your ALTMEMPHY design, enter the difference between that clock phase and the default clock phase into the **Address and command clock phase settings**.
- **Memory Parameters** tab—all parameters must match.
- **Memory Timing** tab—all parameters must match.
- **Board settings** tab—all parameters must match.
- **Controller settings** tab—all parameters must match

In ALTMEMPHY-based designs you can turn off dynamic OCT. However, all UniPHY-based designs use dynamic parallel OCT and you cannot turn it off.

Replacing the ALTMEMPHY Datapath with UniPHY Datapath

To replace the ALTMEMPHY datapath with the UniPHY datapath, follow these steps:

1. In the Quartus II software, open the Assignment Editor, on the Assignments menu click **Assignment Editor**.
2. Manually, delete all of the assignments related to the external memory interface pins, except for the location assignments if you are preserving the pinout. By default, these pin names start with the `mem` prefix, though in your design they may have a different name.
3. Remove the old ALTMEMPHY **.qip** file from the project:
   - On the Assignments menu click **Settings**.
   - Specify the old **.qip**, and click **Remove**.

Your design now uses the UniPHY datapath.

Resolving Port Name Differences

Several port names in the ALTMEMPHY datapath are different than in the UniPHY datapath. The different names may cause compilation errors. This section describes the changes you must make in the RTL for the entity that instantiates the memory IP core. Each change applies to a specific port in the ALTMEMPHY datapath. Unconnected ports require no changes.

In some instances, multiple ports in ALTMEMPHY-based designs are mapped to a single port in UniPHY-based designs. If you use both ports in ALTMEMPHY-based designs, assign a temporary signal to the common port and connect it to the original wires. Table 14–1 shows the changes you must make.

Table 14–1. Changes to ALTMEMPHY Port Names

ALTMEMPHY Port	Changes
aux_full_rate_clk	The UniPHY-based design does not generate this signal. You can generate it if you require it.
aux_scan_clk	The UniPHY-based design does not generate this signal. You can generate it if you require it.
aux_scan_clk_reset_n	The UniPHY-based design does not generate this signal. You can generate it if you require it.
dll_reference_clk	The UniPHY-based design does not generate this signal. You can generate it if you require it.
dqs_delay_ctrl_export	This signal is for DLL sharing between ALTMEMPHY instances and is not applicable for UniPHY-based designs.
local_address	Rename to avl_addr.
local_be	Rename to avl_be.
local_burstbegin	Rename to avl_burstbegin.
local_rdata	Rename to avl_rdata.
local_rdata_valid	Rename to avl_rdata_valid.
local_read_req	Rename to avl_read_req.
local_ready	Rename to avl_ready.
local_size	Rename to avl_size.
local_wdata	Rename to avl_wdata.
local_write_req	Rename to avl_write_req.
mem_addr	Rename to mem_a.
mem_clk	Rename to mem_ck.
mem_clk_n	Rename to mem_ck_n.
mem_dqsn	Rename to mem_dqs_n.
oct_ctl_rs_value	Remove from design ("Creating OCT Signals" on page 14–4).
oct_ctl_rt_value	Remove from design ("Creating OCT Signals" on page 14–4).
phy_clk	Rename to afi_clk.
reset_phy_clk_n	Rename to afi_reset_n.
local_refresh_ack local_wdata_req reset_request_n	The controller no longer exposes these signals to the top-level design, so comment out these outputs. If you need it, bring the wire out from the high-performance II controller entity in <*project directory*>**/uniphy/rtl/**<*variation name*>**_controller_phy.sv**.

Creating OCT Signals

In ALTMEMPHY-based designs, the Quartus II Fitter creates the `alt_oct` block outside the IP core and connects it to the `oct_ctl_rs_value` and `oct_ctl_rt_value` signals. In UniPHY-based designs, the OCT block is part of the IP core, so the design no longer requires these two ports. Instead, the UniPHY-based design requires two additional ports, `oct_rup` and `oct_rdn`. You must create these ports in the instantiating entity as input pins and connect to the UniPHY instance. Then route these pins to the top-level design and connect to the OCT R_{UP} and R_{DOWN} resistors on the board.

For information on OCT control block sharing, refer to "The OCT Sharing Interface" in this volume.

Running Pin Assignments Script

Remap your design by running analysis and synthesis. When analysis and synthesis completes, run the pin assignments Tcl script and then verify the new pin assignments in the Assignment Editor.

Removing Obsolete Files

After you upgrade the design, you may remove the unnecessary ALTMEMPHY design files from your project. To identify these files, examine the original ALTMEMPHY-generated **.qip** file in any text editor.

Simulating your Design

You must use the UniPHY memory model to simulate your new design. To use the UniPHY memory model, follow these steps:

1. Edit your instantiation of the UniPHY datapath to ensure the `local_init_done`, `local_cal_success`, `local_cal_fail`, `soft_reset_n`, `oct_rdn`, `oct_rup`, `reset_phy_clk_n`, and `phy_clk` signals are at the top-level entity so that an instantiating testbench can refer to those signals.
2. To use the UniPHY testbench and memory model, generate the example design when generating your IP instantiation.

3. Specify that your third-party simulator should use the UniPHY testbench and memory model instead of the ALTMEMPHY memory model:
   a. On the Assignments menu, point to **Settings** and click the **Project Settings** window.
   b. Select the **Simulation** tab, click **Test Benches**, click **Edit**, and replace the ALTMEMPHY testbench files with the following files:
   - ****<*project directory*>****<*variation name*>**_example_design\simulation\verilog\submodules\altera_avalon_clock_source.sv**
     or
     ****<*project directory*>****<*variation name*>**_example_design\simulation\vhdl\submodules\altera_avalon_clock_source.vhd**
   - ****<*project directory*>****<*variation name*>**_example_design\simulation\verilog\submodules\altera_avalon_reset_source.sv**
     or
     ****<*project directory*>****<*variation name*>**_example_design\simulation\vhdl\submodules\altera_avalon_reset_source.vhd**
   - ****<*project directory*>****<*variation name*>**_example_design\simulation\verilog**<*variation name*>**_example_sim.v**
     or
     \uniphy<*variation name*>**_example_design\simulation\vhdl**<*variation name*>**_example_sim.vhd**
   - ****<*project directory*>****<*variation name*>**_example_design\simulation\verilog\submodules\verbosity_pkg.sv**
   - ****<*project directory*>****<*variation name*>**_example_design\simulation\verilog\submodules\status_checker_no_ifdef_params.sv**
     or
     ****<*project directory*>****<*variation name*>**_example_design\simulation\vhdl\submodules\status_checker_no_ifdef_params.sv**
   - ****<*project directory*>****<*variation name*>**_example_design\simulation\verilog\submodules\alt_mem_if_common_ddr_mem_model_ddr3_mem_if_dm_pins_en_mem_if_dqsn_en.sv**
     or
     ****<*project directory*>****<*variation name*>**_example_design\simulation\vhdl\submodules\alt_mem_if_common_ddr_mem_model_ddr3_mem_if_dm_pins_en_mem_if_dqsn_en.sv**
   - ****<*project directory*>****<*variation name*>**_example_design\simulation\verilog\submodules\alt_mem_if_ddr3_mem_model_top_ddr3_mem_if_dm_pins_en_mem_if_dqsn_en**
     or
     ****<*project directory*>****<*variation name*>**_example_design\simulation\vhdl\submodules\alt_mem_if_ddr3_mem_model_top_ddr3_mem_if_dm_pins_en_mem_if_dqsn_en**
4. Open the <*variation name*>**_example_sim.v** file and find the UniPHY-generated simulation example design module name below: <*variation name*>**_example_sim_e0**.
5. Change the module name above to the name of your top-level design module.

6. Update the following port names of the example design in the UniPHY-generated <*variation name*>**_example_sim.v** file. (Table 14–2).

Table 14–2. Example Design Port Names

Example Design Name	New Name
`pll_ref_clk`	Rename to `clock_source`.
`mem_a`	Rename to `mem_addr`.
`mem_ck`	Rename to `mem_clk`.
`mem_ck_n`	Rename to `mem_clk_n`.
`mem_dqs_n`	Rename to `mem_dqsn`.
`drv_status_pass`	Rename to `pnf`.
`afi_clk`	Rename to `phy_clk`.
`afi_reset_n`	Rename to `reset_phy_clk_n`.
`drv_status_fail`	This signal is not available, so comment out this output.
`afi_half_clk`	This signal is not exposed to the top-level design, so comment out this output.

For more information about generating example simulation files, refer to Simulating Memory IP, in volume 2 of the *External Memory Interface Handbook*.

Document Revision History

Table 14–3 lists the revision history for this document.

Table 14–3. Document Revision History

Date	Version	Changes
November 2012	2.3	Changed chapter number from 12 to 14.
June 2012	2.2	Added Feedback icon.
November 2011	2.1	Revised Simulating your Design section.

Section III. ALTMEMPHY Reference

This section provides reference information about the ALTMEMPHY-based external memory interface IP.

This section includes the following chapters:

- Chapter 15, Introduction to ALTMEMPHY IP
- Chapter 16, Latency for ALTMEMPHY IP
- Chapter 17, Timing Diagrams for ALTMEMPHY IP
- Chapter 18, ALTMEMPHY External Memory Interface Debug Toolkit

For information about the revision history for chapters in this section, refer to "Document Revision History" in each individual chapter.

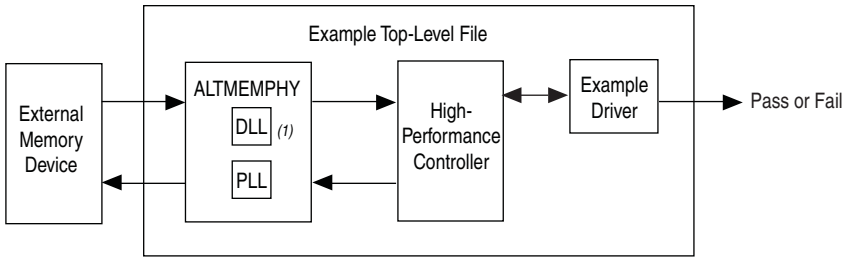
15. Introduction to ALTMEMPHY IP

The Altera® DDR, DDR2, and DDR3 SDRAM Controllers with ALTMEMPHY IP provide simplified interfaces to industry-standard DDR, DDR2, and DDR3 SDRAM. The ALTMEMPHY megafunction is an interface between a memory controller and the memory devices, and performs read and write operations to the memory. The DDR, DDR2, and DDR3 SDRAM Controllers with ALTMEMPHY IP work in conjunction with the Altera ALTMEMPHY megafunction.

The DDR and DDR2 SDRAM Controllers with ALTMEMPHY IP and ALTMEMPHY megafunction offer full-rate or half-rate DDR and DDR2 SDRAM interfaces.The DDR3 SDRAM Controller with ALTMEMPHY IP and ALTMEMPHY megafunction support DDR3 SDRAM interfaces in half-rate mode. The DDR, DDR2, and DDR3 SDRAM Controllers with ALTMEMPHY IP offer the high-performance controller II (HPC II), which provides high efficiency and advanced features.

Figure 15–1 shows a system-level diagram including the example top-level file that the DDR, DDR2, or DDR3 SDRAM Controller with ALTMEMPHY IP creates for you.

Figure 15–1. System-Level Diagram



Note to Figure 15–1:

(1) When you choose **Instantiate DLL Externally**, delay-locked loop (DLL) is instantiated outside the ALTMEMPHY megafunction.

The MegaWizard™ Plug-In Manager generates an example top-level file, consisting of an example driver, and your DDR, DDR2, or DDR3 SDRAM high-performance controller custom variation. The controller instantiates an instance of the ALTMEMPHY megafunction which in turn instantiates a phase-locked loop (PLL) and DLL. You can also instantiate the DLL outside the ALTMEMPHY megafunction to share the DLL between multiple instances of the ALTMEMPHY megafunction. You cannot share a PLL between multiple instances of the ALTMEMPHY megafunction, but you may share some of the PLL clock outputs between these multiple instances.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

The example top-level file is a fully-functional design that you can simulate, synthesize, and use in hardware. The example driver is a self-test module that issues read and write commands to the controller and checks the read data to produce the pass or fail, and test complete signals.

The ALTMEMPHY megafunction creates the datapath between the memory device and the memory controller. The megafunction is available as a stand-alone product or can be used in conjunction with the Altera high-performance memory controller. When using the ALTMEMPHY megafunction as a stand-alone product, use with either custom or third-party controllers.

For new designs, Altera recommends using a UniPHY-based external memory interface, such as the DDR2 and DDR3 SDRAM controllers with UniPHY, QDR II and QDR II+ SRAM controllers with UniPHY, or RLDRAM II controller with UniPHY.

Release Information

Table 15–1 provides information about this release of the DDR3 SDRAM Controller with ALTMEMPHY IP.

Table 15–1. Release Information

Item	Description
Version	11.1
Release Date	November 2011
Ordering Codes	IP-SDRAM/HPDDR (DDR SDRAM HPC) IP-SDRAM/HPDDR2 (DDR2 SDRAM HPC) IP-HPMCII (HPC II)
Product IDs	00BE (DDR SDRAM) 00BF (DDR2 SDRAM) 00C2 (DDR3 SDRAM) 00CO (ALTMEMPHY Megafunction)
Vendor ID	6AF7

Altera verifies that the current version of the Quartus® II software compiles the previous version of each MegaCore function. The *MegaCore IP Library Release Notes and Errata* report any exceptions to this verification. Altera does not verify compilation with MegaCore function versions older than one release. For information about issues on the DDR, DDR2, or DDR3 SDRAM high-performance controller and the ALTMEMPHY megafunction in a particular Quartus II version, refer to the *Quartus II Software Release Notes*.

Device Family Support

Table 15–2 defines the device support levels for Altera IP cores.

Table 15–2. Altera IP Core Device Support Levels

FPGA Device Families	HardCopy Device Families
Preliminary support—The IP core is verified with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. It can be used in production designs with caution.	**HardCopy Companion**—The IP core is verified with preliminary timing models for the HardCopy companion device. The IP core meets all functional requirements, but might still be undergoing timing analysis for the HardCopy device family. It can be used in production designs with caution.
Final support—The IP core is verified with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.	**HardCopy Compilation**—The IP core is verified with final timing models for the HardCopy device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.

Table 15–3 shows the level of support offered by the DDR, DDR2, and DDR3 SDRAM Controllers with ALTMEMPHY IP for Altera device families.

Table 15–3. Device Family Support

Device Family	Protocol	
	DDR and DDR2	DDR3
Arria® GX	Final	No support
Arria II GX	Final	Final
Cyclone® III	Final	No support
Cyclone III LS	Final	No support
Cyclone IV E	Final	No support
Cyclone IV GX	Final	No support
HardCopy II	Refer to the What's New in Altera IP page of the Altera website.	No support
Stratix® II	Final	No support
Stratix II GX	Final	No support
Other device families	No support	No support

Features

ALTMEMPHY Megafunction

Table 15–4 summarizes key feature support for the ALTMEMPHY megafunction.

Table 15–4. ALTMEMPHY Megafunction Feature Support

Feature	DDR and DDR2	DDR3
Support for the Altera PHY Interface (AFI) on all supported devices.	✓	✓
Automated initial calibration eliminating complicated read data timing calculations.	✓	✓
Voltage and temperature (VT) tracking that guarantees maximum stable performance for DDR, DDR2, and DDR3 SDRAM interfaces.	✓	✓
Self-contained datapath that makes connection to an Altera controller or a third-party controller independent of the critical timing paths.	✓	✓
Full-rate interface	✓	—
Half-rate interface	✓	✓
Easy-to-use parameter editor	✓	✓

In addition, the ALTMEMPHY megafunction supports DDR3 SDRAM components without leveling:

- The ALTMEMPHY megafunction supports DDR3 SDRAM components without leveling for Arria II GX devices using T-topology for clock, address, and command bus:
  - Supports multiple chip selects.
- The DDR3 SDRAM PHY without leveling f_{MAX} is 400 MHz for single chip selects.
- No support for data-mask (DM) pins for ×4 DDR3 SDRAM DIMMs or components, so select **No** for **Drive DM pins from FPGA** when using ×4 devices.
- The ALTMEMPHY megafunction supports half-rate DDR3 SDRAM interfaces only.

High-Performance Controller II

Table 15–5 summarizes key feature support for the DDR, DDR2, and DDR3 SDRAM HPC II.

Table 15–5. Feature Support (Part 1 of 2)

Feature	DDR and DDR2	DDR3
Half-rate controller	✓	✓
Support for AFI ALTMEMPHY	✓	✓
Support for Avalon®Memory Mapped (Avalon-MM) local interface	✓	✓

Table 15–5. Feature Support (Part 2 of 2)

Feature	DDR and DDR2	DDR3
Configurable command look-ahead bank management with in-order reads and writes	✓	✓
Additive latency	✓	✓
Support for arbitrary Avalon burst length	✓	✓
Built-in flexible memory burst adapter	✓	✓
Configurable Local-to-Memory address mappings	✓	✓
Optional run-time configuration of size and mode register settings, and memory timing	✓	✓
Partial array self-refresh (PASR)	✓	✓
Support for industry-standard DDR3 SDRAM devices		
Optional support for self-refresh command	✓	✓
Optional support for user-controlled power-down command	✓	✓
Optional support for automatic power-down command with programmable time-out	✓	✓
Optional support for auto-precharge read and auto-precharge write commands	✓	✓
Optional support for user-controller refresh	✓	✓
Optional multiple controller clock sharing in SOPC Builder Flow		
Integrated error correction coding (ECC) function 72-bit	✓	✓
Integrated ECC function, 16, 24, and 40-bit	✓	✓
Support for partial-word write with optional automatic error correction	✓	✓
SOPC Builder ready		
Support for OpenCore Plus evaluation	✓	✓
IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulator	✓	✓

Notes to Table 15–5:

(1) HPC II supports additive latency values greater or equal to t_{RCD}-1, in clock cycle unit (t_{CK}).

(2) This feature is not supported with DDR3 SDRAM with leveling.

Unsupported Features

Table 15–6 summarizes unsupported features for Altera's ALTMEMPHY-based external memory interfaces.

Table 15–6. Unsupported Features

Memory Protocol	Unsuppoted Feature
DDR and DDR2 SDRAM	Timing simulation
	Burst length of 2
	Partial burst and unaligned burst in ECC and non-ECC mode when DM pins are disabled
DDR3 SDRAM	Timing simulation
	Partial burst and unaligned burst in ECC and non-ECC mode when DM pins are disabled
	Stratix III and Stratix IV
	DIMM support
	Full-rate interfaces

MegaCore Verification

Altera performs extensive random, directed tests with functional test coverage using industry-standard Denali models to ensure the functionality of the DDR, DDR2, and DDR3 SDRAM Controllers with ALTMEMPHY IP.

Resource Utilization

This section provides typical resource utilization information for the external memory controllers with ALTMEMPHY for supported device families. This information is provided as a guideline only; for precise resource utilization data, you should generate your IP core and refer to the reports generated by the Quartus II software.

Table 15–7 shows resource utilization data for the ALTMEMPHY megafunction, and the DDR3 high-performance controller II for Arria II GX devices.

Table 15–7. Resource Utilization in Arria II GX Devices (Part 1 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
Controller							
DDR3 (Half rate)	8	1,883	1,505	10	2	0	4,352
	16	1,893	1,505	10	4	0	8,704
	64	1,946	1,521	18	15	0	34,560
	72	1,950	1,505	10	17	0	39,168

Table 15–7. Resource Utilization in Arria II GX Devices (Part 2 of 2)

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
Controller+PHY							
DDR3 (Half rate)	8	3,389	2,760	12	4	0	4,672
	16	3,457	2,856	12	7	0	9,280
	64	3,793	3,696	20	24	0	36,672
	72	3,878	3,818	12	26	0	41,536

Table 15–8 shows resource utilization data for the DDR2 high-performance controller and controller plus PHY, for half-rate and full-rate configurations for Arria II GX devices.

Table 15–8. DDR2 Resource Utilization in Arria II GX Devices

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
Controller							
DDR2 (Half rate)	8	1,971	1,547	10	2	0	4,352
	16	1,973	1,547	10	4	0	8,704
	64	2,028	1,563	18	15	0	34,560
	72	2,044	1,547	10	17	0	39,168
DDR2 (Full rate)	8	2,007	1,565	10	2	0	2,176
	16	2,013	1,565	10	2	0	4,352
	64	2,022	1,565	10	8	0	17,408
	72	2,025	1,565	10	9	0	19,584
Controller+PHY							
DDR2 (Half rate)	8	3,481	2,722	12	4	0	4,672
	16	3,545	2,862	12	7	0	9,280
	64	3,891	3,704	20	24	0	36,672
	72	3,984	3,827	12	26	0	41,536
DDR2 (Full rate)	8	3,337	2,568	29	2	0	2,176
	16	3,356	2,558	11	4	0	4,928
	64	3,423	2,836	31	12	0	19,200
	72	3,445	2,827	11	14	0	21,952

Table 15–9 shows resource utilization data for the DDR2 high-performance controller and controller plus PHY, for half-rate and full-rate configurations for Cyclone III devices.

Table 15–9. DDR2 Resource Utilization in Cyclone III Devices

Protocol	Memory Width (Bits)	Logic Registers	Logic Cells	M9K Blocks	Memory (Bits)
Controller					
DDR2 (Half rate)	8	1,513	3,015	4	4,464
	16	1,513	3,034	6	8,816
	64	1,513	3,082	18	34,928
	72	1,513	3,076	19	39,280
DDR2 (Full rate)	8	1,531	3,059	4	2,288
	16	1,531	3,108	4	4,464
	64	1,531	3,134	10	17,520
	72	1,531	3,119	11	19,696
Controller+PHY					
DDR2 (Half rate)	8	2,737	5,131	6	4,784
	16	2,915	5,351	9	9,392
	64	3,969	6,564	27	37,040
	72	4,143	6,786	28	41,648
DDR2 (Full rate)	8	2,418	4,763	6	2,576
	16	2,499	4,919	6	5,008
	64	2,957	5,505	15	19,600
	72	3,034	5,608	16	22,032

System Requirements

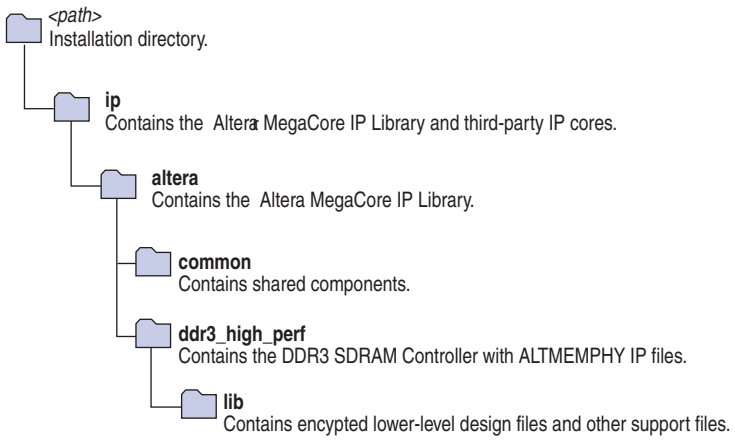
The DDR3 SDRAM Controller with ALTMEMPHY IP is a part of the MegaCore IP Library, which is distributed with the Quartus II software and downloadable from the Altera website, www.altera.com.

For system requirements and installation instructions, refer to *Altera Software Installation & Licensing*.

Installation and Licensing

Figure 15–2 shows the directory structure after you install the DDR3 SDRAM Controller with ALTMEMPHY IP, where <*path*> is the installation directory. The default installation directory on Windows is **c:\altera**\<*version*>; on Linux it is **/opt/altera**<*version*>.

Figure 15–2. Directory Structure



You need a license for the MegaCore function only when you are completely satisfied with its functionality and performance, and want to take your design to production.

To use the DDR3 SDRAM HPC, you can request a license file from the Altera web site at www.altera.com/licensing and install it on your computer. When you request a license file, Altera emails you a **license.dat** file. If you do not have Internet access, contact your local representative.

To use the DDR3 SDRAM HPC II, contact your local sales representative to order a license.

Free Evaluation

Altera's OpenCore Plus evaluation feature is only applicable to the DDR3 SDRAM HPC. With the OpenCore Plus evaluation feature, you can perform the following actions:

- Simulate the behavior of a megafunction (Altera MegaCore function or AMPP^{SM} megafunction) within your system.
- Verify the functionality of your design, as well as evaluate its size and speed quickly and easily.
- Generate time-limited device programming files for designs that include MegaCore functions.
- Program a device and verify your design in hardware.

You need to purchase a license for the megafunction only when you are completely satisfied with its functionality and performance, and want to take your design to production.

OpenCore Plus Time-Out Behavior

OpenCore Plus hardware evaluation can support the following two modes of operation:

- Untethered—the design runs for a limited time
- Tethered—requires a connection between your board and the host computer. If tethered mode is supported by all megafunctions in a design, the device can operate for a longer time or indefinitely

All megafunctions in a device time-out simultaneously when the most restrictive evaluation time is reached. If there is more than one megafunction in a design, a specific megafunction's time-out behavior may be masked by the time-out behavior of the other megafunctions.

For MegaCore functions, the untethered time-out is 1 hour; the tethered time-out value is indefinite.

Your design stops working after the hardware evaluation time expires and the `local_ready` output goes low.

Document Revision History

Table 15–10 lists the revision history for this document.

Table 15–10. Document Revision History

Date	Version	Changes
November 2012	1.2	Changed chapter number from 13 to 15.
June 2012	1.1	Added Feedback icon.
November 2011	1.0	Combined Release Information, Device Family Support, Features list, and Unsupported Features list for DDR, DDR2, and DDR3.

16. Latency for ALTMEMPHY IP

Latency is defined using the local (user) side frequency and absolute time (ns). There are two types of latencies that exists while designing with memory controllers—read and write latencies, which have the following definitions:

- Read latency—the amount of time it takes for the read data to appear at the local interface after initiating the read request.
- Write latency—the amount of time it takes for the write data to appear at the memory interface after initiating the write request.

For a half-rate controller, the local side frequency is half of the memory interface frequency. For a full-rate controller, the local side frequency is equal to the memory interface frequency.

Altera defines read and write latencies in terms of the local interface clock frequency and by the absolute time for the memory controllers. These latencies apply to supported device families with half-rate and full-rate HPC II memory controllers.

The latency defined in this section uses the following assumptions:

- The row is already open, there is no extra bank management needed.
- The controller is idle, there is no queued transaction pending, indicated by the `local_ready` signal asserted high.
- No refresh cycles occur before the transaction.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Latency Stages

The latency for the high-performance controller II comprises many different stages of the memory interface. Figure 16–1 shows a typical memory interface read latency path showing read latency from the time a `local_read_req` signal assertion is detected by the controller up to data available to be read from the dual-port RAM (DPRAM) module.

Figure 16–1. Typical Latency Path

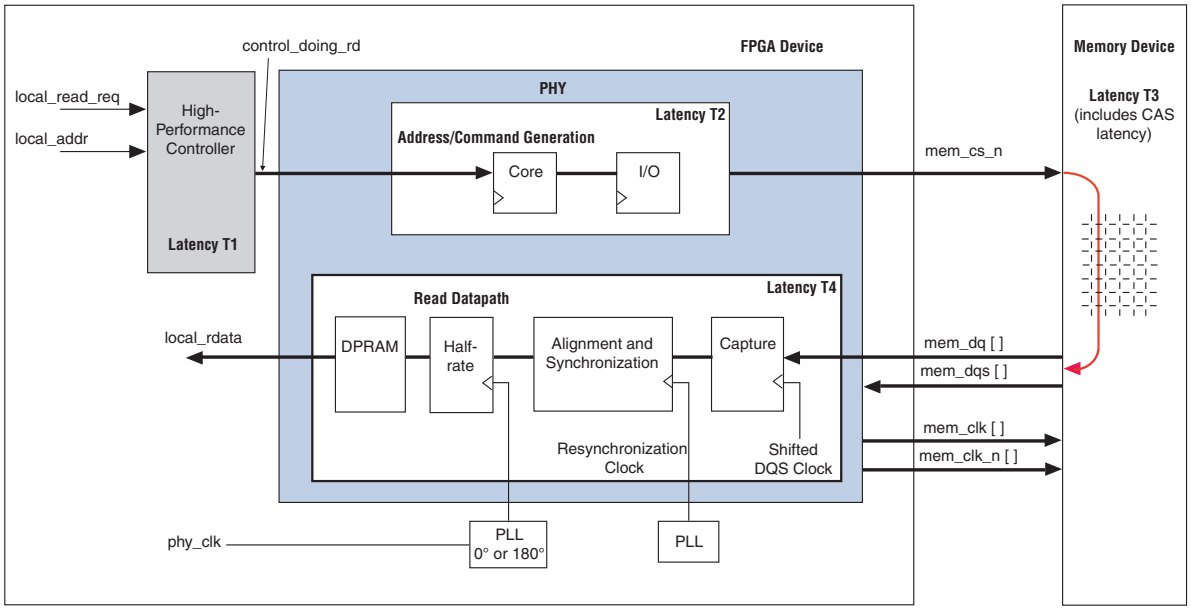


Table 16–1 shows the different stages that make up the whole read and write latency that Figure 16–1 shows.

Table 16–1. High-Performance Controller Latency Stages and Descriptions

Latency Number	Latency Stage	Description
T1	Controller	`local_read_req` or `local_write_req` signal assertion to `ddr_cs_n` signal assertion.
T2	Command Output	`ddr_cs_n` signal assertion to `mem_cs_n` signal assertion.
T3	CAS or WL	Read command to DQ data from the memory or write command to DQ data to the memory.
T4	ALTMEMPHY read data input	Read data appearing on the local interface.
T2 + T3	Write data latency	Write data appearing on the memory interface.

From Figure 16–1, the read latency in the high-performance controllers is made up of four components:

read latency = controller latency (T1) + command output latency (T2) + CAS latency (T3) + PHY read data input latency (T4)

Similarly, the write latency in the high-performance controller II is made up of three components:

write latency = controller latency (T1) + write data latency (T2+T3)

You can separate the controller and ALTMEMPHY read data input latency into latency that occurred in the I/O element (IOE) and latency that occurred in the FPGA fabric.

Table 16–2 shows the minimum and maximum supported CAS latency for the DDR and DDR2 SDRAM high-performance controller II.

Table 16–2. Supported CAS Latency *(1)*

Device Family	Minimum Supported CAS Latency		Maximum Supported CAS Latency	
	DDR	DDR2	DDR	DDR2
Arria GX	3.0	3.0	3.0	6.0
Arria II GX	3.0	3.0	3.0	6.0
Cyclone III	2.0	3.0	3.0	6.0
Cyclone IV	2.0	3.0	3.0	6.0
HardCopy III	3.0	3.0	3.0	6.0
HardCopy IV	3.0	3.0	3.0	6.0
Stratix II	3.0	3.0	3.0	6.0
Stratix III	3.0	3.0	3.0	6.0
Stratix IV	3.0	3.0	3.0	6.0

Note to Table 16–2:

(1) The registered DIMMs, where supported, effectively introduce one extra cycle of CAS latency. For the registered DIMMs, you need to subtract 1.0 from the CAS figures to determine the minimum supported CAS latency, and add 1.0 to the CAS figures to determine the maximum supported CAS latency.

Table 16–3 and Table 16–4 show a typical latency that can be achieved in Arria GX, Arria II GX, Cyclone III, Cyclone IV, Stratix IV, Stratix III, Stratix II, and Stratix II GX devices. The exact latency for your memory controller depends on your precise configuration. You can obtain precise latency from simulation, but this figure can vary slightly in hardware because of the automatic calibration process.

The actual memory CAS and write latencies shown are halved in half-rate designs as the latency calculation is based on the local clock.

The read latency also depends on your board trace delay. The latency found in simulation can be different from that found in board testing as functional simulation does not take into account the board trace delays. For a given design on a given board, the latency may change by one clock cycle (for full-rate designs) or two clock cycles (for half-rate designs) upon resetting the board. Different boards could also show different latencies even with the same design.

The CAS and write latencies are different between DDR and DDR2 SDRAM interfaces. To calculate latencies for DDR SDRAM interfaces, use the numbers from DDR2 SDRAM listed below and replace the CAS and write latency with the DDR SDRAM values.

Table 16–3. Typical Read Latency in HPC II (1), (2)

Device	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		CAS Latency (4)	Read Data Latency		Total Read Latency (5)	
				FPGA	I/O		FPGA	I/O	Local Clock Cycles	Time (ns)
Arria GX	233	Half-rate	5	3	1	2	4.5	1	18	154
	167	Full-rate	5	2	1	4	5	1	19	114
Arria II GX	233	Half-rate	5	3	1	2.5	5.5	1	18	154
	167	Full-rate	5	2	1	4	6	1	20	120
Cyclone III and Cyclone IV	200	Half-rate	5	3	1	2	4.5	1	18	180
	167	Full-rate	5	2	1	4	5	1	19	114
Stratix II and Stratix II GX	333	Half-rate	5	3	1	2	4.5	1	18	108
	267	Half-rate	5	3	1	2	4.5	1	18	135
	200	Full-rate	5	2	1	4	5	1	19	95
Stratix III and Stratix IV	400	Half-rate	5	3	1	2.5	7.125	1.5	20	100
	267	Full-rate	4	2	1.5	4	7	1	20	75

Notes to Table 16–3:

(1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.

(2) Numbers shown may have been rounded up to the nearest higher integer.

(3) The controller latency value is from the Altera high-performance controller.

(4) CAS latency is per memory device specification and is programmable in the MegaWizard Plug-In Manager.

(5) Total read latency is the sum of controller, address and command, CAS, and read data latencies.

Table 16–4. Typical Write Latency in HPC II (1), (2) (Part 1 of 2)

Device	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
Arria GX	233	Half-rate	5	3	1	1.5	12	103
	167	Full-rate	5	2	1	3	12	72
Arria II GX	233	Half-rate	5	3	1	2.5	12	103
	167	Full-rate	5	2	1	4	12	72
Cyclone III and Cyclone IV	200	Half-rate	5	3	1	1.5	12	120
	167	Full-rate	5	2	1	3	12	72

Table 16–4. Typical Write Latency in HPC II (1), (2) (Part 2 of 2)

Device	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
Stratix II and Stratix II GX	333	Half-rate	5	3	1	1.5	12	72
	267	Half-rate	5	3	1	1.5	12	90
	200	Full-rate	5	2	1	3	12	60
Stratix III and Stratix IV	400	Half-rate	5	3	1	2	12	60
	267	Full-rate	5	2	1.5	3	13	49

Notes to Table 16–4:

(1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.

(2) Numbers shown may have been rounded up to the nearest higher integer.

(3) The controller latency value is from the Altera high-performance controller.

(4) Memory write latency is per memory device specification. The latency from when you provide the command to write to when you need to provide data at the memory device.

(5) Total write latency is the sum of controller, address and command, and memory write latencies.

To see the latency incurred in the IOE for both read and write paths for ALTMEMPHY variations in Stratix IV and Stratix III devices refer to the IOE figures in the *External Memory Interfaces in Stratix III Devices* chapter of the *Stratix III Device Handbook* and the *External Memory Interfaces in Stratix IV Devices* chapter of the *Stratix IV Device Handbook*.

Document Revision History

Table 16–5 lists the revision history for this document.

Table 16–5. Document Revision History

Date	Version	Changes
November 2012	1.2	Changed chapter number from 14 to 16.
June 2012	1.1	Added Feedback icon.
November 2011	1.0	Consolidated latency information from 11.0 **DDR and DDR2 SDRAM Controller with ALTMEMPHY IP User Guide**, and **DDR3 SDRAM Controller with ALTMEMPHY IP User Guide**.

17. Timing Diagrams for ALTMEMPHY IP

EMI_RM_015-1.3

This chapter shows timing diagrams for the DDR, DDR2, and DDR3 SDRAM high-performance controllers II (HPC II).

DDR and DDR2 High-Performance Controllers II

This section discusses the following timing diagrams for the DDR and DDR2 HPC II:

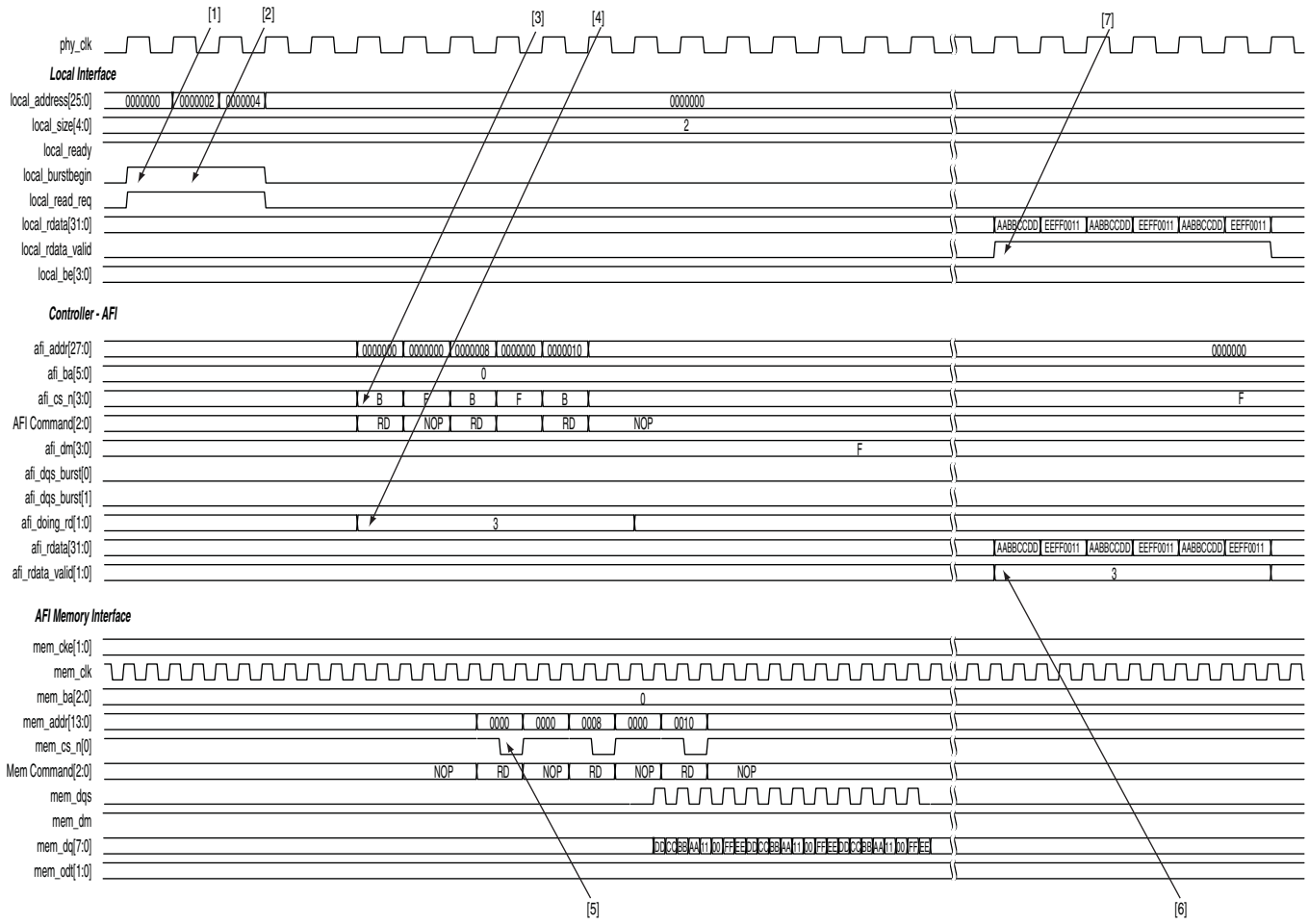
- "Half-Rate Read"
- "Half-Rate Write"
- "Full-Rate Read"
- "Full-Rate Write"

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

Half-Rate Read

Figure 17–1. Half-Rate Read Operation for HPC II



The following sequence corresponds with the numbered items in Figure 17–1:

1. The user logic requests the first read by asserting the `local_read_req` signal, and the size and address for this read. In this example, the request is a burst of length of 2 to the local address `0x000000`. This local address is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x000000

   mem_col_address = 0x0000

   mem_bank_address = 0x00
   ```

2. The user logic initiates a second read to a different memory column within the same row. The request for the second write is a burst length of 2. In this example, the user logic continues to accept commands until the command queue is full. When the command queue is full, the controller deasserts the `local_ready` signal. The starting local address 0x000002 is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x0002<<2 = 0x0008

   mem_bank_address = 0x00
   ```

3. The controller issues the first read memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.

4. The controller asserts the `afi_doing_rd` signal to indicate to the ALTMEMPHY megafunction the number of clock cycles of read data it must expect for the first read. The ALTMEMPHY megafunction uses the `afi_doing_rd` signal to enable its capture registers for the expected duration of memory burst.

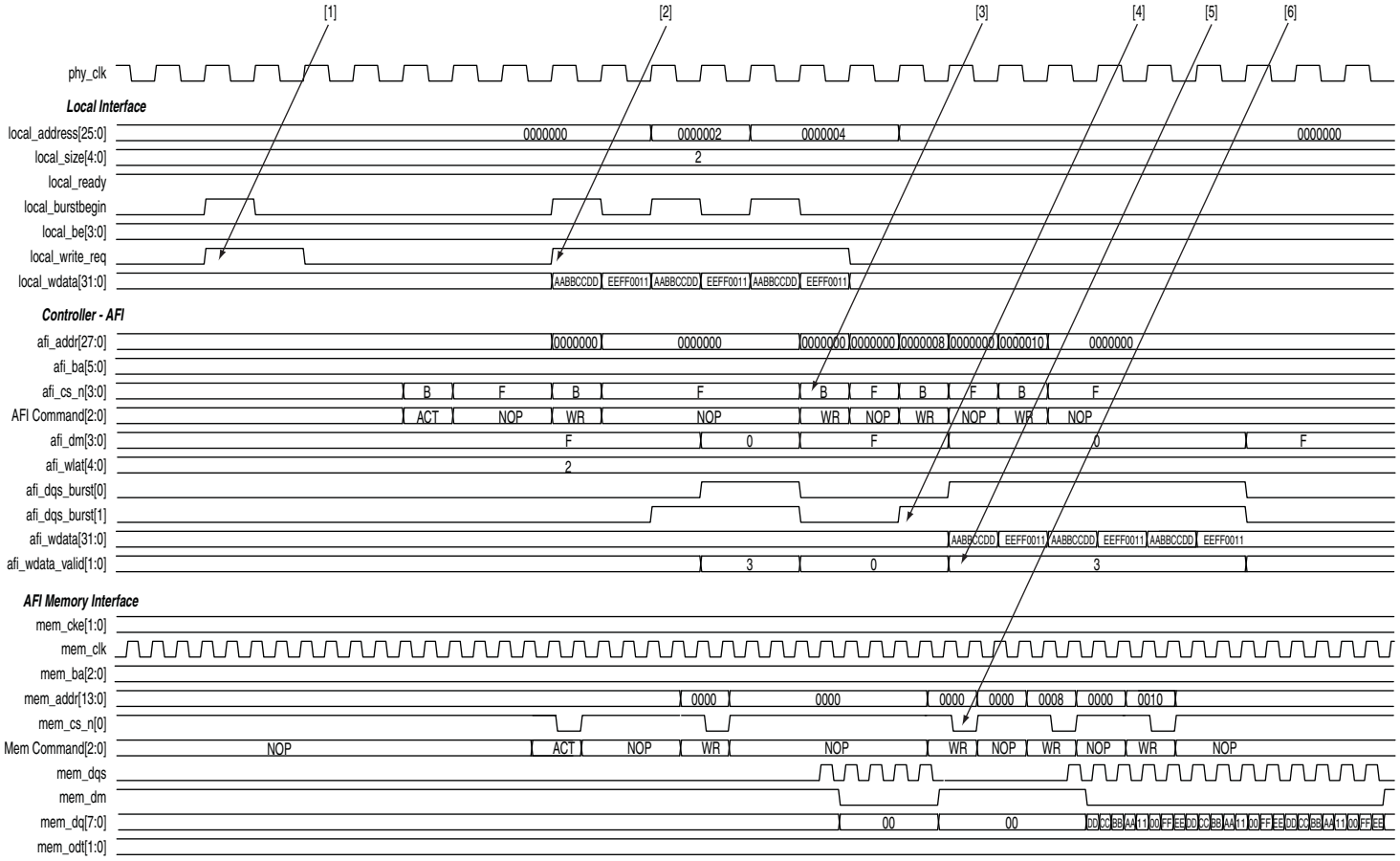
5. The ALTMEMPHY megafunction issues the first read command to the memory and captures the read data from the memory.

6. The ALTMEMPHY megafunction returns the first data read to the controller after resynchronizing the data to the `phy_clk` domain, by asserting the `afi_rdata_valid` signal when there is valid read data on the `afi_rdata` bus.

7. The controller returns the first read data to the user by asserting the `local_rdata_valid` signal when there is valid read data on the `local_rdata` bus. If the ECC logic is disabled, there is no delay between the `afi_rdata` and the `local_rdata` buses. If there is ECC logic in the controller, there is one or three clock cycles of delay between the `afi_rdata` and `local_rdata` buses.

Half-Rate Write

Figure 17–2. Half-Rate Write Operation for HPC II

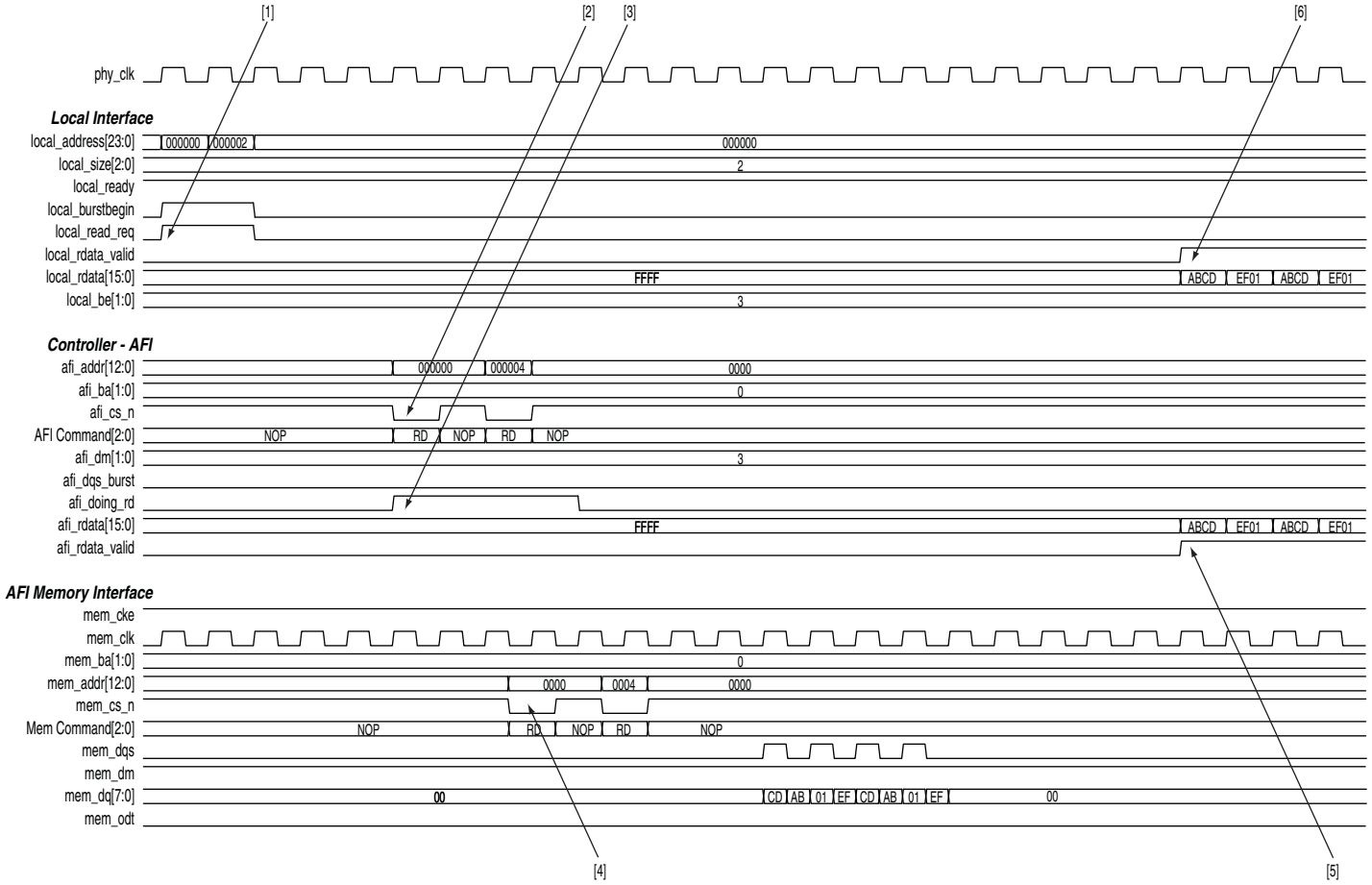


The following sequence corresponds with the numbered items in Figure 17–2:

1. The user logic asserts the first write request to row 0 so that row 0 is open before the next transaction.
2. The user logic asserts a second `local_write_req` signal with size of 2 and address of 0 (col = 0, row = 0, bank = 0, chip = 0). The `local_ready` signal is asserted along with the `local_write_req` signal, which indicates that the controller has accepted this request, and the user logic can request another read or write in the following clock cycle. If the `local_ready` signal was not asserted, the user logic must keep the write request, size, and address signals asserted until the `local_ready` signal is registered high.
3. The controller issues the necessary memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
4. The controller asserts the `afi_wdata_valid` signal to indicate to the ALTMEMPHY megafunction that valid write data and write data masks are present on the inputs to the ALTMEMPHY megafunction.
5. The controller asserts the `afi_dqs_burst` signals to control the timing of the DQS signal that the ALTMEMPHY megafunction issues to the memory.
6. The ALTMEMPHY megafunction issues the write command, and sends the write data and write DQS to the memory.

Full-Rate Read

Figure 17–3. Full-Rate Read Operation for HPC II



The following sequence corresponds with the numbered items in Figure 17–3:

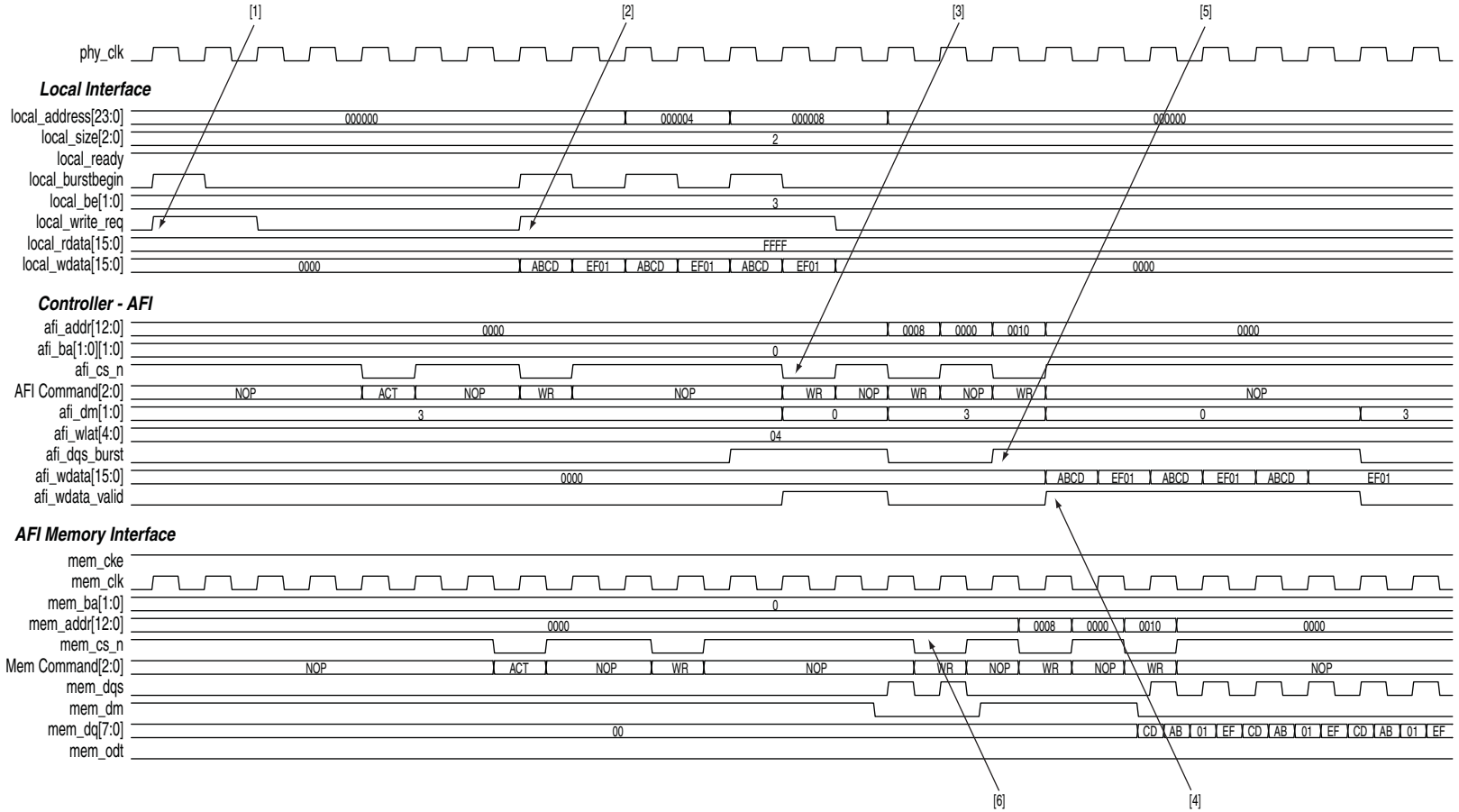
1. The user logic requests the first read by asserting `local_read_req` signal, and the size and address for this read. In this example, the request is a burst length of 2 to a local address `0x000000`. This local address is mapped to the following memory address in full-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x0000<<2 = 0x0000

   mem_bank_address = 0x00
   ```

2. The controller issues the first read memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
3. The controller asserts the `afi_doing_rd` signal to indicate to the ALTMEMPHY megafunction the number of clock cycles of read data it must expect for the first read. The ALTMEMPHY megafunction uses the `afi_doing_rd` signal to enable its capture registers for the expected duration of memory burst.
4. The ALTMEMPHY megafunction issues the first read command to the memory and captures the read data from the memory.
5. The ALTMEMPHY megafunction returns the first data read to the controller after resynchronizing the data to the `phy_clk` domain, by asserting the `afi_rdata_valid` signal when there is valid read data on the `afi_rdata` bus.
6. The controller returns the first read data to the user by asserting the `local_rdata_valid` signal when there is valid read data on the `local_rdata` bus. If the ECC logic is disabled, there is no delay between the `afi_rdata` and the `local_rdata` buses. If there is ECC logic in the controller, there is one or three clock cycles of delay between the `afi_rdata` and `local_rdata` buses.

Full-Rate Write

Figure 17–4. Full-Rate Write Operation for HPC II



The following sequence corresponds with the numbered items in Figure 17–4:

1. The user logic asserts the first write request to row 0 so that row 0 is open before the next transaction.
2. The user logic asserts a second `local_write_req` signal with a size of 2 and address of 0 (col = 0, row = 0, bank = 0, chip = 0). The `local_ready` signal is asserted along with the `local_write_req` signal, which indicates that the controller has accepted this request, and the user logic can request another read or write in the following clock cycle. If the `local_ready` signal was not asserted, the user logic must keep the write request, size, and address signals asserted until the `local_ready` signal is registered high.
3. The controller issues the necessary memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
4. The controller asserts the `afi_wdata_valid` signal to indicate to the ALTMEMPHY megafunction that valid write data and write data masks are present on the inputs to the ALTMEMPHY megafunction.
5. The controller asserts the `afi_dqs_burst` signals to control the timing of the DQS signal that the ALTMEMPHY megafunction issues to the memory.
6. The ALTMEMPHY megafunction issues the write command, and sends the write data and write DQS to the memory.

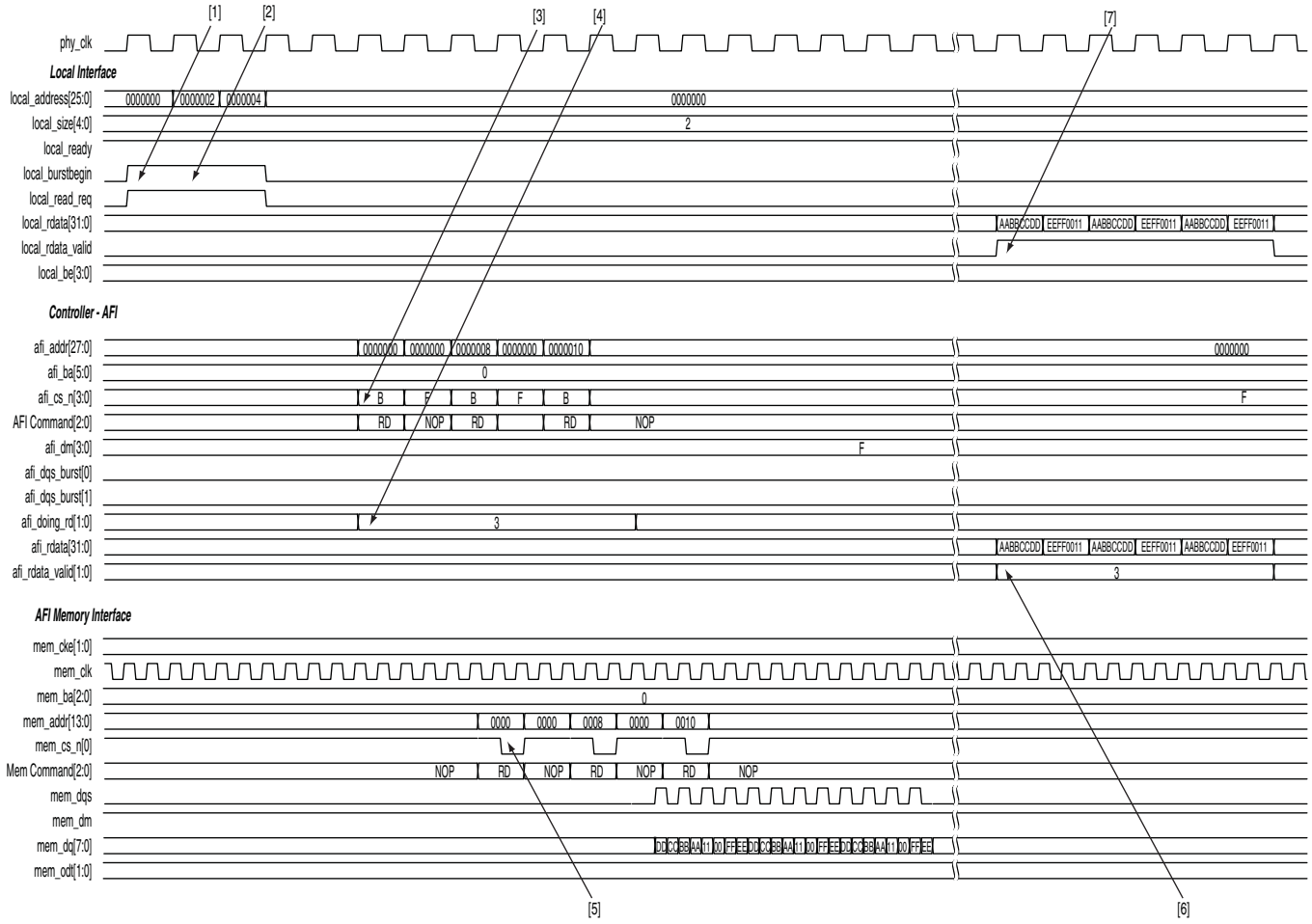
DDR3 High-Performance Controller II

This section discusses the following timing diagrams for the DDR3 HPC II:

- "Half-Rate Read"
- "Half-Rate Write"
- "Half-Rate Read (Non Burst-Aligned Address)"
- "Half-Rate Write (Non Burst-Aligned Address)"
- "Half-Rate Read With Gaps"
- "Full-Rate Read"
- "Half-Rate Write Operation (Merging Writes)"
- "Write-Read-Write-Read Operation"

Half-Rate Read (Burst-Aligned Address)

Figure 17–5. Half-Rate Read Operation for HPC II—Burst-Aligned Address



The following sequence corresponds with the numbered items in Figure 17–1:

1. The user logic requests the first read by asserting the `local_read_req` signal, and the size and address for this read. In this example, the request is a burst of length of 2 to the local address `0x000000`. This local address is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x000000

   mem_col_address = 0x0000

   mem_bank_address = 0x00
   ```

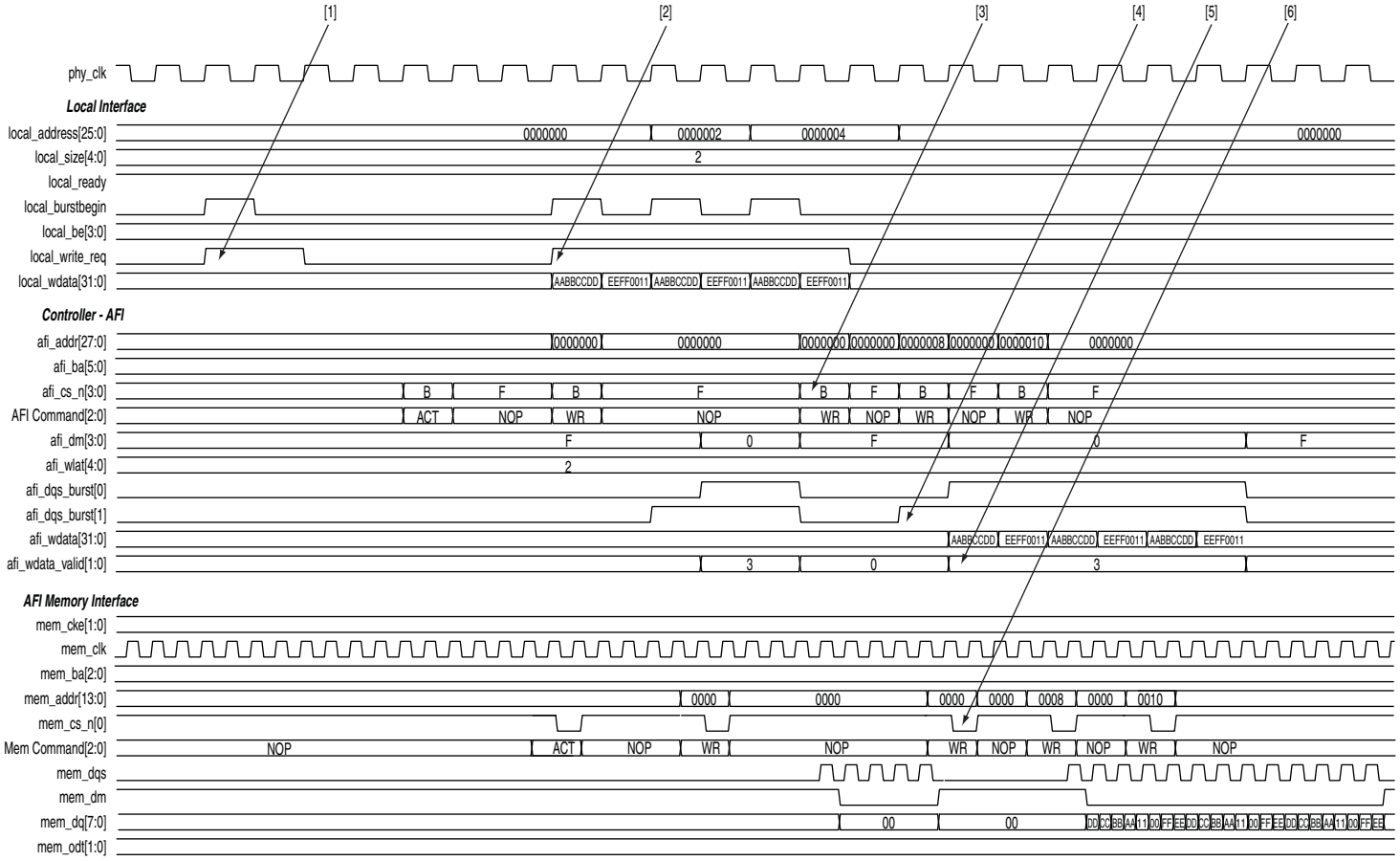
2. The user logic initiates a second read to a different memory column within the same row. The request for the second read is a burst length of 2. In this example, the user logic continues to accept commands until the command queue is full. When the command queue is full, the controller deasserts the `local_ready` signal. The starting local address 0x000002 is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x0002<<2 = 0x0008

   mem_bank_address = 0x00
   ```

3. The controller issues the first read memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
4. The controller asserts the `afi_doing_rd` signal to indicate to the ALTMEMPHY megafunction the number of clock cycles of read data it must expect for the first read. The ALTMEMPHY megafunction uses the `afi_doing_rd` signal to enable its capture registers for the expected duration of memory burst.
5. The ALTMEMPHY megafunction issues the first read command to the memory and captures the read data from the memory.
6. The ALTMEMPHY megafunction returns the first data read to the controller after resynchronizing the data to the `phy_clk` domain, by asserting the `afi_rdata_valid` signal when there is valid read data on the `afi_rdata` bus.
7. The controller returns the first read data to the user by asserting the `local_rdata_valid` signal when there is valid read data on the `local_rdata` bus. If the ECC logic is disabled, there is no delay between the `afi_rdata` and the `local_rdata` buses. If there is ECC logic in the controller, there is one or three clock cycles of delay between the `afi_rdata` and `local_rdata` buses.

Half-Rate Write (Burst-Aligned Address)

Figure 17–6. Half-Rate Write Operation for HPC II—Burst-Aligned Address

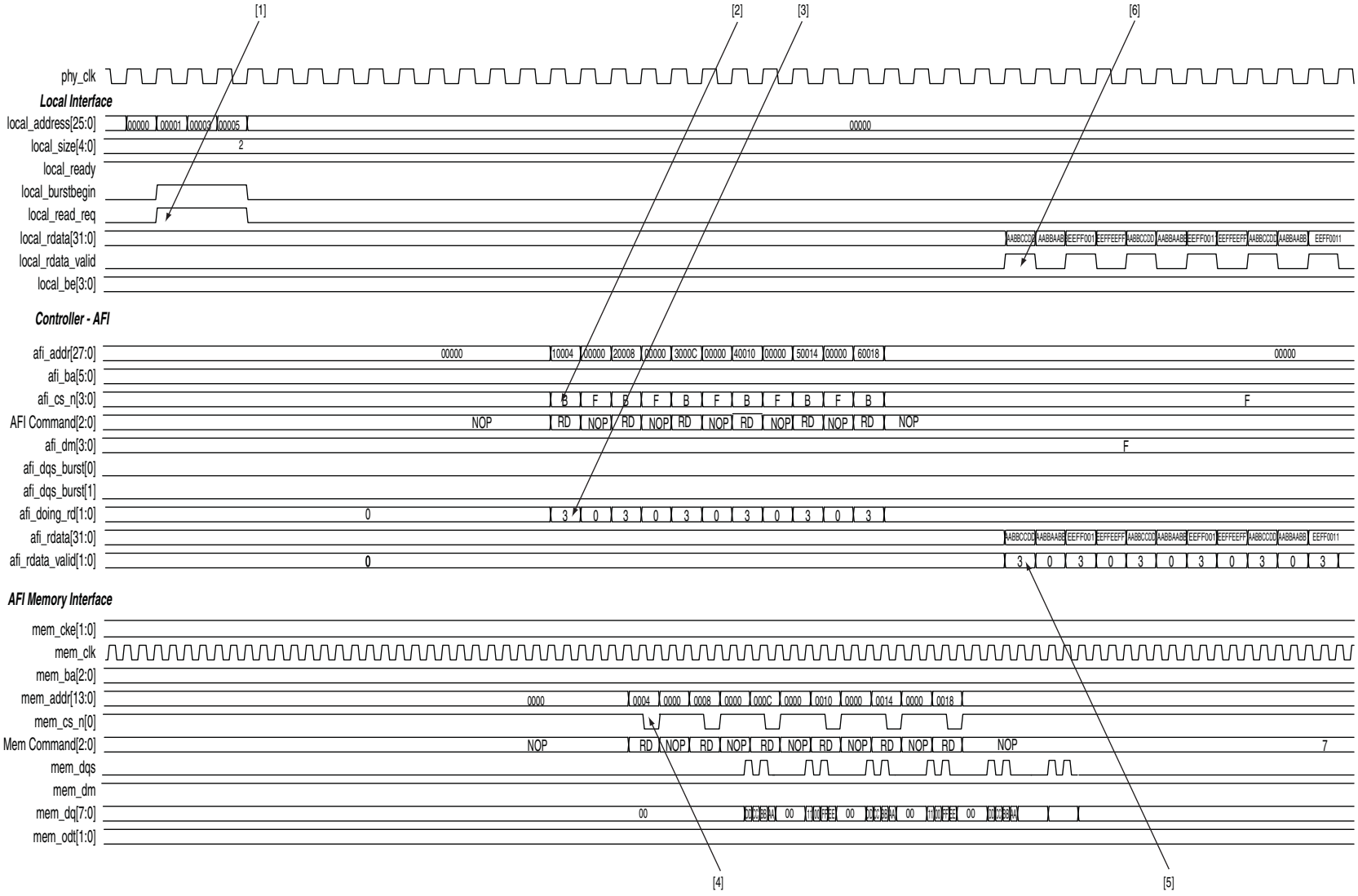


The following sequence corresponds with the numbered items in Figure 17–2:

1. The user logic asserts the first write request to row 0 so that row 0 is open before the next transaction.
2. The user logic asserts a second `local_write_req` signal with size of 2 and address of 0 (col = 0, row = 0, bank = 0, chip = 0). The `local_ready` signal is asserted along with the `local_write_req` signal, which indicates that the controller has accepted this request, and the user logic can request another read or write in the following clock cycle. If the `local_ready` signal was not asserted, the user logic must keep the write request, size, and address signals asserted until the `local_ready` signal is registered high.
3. The controller issues the necessary memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
4. The controller asserts the `afi_wdata_valid` signal to indicate to the ALTMEMPHY megafunction that valid write data and write data masks are present on the inputs to the ALTMEMPHY megafunction.
5. The controller asserts the `afi_dqs_burst` signals to control the timing of the DQS signal that the ALTMEMPHY megafunction issues to the memory.
6. The ALTMEMPHY megafunction issues the write command, and sends the write data and write DQS to the memory.

Half-Rate Read (Non Burst-Aligned Address)

Figure 17–7. Half-Rate Read Operation for HPC II—Non Burst-Aligned Address



The following sequence corresponds with the numbered items in Figure 17–7:

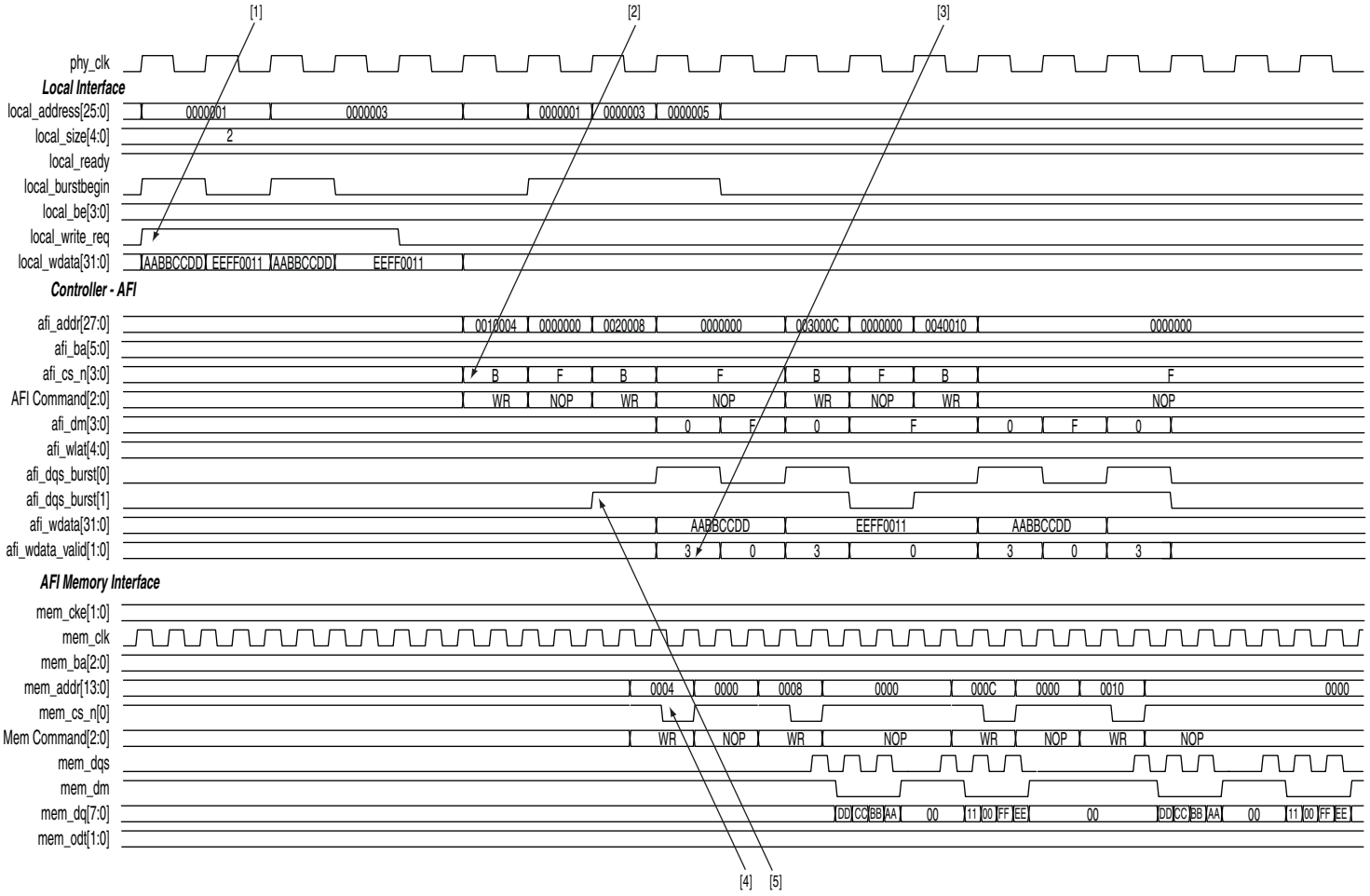
1. The user logic requests the first read by asserting the `local_read_req` signal, and the size and address for this read. In this example, the request is a burst of length of 2 to the local address `0x000001`. This local address is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x0001<<2 = 0x0004

   mem_bank_address = 0x00
   ```

2. The controller issues the first read memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
3. The controller asserts the `afi_doing_rd` signal to indicate to the ALTMEMPHY megafunction the number of clock cycles of read data it must expect for the first read. The ALTMEMPHY megafunction uses the `afi_doing_rd` signal to enable its capture registers for the expected duration of memory burst.
4. The ALTMEMPHY megafunction issues the first read command to the memory and captures the read data from the memory.
5. The ALTMEMPHY megafunction returns the first data read to the controller after resynchronizing the data to the `phy_clk` domain, by asserting the `afi_rdata_valid` signal when there is valid read data on the `afi_rdata` bus.
6. The controller returns the first read data to the user by asserting the `local_rdata_valid` signal when there is valid read data on the `local_rdata` bus. If the ECC logic is disabled, there is no delay between the `afi_rdata` and the `local_rdata` buses. If there is ECC logic in the controller, there is one or three clock cycles of delay between the `afi_rdata` and `local_rdata` buses.

Half-Rate Write (Non Burst-Aligned Address)

Figure 17–8. Half-Rate Write Operation for HPC II—Non Burst-Aligned Address



The following sequence corresponds with the numbered items in Figure 17–8:

1. The user logic asserts the first `local_write_req` signal with a size of 2 and an address of `0x000001`. The `local_ready` signal is asserted along with the `local_write_req` signal, which indicates that the controller has accepted this request, and the user logic can request another read or write in the following clock cycle. If the `local_ready` signal was not asserted, the user logic must keep the write request, size, and address signals asserted until the `local_ready` signal is registered high. The local address 0x000001 is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x000001<<2 = 0x000004

   mem_bank_address = 0x00
   ```

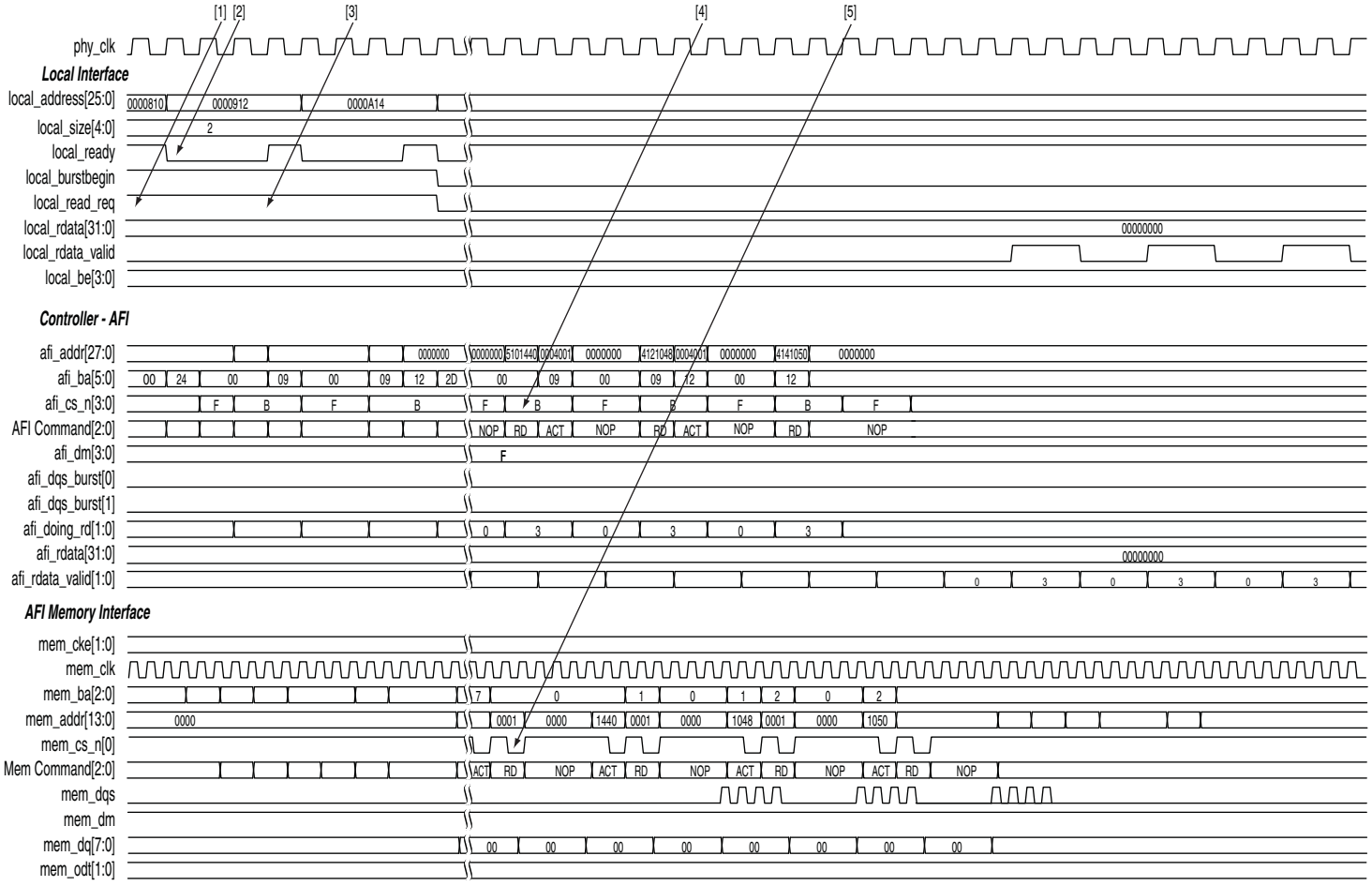
2. The user logic asserts the second `local_write_req` signal with a size of 2 and an address of `0x000003`. The local address `0x000003` is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x000003<<2 = 0x00000C

   mem_bank_address = 0x00
   ```

3. The controller issues the necessary memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
4. The controller asserts the `afi_wdata_valid` signal to indicate to the ALTMEMPHY megafunction that valid write data and write data masks are present on the inputs to the ALTMEMPHY megafunction.
5. The controller asserts the `afi_dqs_burst` signals to control the timing of the DQS signal that the ALTMEMPHY megafunction issues to the memory.
6. The ALTMEMPHY megafunction issues the write command, and sends the write data and write DQS to the memory.
7. The controller generates another write because the first write is to a non-aligned memory address, `0x0004`. The controller performs the second write burst at the memory address of `0x0008`.

Half-Rate Read With Gaps

Figure 17–9. Half-Rate Read Operation for HPC II—With Gaps



The following sequence corresponds with the numbered items in Figure 17–9:

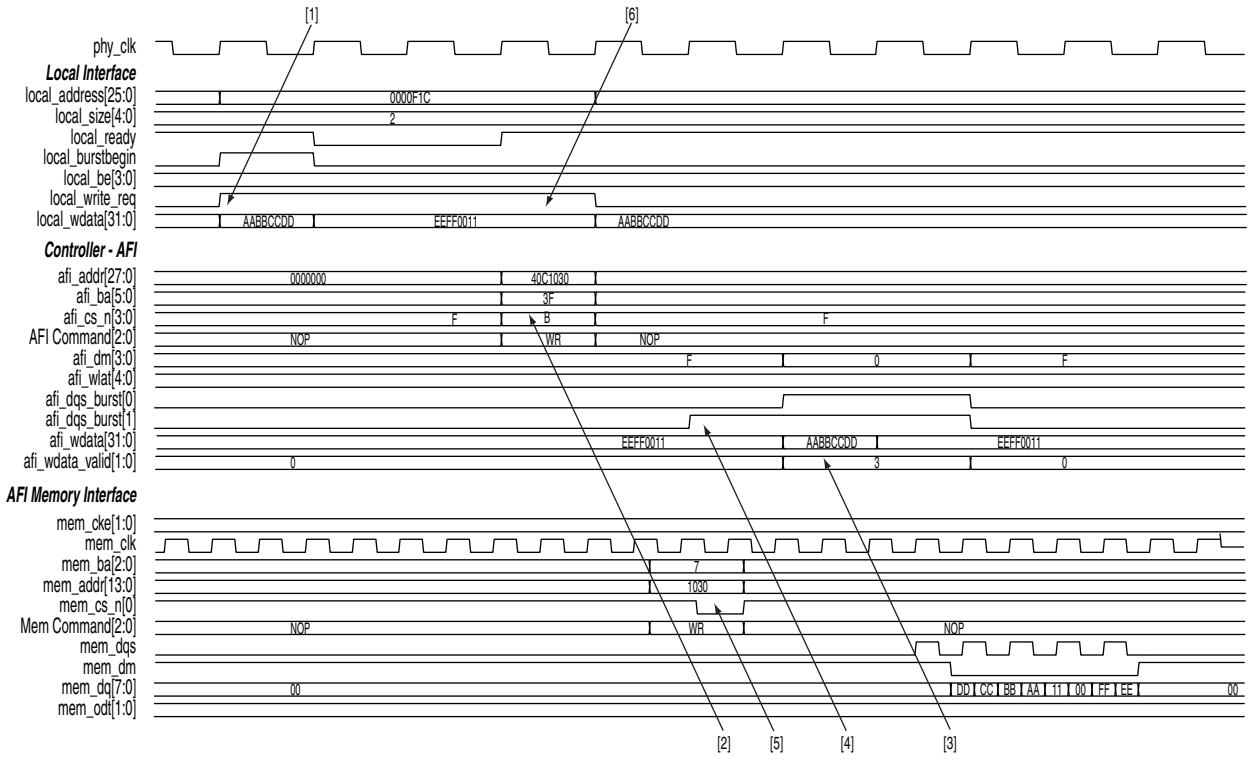
1. The user logic requests the first read by asserting the `local_read_req` signal, and the size and address for this read. In this example, the request is a burst of length of 2 to the local address `0x0000810`. This local address is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0001

   mem_col_address = 0x0010<<2 = 0x0040

   mem_bank_address = 0x00
   ```

2. When the command queue is full, the controller deasserts the `local_ready` signal to indicate that the controller has not accepted the command. The user logic must keep the read request, size, and address signal until the `local_ready` signal is asserted again.
3. The user logic asserts a second `local_read_req` signal with a size of 2 and address of `0x0000912`.
4. The controller issues the first read memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
5. The ALTMEMPHY megafunction issues the read command to the memory and captures the read data from the memory.

Half-Rate Write With Gaps

Figure 17–10. Half-Rate Write Operation for HPC II—With Gaps

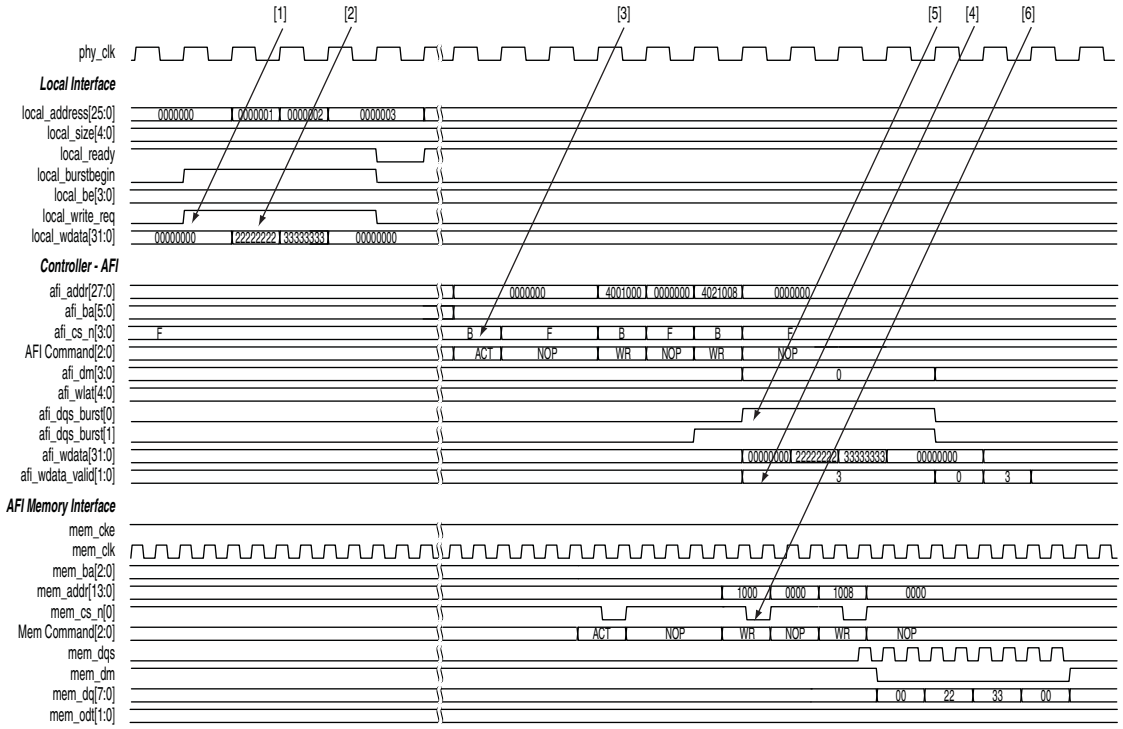


The following sequence corresponds with the numbered items in Figure 17–10:

1. The user logic asserts a `local_write_req` signal with a size of 2 and an address of `0x0000F1C`.
2. The controller issues the necessary memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
3. The controller asserts the `afi_wdata_valid` signal to indicate to the ALTMEMPHY megafunction that valid write data and write data masks are present on the inputs to the ALTMEMPHY megafunction.
4. The controller asserts the `afi_dqs_burst` signals to control the timing of the DQS signal that the ALTMEMPHY megafunction issues to the memory.
5. The ALTMEMPHY megafunction issues the write command, and sends the write data and write DQS to the memory.
6. For transactions with a local size of two, the `local_write_req` and `local_ready` signals must be high for two clock cycles so that all the write data can be transferred to the controller.

Half-Rate Write Operation (Merging Writes)

Figure 17–11. Write Operation for HPC II—Merging Writes



The following sequence corresponds with the numbered items in Figure 17–11:

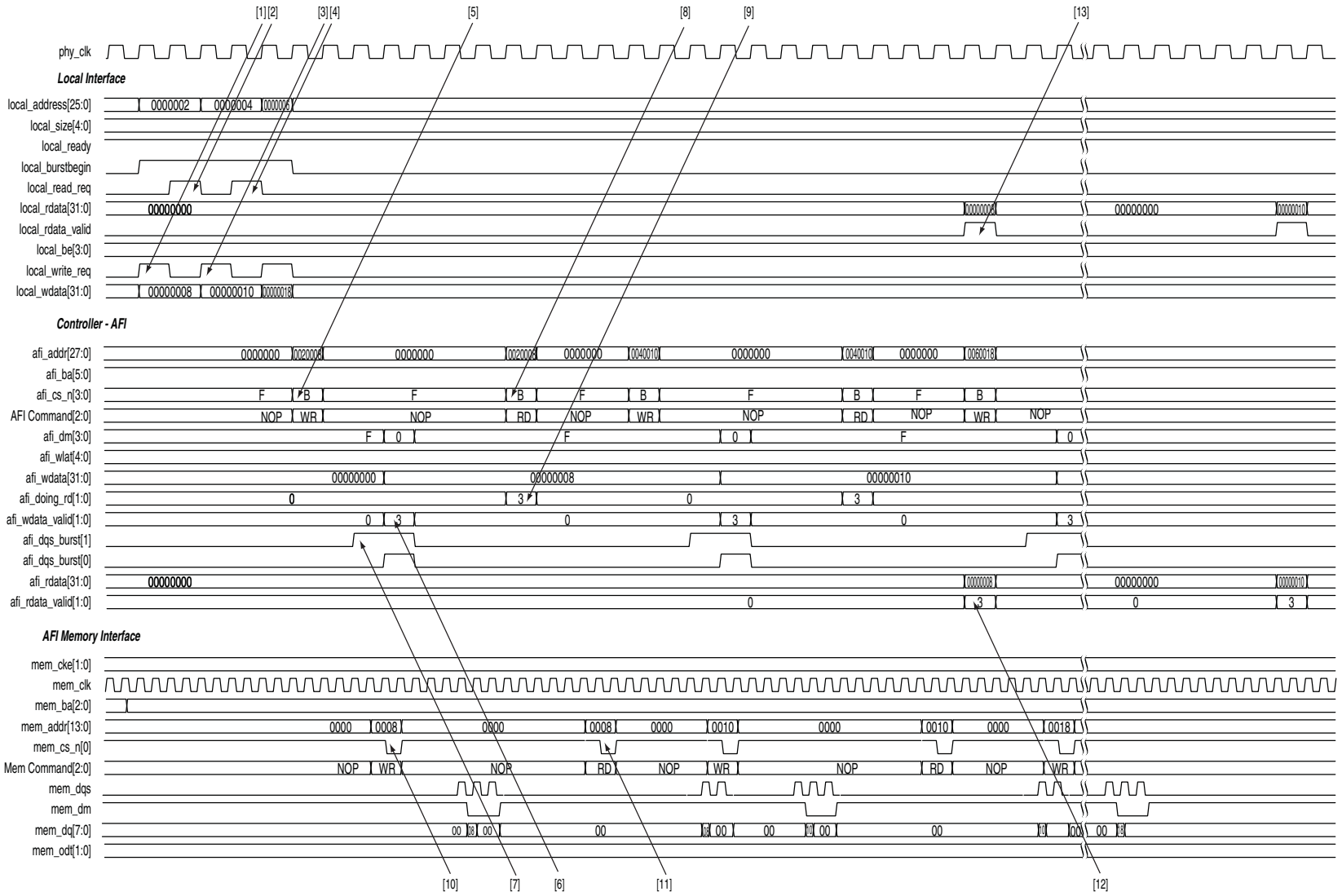
1. The user logic asserts the first `local_write_req` signal with a size of 1 and an address of `0x000000`. The `local_ready` signal is asserted along with the `local_write_req` signal, which indicates that the controller has accepted this request, and the user logic can request another read or write in the following clock cycle. If the `local_ready` signal was not asserted, the user logic must keep the write request, size, and address signals asserted until the `local_ready` signal is registered high. The local address `0x000000` is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x0000<<2 = 0x0000

   mem_bank_address = 0x00
   ```

2. The user logic asserts a second `local_write_req` signal with a size of 1 and address of 1. The `local_ready` signal is asserted along with the `local_write_req` signal, which indicates that the controller has accepted this request. Since the second write request is to a sequential address (same row, same bank, and column increment by 1), this write and the first write can be merged at the memory transaction.
3. The controller issues the necessary memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
4. The controller asserts the `afi_wdata_valid` signal to indicate to the ALTMEMPHY megafunction that valid write data and write data masks are present on the inputs to the ALTMEMPHY megafunction.
5. The controller asserts the `afi_dqs_burst` signals to control the timing of the DQS signal that the ALTMEMPHY megafunction issues to the memory.
6. The ALTMEMPHY megafunction issues the write command, and sends the write data and write DQS to the memory.

Write-Read-Write-Read Operation

Figure 17–12. Write-Read Sequential Operation for HPC II



The following sequence corresponds with the numbered items in Figure 17–12:

1. The user logic requests the first write by asserting the `local_write_req` signal, and the size and address for this write. In this example, the request is a burst length of 1 to a local address `0x000002`. This local address is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x0002<<2 = 0x0008

   mem_bank_address = 0x00
   ```

2. The user logic initiates the first read to the same address as the first write. The request for the read is a burst length of 1. The controller continues to accept commands until the command queue is full. When the command queue is full, the controller deasserts the `local_ready` signal. The starting local address 0x000002 is mapped to the following memory address in half-rate mode:

   ```
   mem_row_address = 0x0000

   mem_col_address = 0x0002<<2 = 0x0008

   mem_bank_address = 0x00
   ```

3. The user logic asserts a second `local_write_req` signal with a size of 1 and address of `0x000004`.
4. The user logic asserts a second `local_read_req` signal with a size of 1 and address of `0x000004`.
5. The controller issues the necessary memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
6. The controller asserts the `afi_wdata_valid` signal to indicate to the ALTMEMPHY megafunction that valid write data and write data masks are present on the inputs to the ALTMEMPHY megafunction.
7. The controller asserts the `afi_dqs_burst` signals to control the timing of the DQS signals that the ALTMEMPHY megafunction issues to the memory.
8. The controller issues the first read memory command and address signals to the ALTMEMPHY megafunction for it to send to the memory device.
9. The controller asserts the `afi_doing_rd` signal to indicate to the ALTMEMPHY megafunction the number of clock cycles of read data it must expect for the first read. The ALTMEMPHY megafunction uses the `afi_doing_rd` signal to enable its capture registers for the expected duration of memory burst.
10. The ALTMEMPHY megafunction issues the write command, and sends the write data and write DQS to the memory.
11. The ALTMEMPHY megafunction issues the first read command to the memory and captures the read data from the memory.
12. The ALTMEMPHY megafunction returns the first data read to the controller after resynchronizing the data to the `phy_clk` domain, by asserting the `afi_rdata_valid` signal when there is valid read data on the `afi_rdata` bus.

13. The controller returns the first read data to the user by asserting the `local_rdata_valid` signal when there is valid read data on the `local_rdata` bus. If the ECC logic is disabled, there is no delay between the `afi_rdata` and the `local_rdata` buses. If there is ECC logic in the controller, there is one or three clock cycles of delay between the `afi_rdata` and `local_rdata` buses.

Document Revision History

Table 17–1 lists the revision history for this document.

Table 17–1. Document Revision History

Date	Version	Changes
November 2012	1.3	Changed chapter number from 15 to 17.
June 2012	1.2	Added Feedback icon.
November 2011	1.1	Consolidated timing diagrams from 11.0 version **DDR and DDR2 SDRAM Controller with ALTMEMPHY IP User Guide** and **DDR3 SDRAM Controller with ALTMEMPHY IP User Guide**.

18. ALTMEMPHY External Memory Interface Debug Toolkit

EMI_RM_016-1.2

This chapter describes a debug toolkit for ALTMEMPHY-based high performance controllers. The debug toolkit uses a JTAG connection to a Windows PC. The debug toolkit supports the following Altera AFI-based IP:

- ALTMEMPHY megafunction
- DDR2 and DDR3 SDRAM High-Performance Controller and High Performance Controller II

The debug toolkit supports all FPGA device families supported by the high-performance controller (HPC) and high-performance controller II (HPC II) and ALTMEMPHY.

The debug toolkit does not support the QDR II and II+ SRAM, RLDRAM II with UniPHY controllers.

The debug toolkit provides detailed information regarding the calibration process. The debug toolkit and the SignalTap II logic analyzer can be run at the same time. However using **Autorun Analysis** in the SignalTap II logic analyzer slows down the JTAG communication with the debug toolkit.

This chapter provides the following information:

- "Debug Toolkit Overview"
- "Install the Debug Toolkit"
- "Modify the Example Top-Level File to use the Debug Toolkit"
- "Use the Debug Toolkit"
- "Interpret the Results"
- "Understand the Checksum and Failure Code"

The debug toolkit provides information on the failures and calibration results that assist and direct the hardware debug process. The debug toolkit does not fix a failing design. Before you use the debug toolkit, refer to Debugging Memory IP in volume 2, section 1, of the *External Memory Interface Handbook*.

Debug Toolkit Overview

The debug TOOLKIT provides the following information:

- Lists the various calibration stages and indicates whether each stage was successful or not.
- States an error code specific to the exact type of calibration failure.

© 2012 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO 9001:2008 Registered

- Provides possible causes for calibration failures.
- Provides graphics to visualize the following parameters:
  - Resync clock phase setup (per pin)
  - Read deskew multipurpose registers (MPR) (prime DQ pins only)
  - Read deskew block training pattern (per pin)
  - Write deskew (per pin)
  - Write leveling report

You can export a **.doom** file from the debug toolkit. This file provides a record of your calibration results and the ALTMEMPHY IOE configuration specific to your system, allowing you to refer to this data offline later.

Install the Debug Toolkit

To install the debug toolkit, follow these steps:

1. Download the debug toolkit, **debug-toolkit.zip,** file from Altera website.
2. Unzip the **debug-toolkit.zip** file.
3. To start the debug toolkit, navigate to the directory where you unzipped the **.zip** file and run **debug-toolkit.exe**.

To install the debug toolkit on a Quartus II production programming PC, follow these steps:

1. On a PC running the Windows OS, copy the **debug-toolkit.zip** file to your project directory or a common programming directory that you also use to program your test platform using a USB-Blaster™ download cable.
2. Unzip the **debug-toolkit.zip** file to either your project folder or a common programming folder.

Modify the Example Top-Level File to use the Debug Toolkit

Before you use the debug toolkit, you must modify your design's example-top-level file, by following these steps:

- Verify the Design
- Regenerate the IP
- Instantiate the JTAG Avalon-MM port in to the Example-Top Level Project
- Add Additional Signals
- Add alt_jtagavalon.v to your Quartus II Project Settings Files List
- Recompile your Quartus II Test Design
- Program Hardware with Debug Enabled .sof

Your design must follow the recommended flow; refer to the *Recommended Design Flow* chapter in volume 1 of the *External Memory Interface Handbook*.

Verify the Design

Ensure your design meets the following conditions:

- The parameters entered into the IP are correct for the memory and data rate.
- The design passes functional simulation.
- The Quartus II project has the correct board trace models specified for the PCB you are using.
- For Cyclone III devices, ensure that the **set t(additional_addresscmd_tpd)** parameter is correctly specified in your **.sdc** file.
- For Arria II GX devices, ensure that the **Address and Command to CK skew** parameter is correctly specified in the **Board Settings** tab of the IP wizard.
- The address and command clock phase is correct, to ensure optimum balanced setup and hold times.
- The Quartus II design successfully closes timing.
- The Quartus II project has the correct pin location assignments for the PCB that you are using.
- The autogenerated IP assignments are correctly applied to the example top-level file.
- The **.sdc** constraint files are correctly applied to the example top-level file.
- The Quartus II settings are correctly applied.
- The R_{UP}/R_{DN} pin locations are correctly specified in the example top-level file if required.
- The SignalTap II logic analyzer is added to the example top-level file.

Before you use the debug toolkit, follow these steps:

1. Edit the example top-level file to enable debugging:
   a. Open the <*variation name*>**.v** or **.vhd** and find the `export_debug_port` private value.

   Do not edit this value in the file <*variation name*>**_phy.v** or **.vhd** file.

   The value is at the bottom of the file:

```
// ========================================================
// DDR3 High Performance Controller Wizard Data
// ==============================
// DO NOT EDIT FOLLOWING DATA
// @Altera, IP Toolbench@
...
...
// Retrieval info: <PRIVATE name = "export_debug_port" value="false"
type="STRING" enable="1" />
```

   b. Edit the `export_debug_port` private value to `true`:

```
// Retrieval info: <PRIVATE name = "export_debug_port" value="true"
type="STRING" enable="1" />
```

Regenerate the IP

To regenerate the IP, follow these steps:

1. Open the MegaWizard Plug In Manager, and select **Edit an existing custom megafunction variation**.
2. Select your modified high-performance controller.
3. Click **Next** to open the IP.
4. Click **Finish** to regenerate the IP.

You now have a version of the design with debug enabled. Seven new ports with the prefix `dbg_*` are added to the controller instance through the design hierarchy up to the <*variation name*>**_example_top.v** or **.vhd**.

Instantiate the JTAG Avalon-MM port in to the Example-Top Level Project

To instantiate the JTAG Avalon-MM port, follow these steps:

1. Declare the following wires in <*variation name*>**_example_top.v** or **.vhd**.

```
wire [12: 0] av_address;
wire av_write_n;
wire [31: 0] av_writedata;
wire av_read_n;
wire av_waitrequest;
wire [31: 0] av_readdata;
```

2. Add the following instances in <*variation name*>**_example_top.v** or **.vhd**.

```
// inst jtag avalon:
alt_jtagavalon alt_jtagavalon(
.clk (phy_clk),
 .rst_n (reset_phy_clk_n),
 .av_address (av_address),
 .av_write_n (av_write_n),
 .av_writedata (av_writedata),
 .av_read_n (av_read_n),
 .av_readdata (av_readdata),
 .av_waitrequest (av_waitrequest)
);
defparam alt_jtagavalon.SLD_NODE_INFO = 203976192;
defparam alt_jtagavalon.ADDR_WIDTH = 13;
defparam alt_jtagavalon.DATA_WIDTH = 32;
defparam alt_jtagavalon.MODE_WIDTH = 3;
```

3. Update the following port connections in the DDR2 or DDR3 SDRAM instance in <*variation name*>**_example_top.v** or **.vhd**:

   a. Locate the PHY or controller instance in the top-level file and locate the following debug port connections:

```
//<< START MEGAWIZARD INSERT WRAPPER_NAME
<variation_name> <variation_name>_inst
(
.dbg_addr (13'b0),
.dbg_cs (1'b0),
.dbg_rd (1'b0),
.dbg_rd_data (dbg_rd_data_sig),
.dbg_waitrequest (dbg_waitrequest_sig),
.dbg_wr (1'b0),
.dbg_wr_data (32'b0),
```

   b. Change the following debug port connections to:

```
//<< START MEGAWIZARD INSERT WRAPPER_NAME
<variation_name> <variation_name>_inst
(
.dbg_addr (av_address),
.dbg_cs (1'b1),
.dbg_rd (~av_read_n),
.dbg_rd_data (av_readdata),
.dbg_waitrequest (av_waitrequest),
.dbg_wr (~av_write_n),
.dbg_wr_data (av_writedata),
```

The debug toolkit is added to your example top-level file.

Add Additional Signals

In addition to the standard SignalTap II signals, you can add the following signals during debug to understand the following situations:

- Where calibration failed:
  - `*ctl_init_fail -phy_inst`
  - `*ctl_init_success -phy_inst`
  - `ctl_cal_fail -phy_inst`
  - `ctl_cal_success -phy_inst`
- How much resynchronization margin is available:
  - `*cal_codvw_phase *DT-phy_inst`
  - `*cal_codvw_size *DT-phy_inst`
  - `*codvw_trk_shift *DT-phy_inst`

- What the read and write latency is calibrated as:
  - `ctl_rlat *DT-phy_inst`
  - `ctl_wlat *DT-phy_inst`
- If the PLL is locked and phase stepping as expected:
  - `Locked -altpll_component`
  - `Phasecounterselect *DT-altpll_component`
  - `Phaseupdown -altpll_component`
  - `Phasestep -altpll_component`
  - `phasedone -altpll_component`
  - `dqs_delay_ctrl_export *DT-phy_inst`

For signals marked with `*DT`, disable trigger enable in the SignalTap II logic analyzer to reduce memory requirement.

Table 18–1 shows sequencer signals that you can also probe using the SignalTap II logic analyzer, to help you understand where calibration failure is occurring. The signals are in the <*vaiation_name*>**_alt_mem_phy_seq.vhd** file.

All signals are active high.

Table 18–1. Sequencer Signals (Part 1 of 2)

Port Name	Description
`Flag_done_timeout`	Calibration stage timeout failure, memory did not respond.
`Flag_ack_timeout`	Sequencer failed to respond.
`state.s_phy_initialise`	PHY initialization Stage: wait for DLL lock and `init_done`.
`state.s_init_dram`	DRAM initialization stage: reset sequence.
`State.s_prog_cal_mrs`	DRAM initialization stage: programming mode registers (once per chip select).
`state.s_write_ihi`	Write internal RAM header initialization.
`state.s_cal`	Calibration required stage.
`state.s_write_btp`	Write block training pattern stage: 00001111.
`state.s_write_mtp`	Write memory training patterns: 00110101.
`state.s_rrp_reset`	Read resynchronization phase reset: PLL initial condition.
`state.s_rrp_sweep`	Read resynchronization phase sweep: sweep PLL phases per chip select.
`state.s_read_mtp`	Read memory training patterns to find correct alignment.
`State.s_rrp_seek`	Read resynchronization phase setup stage: set PLL to center of valid window.
`state.s_rdv`	Read data valid stage.
`state.s_poa`	Postamble calibration stage.
`state.s_was`	Write datapath setup: write data to DRAM so that latency can be determined.
`state.s_adv_rd_lat`	Advertise read latency stage.

Table 18–1. Sequencer Signals (Part 2 of 2)

Port Name	Description
state.s_adv_wr_lat	Advertise write latency stage.
state.s_tracking_setup	Tracking setup stage (first pass to setup mimic window).
state.s_prep_customer_mr_setup	Set custom mode register settings (admin).
state.s_tracking	Tracking stage (mimic path tracking in user mode).
state.s_operational	Calibration success: user mode.
state.s_non_operational	Calibration failed or tracking failed in user mode.
state.s_reset	Reset stage.
dgrb_ctrl.command_err	Error in the data gather read bias block.
dgrb_ctrl.command_result[7..0]	Data gather read block (DGRB) error code.
dgwb_ctrl.command_err	Error in the data gather write bias block.
dgwb_ctrl.command_result[7..0]	Data gather write block (DGWB) error bode.
admin_ctrl.command_err	Error in the admin (DRAM initialization and control) block.
admin_ctrl.command_result[7..0]	Admin block error code.

Add alt_jtagavalon.v to your Quartus II Project Settings Files List

Before you compile your design, you must add the **alt_jtagavalon.v** file to your projects file list. This **alt_jtagavalon.v** file is included with the debug toolkit.

Recompile your Quartus II Test Design

You must compile your modified design to generate a new **.sof** for testing that includes the debug toolkit code. Altera recommends you ensure that this modified design continues to pass timing analysis. Any timing failures should be assessed and corrected before using the debug toolkit.

Program Hardware with Debug Enabled .sof

To program hardware with the debug enabled **.sof**, program the device using the SignalTap II logic analyzer. Then click **Run Analysis** to run once. Typically, the SignalTap II logic analyzer is initially configured to trigger on the signal `test_complete`, which is fine for working designs.

For designs that are failing calibration, Altera recommends modifying the trigger based on the observed results. Combine this SignalTap II trigger fault isolation activity with use of the debug toolkit. For example:

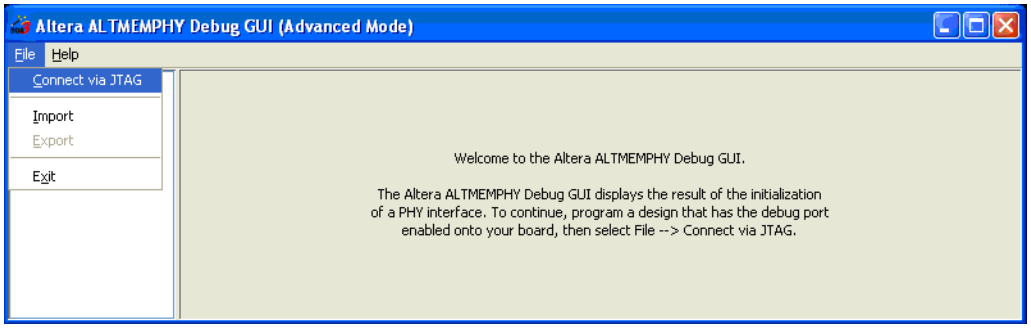
1. Initially trigger on `test_complete`, if the interface works first time.
2. Trigger on `cal_fail`, if the PHY is failing calibration.
3. Trigger on the same `state.s_*` or error code that is reported as the calibration failure point in the debug toolkit.
4. Trigger on `init_fail`, if the memory is failing to initialize.
5. Trigger on `pll_locked`, if the PLL is operating incorrectly.

Use the Debug Toolkit

To use the debug toolkit, follow these steps:

1. Double-click **debug-toolkit.exe**.
2. On the File menu, click **Connect via JTAG** (Figure 18–1).

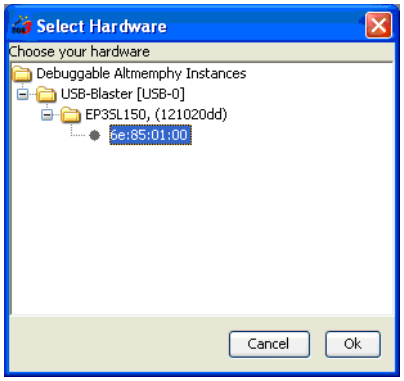
Figure 18–1. Connect to JTAG



3. Navigate down the hierarchy and click on the Avalon-MM JTAG node (Figure 18–2).

If you encounter connection problems, Altera recommends that you have only a single USB-Blaster™ download cable programming adaptor connected to your PC.

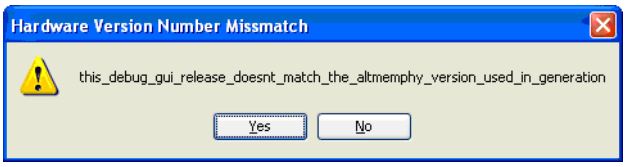
Figure 18–2. Select Hardware



4. If you receive a prompt stating the following message, verify you have the latest debug toolkit, and click **Yes** (Figure 18–3).

```
Hardware Version Number Mismatch -
this_debug_GUI_release_doesnt_match_the_altmemphy_version_used_in_g
eneration
```

Figure 18–3. Hardware Mismatch



Interpret the Results

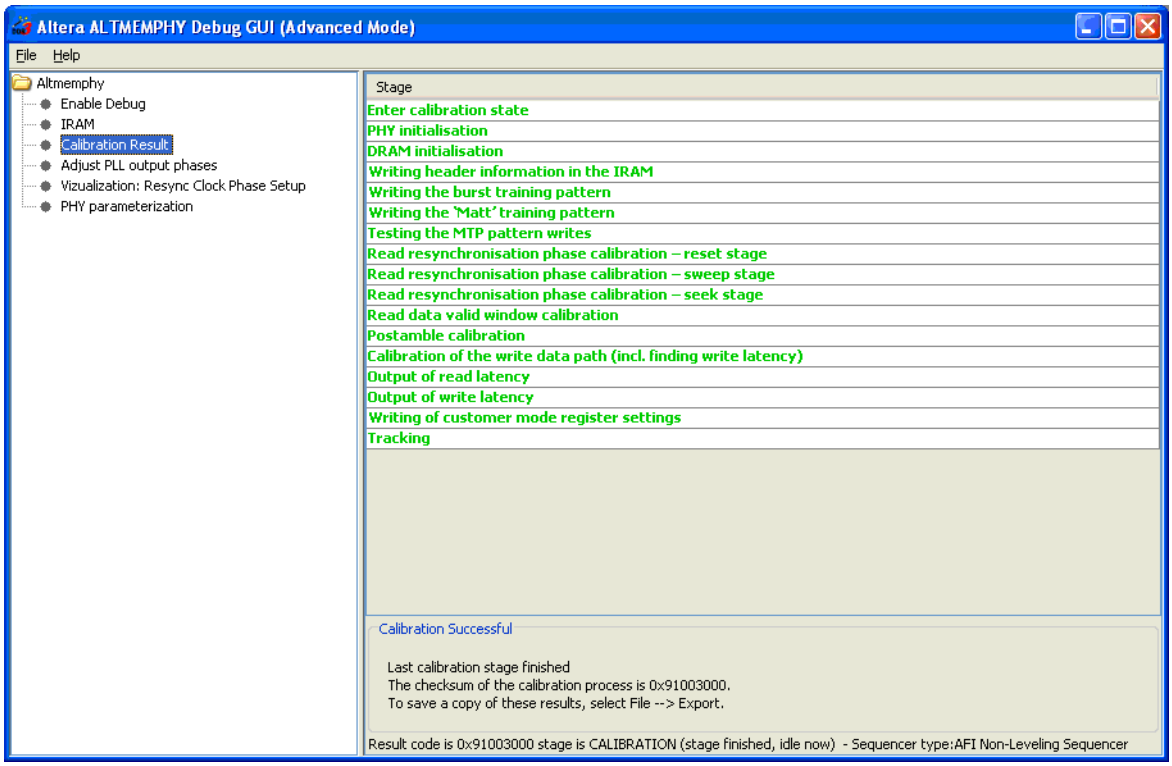
This topic discusses:

- Calibration Successful
- Calibration Fails

Calibration Successful

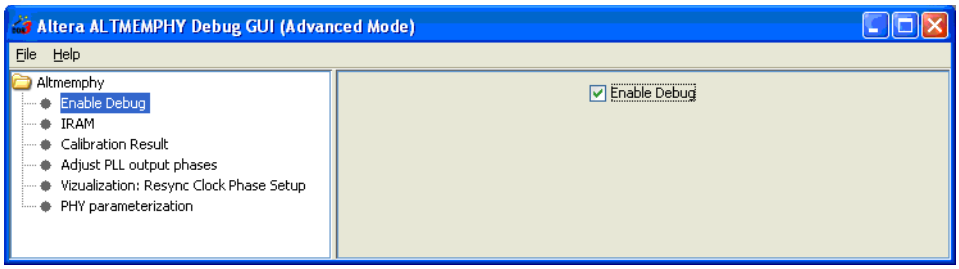
If calibration is successful, you see the following screen (Figure 18–4).

Figure 18–4. Calibration Successful



For optimum operation of the debug toolkit, ensure that you turn on **Enable Debug** (Figure 18–5).

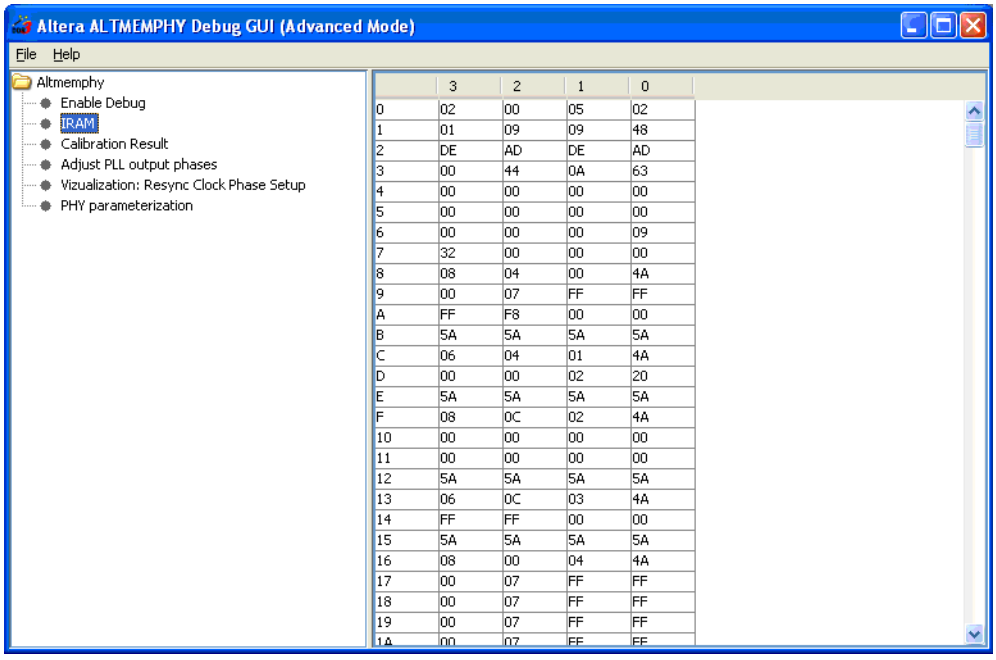
Figure 18–5. Enable Debug



Click **internal RAM** to display the calibration memory results (Figure 18–6).

This setting is not typically used.

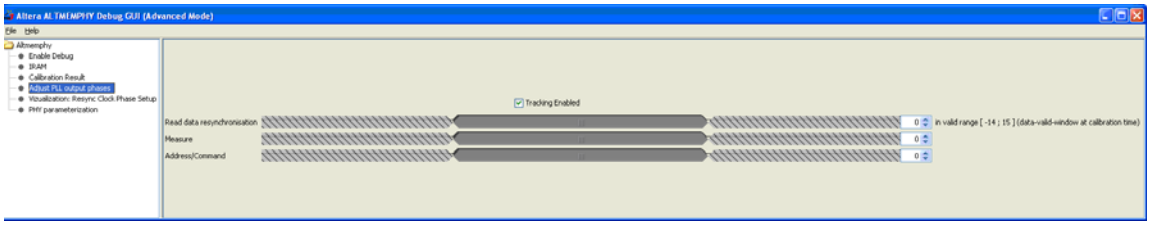
Figure 18–6. Calibration Memory Results



The debug toolkit can dynamically alter the PLL clock phases (Figure 18–7).

This setting is not typically used.

Figure 18–7. Altering PLL Clock Phases



The debug toolkit states the number of resynchronization clock phase steps that are valid at calibration time. For example, the resynchronization window size in PLL phase steps at calibration in Figure 18–7 is 26 PLL phase steps wide.

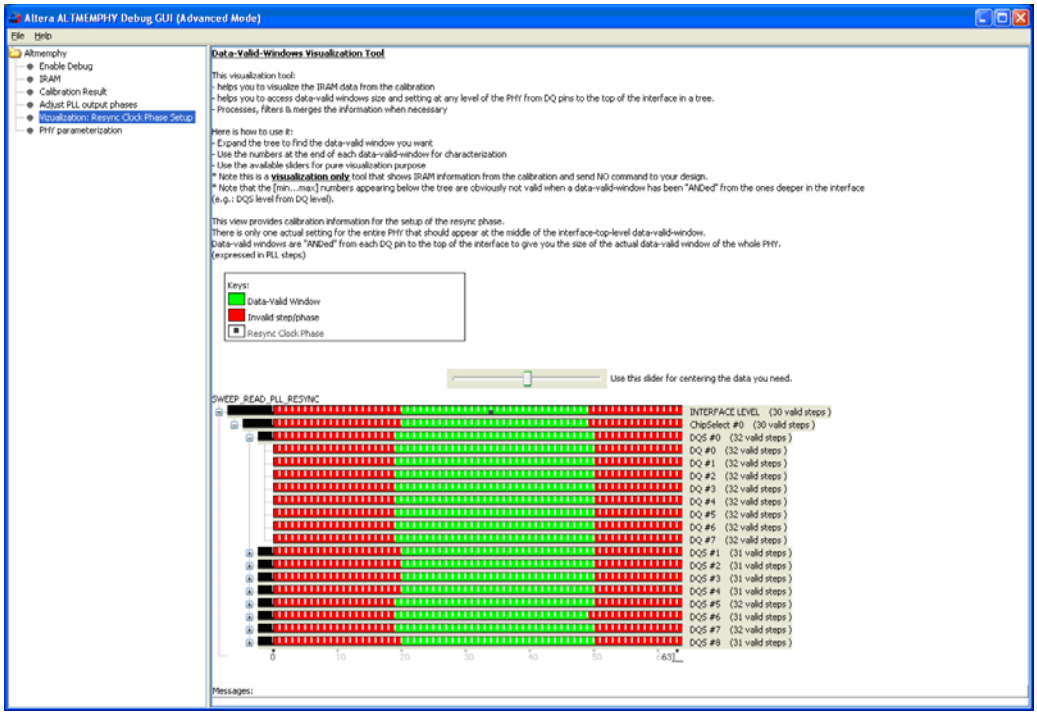
The address and command phase sweep is limited to either address and command pin margins or address and command core-to-I/O transfer margins.

In Figure 18–7, moving the slider to the left increases the value, while moving the slider to the right decreases the value.

Click **Visualization: resync clock phase setup** (Figure 18–8) to show the PHY resynchronization pass and fail results in an expandable tree structure:

- For the whole interface including the chosen phase (black dot)
- On a DQS group basis
- On a per DQ pin basis

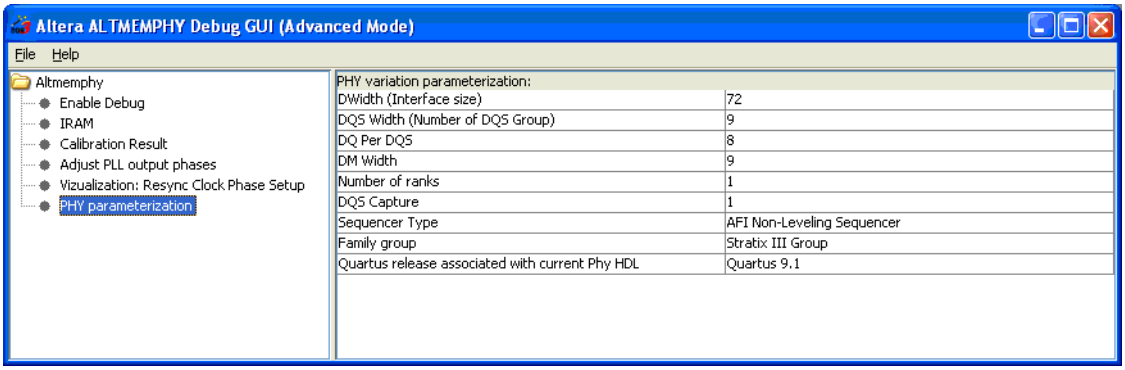
Figure 18–8. Visualization



The debug toolkit additionally states the number of passing phase steps that it finds during calibration. Thus to calculate the resynchronization margin in this design, the resynchronization clock (C6) from the PLL has a phase-shift step resolution of 78.12 ps or 5.62 degrees. So 30 valid steps means that the window size = 2.343 ns or 168.6 degrees.

Click **PHY parameterization** (Figure 18–9), to show the exact calibration configuration of the generated IP.

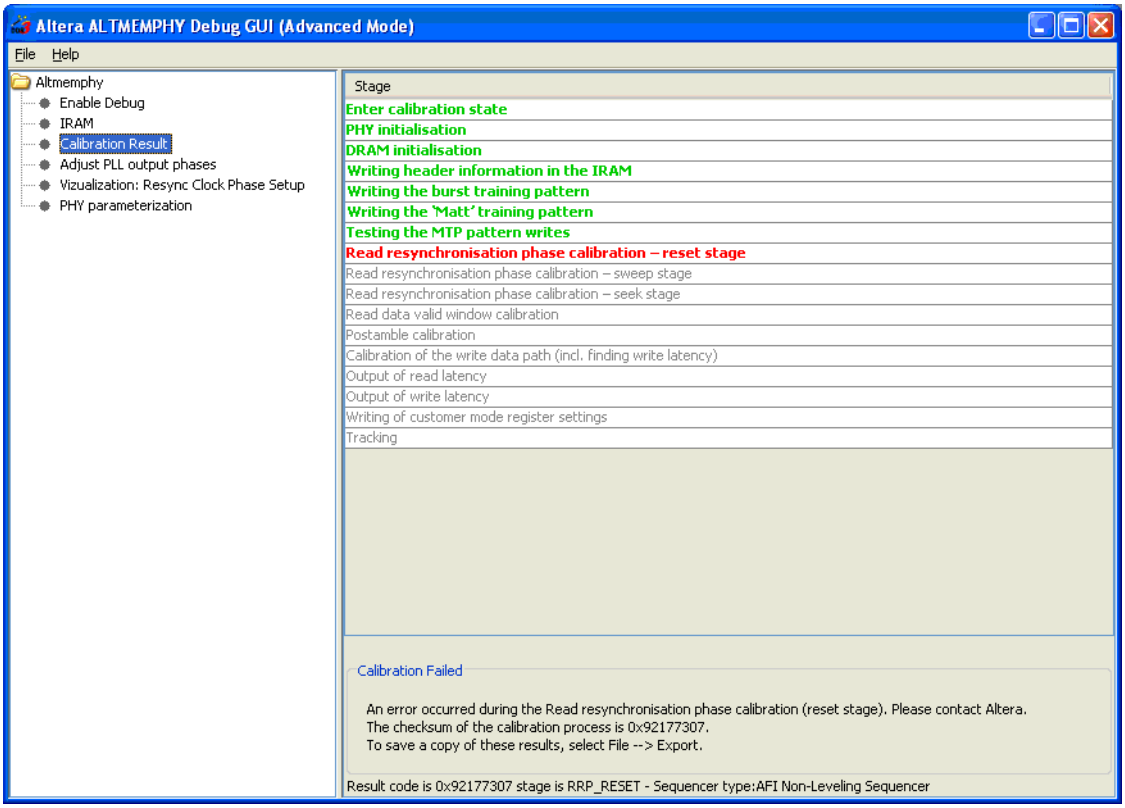
Figure 18–9. PHY Parameterization



Calibration Fails

If calibration fails, you see the following screen (Figure 18–10).

Figure 18–10. Calibration Fails



The stage at which calibration fails is highlighted in red; the stages that have successfully passed are in green. When possible, the debug toolkit also provides a possible cause for the failure code.

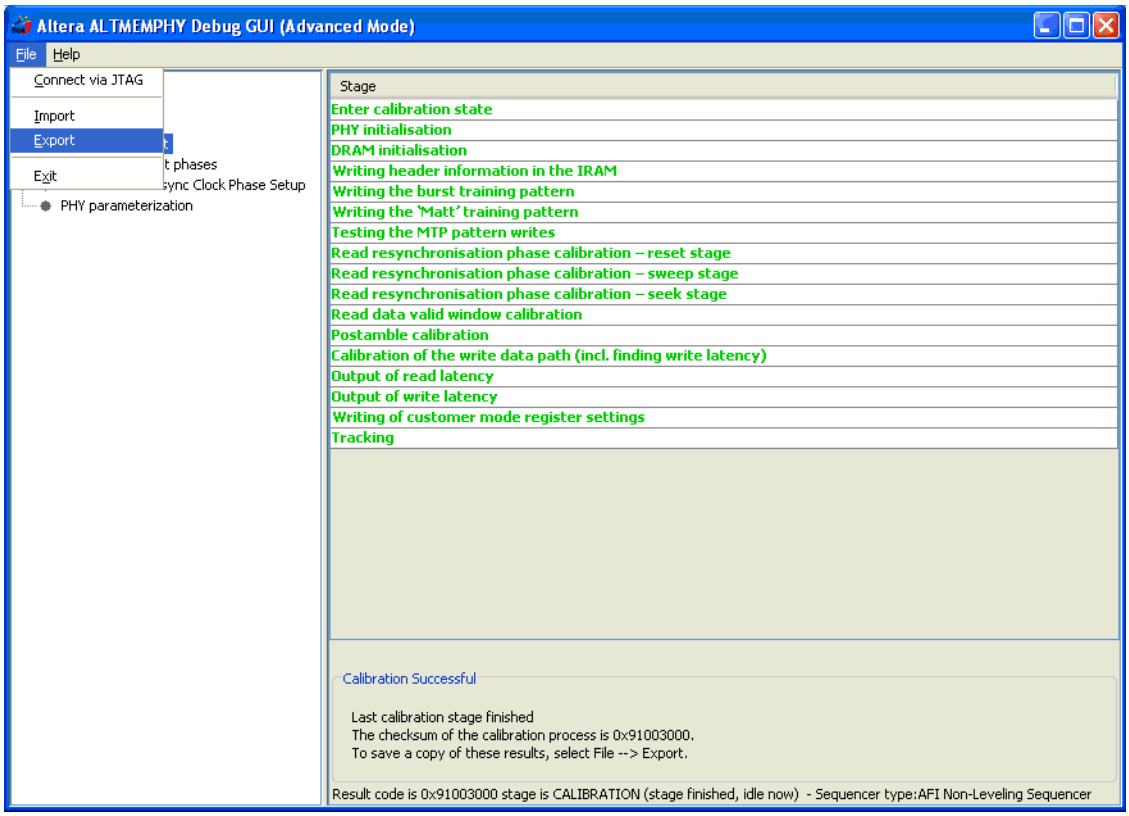
This failure code is: `0x92177307`, for more information on failure codes, refer to "Understand the Checksum and Failure Code" on page 18–15.

Save the Calibration Results

Often the calibration failure stage, the reported suggested failure cause, or the combined debug toolkit result and waveforms viewed in the SignalTap II logic analyzer provide enough detail to resolve the failure directly. However, you may wish to save your calibration results, so that you can refer to them later.

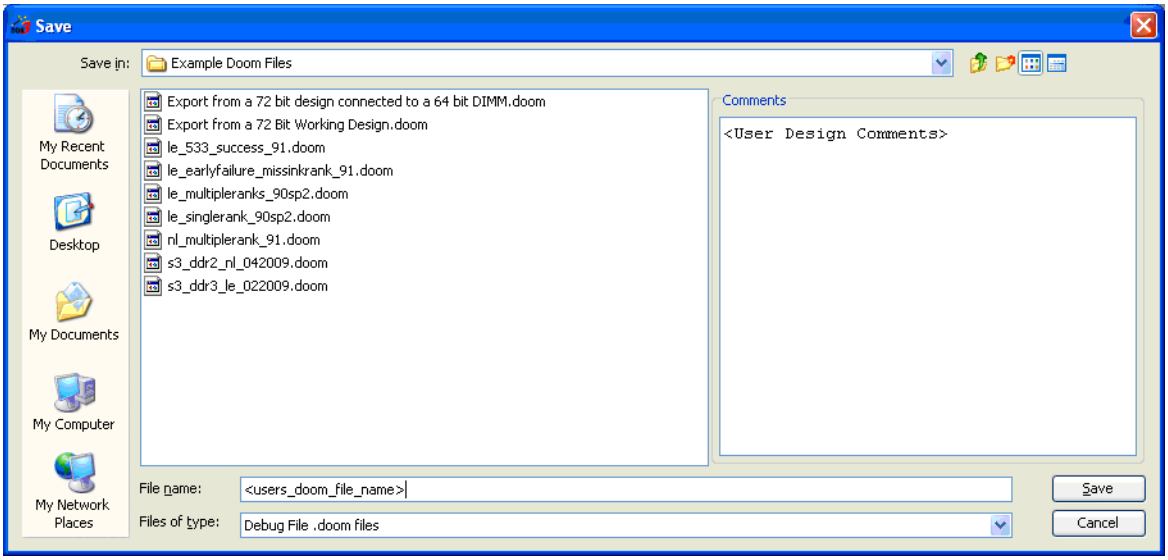
With your calibration process results still displayed on the File menu, click **Export** (Figure 18–11).

Figure 18–11. Export



In the **Save** dialog box (Figure 18–12), specify a file name and any comments to help with the identification and understanding of the controller configuration that you have evaluated, and details on what you may have tested.

Figure 18–12. Save



You can save this **.doom** file with a Quartus II archive (**.qar)** file of the test design, and a copy of the captured SignalTap II waveform files, as a single design archive. This archive provides a record of your calibration results and the ALTMEMPHY IOE configuration specific to your system, so you can refer to this data at a later date.

Understand the Checksum and Failure Code

The debug toolkit checksum provides a direct correlation to the exact stage that calibration failed and the error code for that failure.

For example, the hexadecimal code in the format `0xAABBCCDD` represents the full 32-bit contents of the calibration status register.

The code and the subcode have the following definitions:

- The code or calibration stage is the first byte (`DD`) or `[7..0]`
- The subcode or error code is the third byte (`BB`) or `[23..16]`

Take the hexadecimal value of each code and convert that to decimal. When you have these two numbers in decimal format, you can open the **failuremessages_nl.csv** spreadsheet file and look up the likely causes of your calibration failure.

In a passing interface, these two numbers are zero.

Table 18–2 shows the codes that correspond to the indicated calibration stages.

Table 18–2. Calibration Stages

Stage	Code
Enter calibration state	0
PHY initialization	1
DRAM initialization	2
Writing header information in the internal RAM	3
Writing the burst training pattern	4
Writing more training patterns	5
Testing more training pattern writes	6
Read resynchronization phase calibration—reset stage	7
Read resynchronization phase calibration—sweep stage	8
Read resynchronization phase calibration—seek stage	9
Read data valid window calibration	10
Postamble calibration	11
Calibration of the write datapath (including finding write latency)	12
Output of read latency	13
Output of write latency	14
Writing of customer mode register settings	15
Tracking	16

You can find the same information from the SignalTap II logic analyzer if the `*_ctrl.command_err`, `*_ctrl.command_result *` and `state.s_*` signals are added. The command error and state signals identify within which calibration stage the interface fails. The corresponding command result then includes the same information as the subcode.

For more information on the stages of calibration, refer to "ALTMEMPHY Calibration Stages" in section 1, chapter 2, of this volume.

Document Revision History

Table 18–3 lists the revision history for this document.

Table 18–3. Document Revision History

Date	Version	Changes
November 2012	1.2	Changed chapter number from 16 to 18.
June 2012	1.1	Added Feedback icon.
November 2011	1.0	Harvested 11.0 **Debug Toolkit for DDR2 and DDR3 SDRAM Controllers with ALTMEMPHY IP** content.