External Memory Interface Handbook
Volume 4: Simulation, Timing Analysis, and Debugging
External Memory Interface Handbook Volume 4

Section I. Simulation
## Chapter 1. Simulation Overview

## Chapter 2. Simulation Walkthrough with UniPHY IP
- Before Simulating .......................................................... 2–2
- Preparing the Vendor Memory Model ................................. 2–2
- Simulating Using NativeLink .......................................... 2–5
- IP Functional Simulations .............................................. 2–5
  - Verilog HDL ............................................................... 2–6
  - VHDL ................................................................. 2–7
- Abstract PHY ............................................................. 2–8
- Single-Language VHDL Simulation Support ...................... 2–9
- Simulation Issues ......................................................... 2–10

## Chapter 3. Simulation Walkthrough with ALTMEMPHY IP
- Before Simulating .......................................................... 3–2
- Preparing the Vendor Memory Model ................................. 3–2
- Simulating Using NativeLink .......................................... 3–4
- IP Functional Simulations .............................................. 3–6
  - VHDL ................................................................. 3–6
  - Verilog HDL ............................................................... 3–7
- Simulation Tips and Issues ............................................ 3–9
  - Tips ........................................................................... 3–9
  - DDR3 SDRAM (without Leveling) Warnings and Errors .............. 3–9

## Additional Information
- Document Revision History ........................................ Info–1
- How to Contact Altera .................................................. Info–1
- Typographic Conventions ............................................. Info–2
This chapter covers 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
- 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

The simulator should be any Altera-supported VHDL or Verilog HDL simulator. This example design already contains these blocks.

For more information about single language VHDL simulation support, refer to “Simulation Walkthrough with UniPHY IP” chapter.

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 models 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, Altera does not currently provide support for Denali models. All memory vendor simulation models that you use to simulate Altera memory IP must be JEDEC compliant.

With the example testbench, the following simulations 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 one bit per group before entering user mode to save calibration time.

- Skip calibration—Provides the fastest simulation. It loads the settings calculated from the memory configuration and enters user mode.
Table 1–1 lists the typical simulation times implemented using UniPHY IP.

These simulation times are estimates based on average run times of a few designs. The simulation times for your design may vary depending on the memory interface specifications, simulator, or the system you are using.

Table 1–1. Typical Simulation Times Using UniPHY IP

<table>
<thead>
<tr>
<th>Calibration Mode/Run Time (Note 1)</th>
<th>Simulation Time</th>
<th>Small Interface</th>
<th>Large Interface (×72 Quad Rank)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Full (2)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ Full calibration</td>
<td></td>
<td>10 minutes</td>
<td>~ 1 day</td>
</tr>
<tr>
<td>▪ Includes all phase/delay sweeps and centering</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Quick (2)</strong></td>
<td></td>
<td>3 minutes</td>
<td>4 hours</td>
</tr>
<tr>
<td>▪ Scaled down calibration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ Calibrate one pin</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Skip</strong></td>
<td></td>
<td>3 minutes</td>
<td>20 minutes</td>
</tr>
<tr>
<td>▪ Skip all calibration, jump to user mode</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ Preload calculated settings</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note to Table 1–1:
(1) Uses one loop of driver test. One loop of driver is approximately 600 read or write requests, with burst length up to 64.
(2) Not applicable for Stratix V devices.

If you are using Stratix V devices, you can only use quick initialization and skip calibration in auto-calibration mode.

For more information about steps to follow before simulating, modifying the vendor memory model, and using the Nativelink flow to perform simulation for both ALTMEMPHY and UniPHY IPs, refer to the “Simulation Walkthrough with UniPHY IP” and “Simulation Walkthrough with ALTMEMPHY IP” chapters.
For high-performance memory controllers with UniPHY IP, you can simulate the example top-level file with the MegaWizard-generated IP functional simulation models. The MegaWizard™ Plug-In generates a Verilog HDL testbench for the example top-level file, which is in the \\
<variation_name>_example_design\simulation 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 for Verilog HDL UniPHY design. This feature does not support the VHDL flow.

The UniPHY always generates the example design in Verilog HDL.

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 only supports functional simulation; it does not support gate-level simulation.

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
  - arria1i_atoms, strati1xv_atoms, strati1xiii_atoms, arria1igz_atoms, strati1xv_atoms (device dependent)
  - altera_insim

You require the altera_insim library to simulate designs targeting all device families.
Unless you are using NativeLink, if you are targeting Stratix IV devices, you need both the Stratix IV and Stratix III files (stratixiv_atoms and stratixiii_atoms) to simulate and if you are targeting Stratix V devices, you need both the Stratix V and Stratix IV files (stratixv_atoms and stratixiv_atoms) to simulate.

- UniPHY module
- User logic, or a driver for the PHY
- Testbench
- Vendor memory model

Prepared 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. 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 file, \\
   \<variation_name>_example_sim_<variation_name>_example_sim.v, located in the \\
   \<variation_name>_example_design\simulation\<variation_name>_example_sim\submodules 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.

6. Instantiate the downloaded memory model and connect its signals to the rest of the design.

7. 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.

8. Save the testbench file.

The original instantiation may be similar to the following code:

```vhdl
class alt_mem_if_ddr2_mem_model_top_mem_if_dm_pins_en_mem_if_dqsn_en is
  @(#( '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
  }
end class;
```
Simulating Using NativeLink

To set up simulation in the Quartus® II software using NativeLink, follow these steps:

1. Create a custom variation using the MegaWizard Plug-in Manager.
2. Create a custom variation with an IP functional simulation model.
3. Open the simulation example project.
   a. On the File menu, click Open.
   b. Browse to `<variation name>_example` and click Open.
   c. Select `<variation_name>_example_design/simulation/<variation_name>_example_sim.qpf`
4. To elaborate your design, on the Processing menu, point to Start and click Start Analysis & Elaboration.

5. 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.

6. 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.

7. On the Tools menu, point to Run EDA Simulation Tool and click EDA 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 before filing a service request.

**IP Functional Simulations**

This topic discusses VHDL and Verilog HDL simulations with IP functional simulation models.

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**

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`
   - `altera_lnsim_ver`
   - `<device name>_hssi_ver`
   - `simulation_files`
3. Compile the files into the appropriate library as shown in Table 2–1 on page 2–6.

   If you are using ModelSim Altera Edition (ModelSim-AE), skip this step because ModelSim-AE comes with precompiled versions of the libraries.

<table>
<thead>
<tr>
<th>Library</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>altera_mf_ver</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_mf.v</code></td>
</tr>
<tr>
<td>lpm_ver</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\220model.v</code></td>
</tr>
<tr>
<td>sgate_ver</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\sgate.v</code></td>
</tr>
<tr>
<td>&lt;device name&gt;_ver</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\&lt;device name&gt;\atoms.v</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\&lt;device name&gt;\atoms_ncrypt.v</code> (1)</td>
</tr>
<tr>
<td>altera_ver</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_primitives.v</code></td>
</tr>
<tr>
<td>altera_lnsim_ver</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_lnsim.sv</code></td>
</tr>
<tr>
<td>&lt;device name&gt;_hssi_ver</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\&lt;device name&gt;\hssi_atoms.v</code> (2)</td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\mentor\&lt;device name&gt;\hssi_atoms_ncrypt.v</code> (1)</td>
</tr>
<tr>
<td>example_design_sim_files</td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\example_design\example_sim_tb.v</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\example_design\example_sim.v</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\example_design\example_sim\submodules\*.v</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\example_design\example_sim\submodules\*.sv</code></td>
</tr>
<tr>
<td>sim_fileset (4)</td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\sim\alt_mem_ddrx_mm_st_converter\*.v*.sv</code> (5)</td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\sim\altera_mem_if_nextgen\controller\*.v*.sv</code> (5)</td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\sim\altera_mem_if\controller_core\*.v*.sv</code> (5)</td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\sim\altera_mem_if\controller\*.v*.sv</code> (6)</td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;&lt;variation name&gt;\sim\&lt;variation name&gt;.v</code></td>
</tr>
</tbody>
</table>

Notes to Table 2–1:

(1) Applicable only for Stratix V devices.
(2) Applicable only for Arria II GX, Arria II GZ, Stratix IV, and Stratix V devices.
(3) Only use these files in your simulation if you are using the Altera-provided design example.
(4) Only use these files in your simulation if you are using your own design.
(5) Applicable only for DDR2 and DDR3 SDRAM controllers with UniPHY.
(6) Applicable only for QDR II and QDR II+ SRAM, and RLDRAM II controllers with UniPHY.

4. Configure your simulator to use transport delays, a timestep of picoseconds, and to include all the libraries in Table 2–1.

5. Compile the testbench file.
6. To load the testbench in your simulator, type the following command:

```
vsim -t 1ps -L work -L <lib1>_ver -L <lib2>_ver -L <libn>_ver work. 
<testbench module name>
```

**VHDL**

You can only simulate the UniPHY VHDL simulation fileset with ModelSim. Because the simulation fileset has several limitations, you should use this option only when your simulator does not support mixed HDL language (a combination of Verilog and VHDL language).

The UniPHY VHDL fileset is specifically for ModelSim customers who use VHDL exclusively and who do not have a mixed-language (VHDL + Verilog) simulation license. All other customers should either select the Verilog language option during generation, or simulate using the synthesis fileset. Because some of the files in the VHDL fileset are encrypted specifically for ModelSim, you cannot use these files with other simulators. In addition, simulation with the VHDL fileset is slower and harder to debug, and the generation time of the VHDL UniPHY IP may be up to two times slower than the generation time of the Verilog UniPHY IP. The UniPHY IP only generates the example design in Verilog HDL, and not in VHDL. Only the VHDL simulation fileset is available for VHDL simulation.

For all of these reasons you should only use VHDL when mixed-language simulation is not an option and only with ModelSim.

The UniPHY IP VHDL simulation fileset consists of the following types of files: simgen-generated VHDL files, IEEE Encrypted Verilog HDL files, and regular 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.

The UniPHY IP does not provide any testbench or memory model for VHDL simulations, so you have to provide your own testbench and memory model to perform simulations.

To use your own testbench, follow these steps:

1. Combine the simulation fileset with your own design files.
2. Compile all the respective Verilog HDL (Table 2–1) and VHDL (Table 2–2) files with the appropriate compiler.
3. Compile the Verilog HDL and VHDL libraries, if you are not using ModelSim-AE.
4. Compile the Verilog HDL libraries under the `<lib>_ver` name and the VHDL libraries under the `<lib>` name.
5. Link all the files by typing `-L <lib>_ver` for each Verilog HDL library.

You do not need to specify the VHDL libraries because the VHDL standard uses a built-in method to find the VHDL versions of the libraries.
Table 2–2 shows the VHDL files to be compiled into the appropriate libraries.

### Table 2–2. Files to Compile—VHDL IP Functional Simulation Models

<table>
<thead>
<tr>
<th>Library</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>altera_mf</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_mf\altera_mf_components.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_mf.vhd</code></td>
</tr>
<tr>
<td>lpm</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\220pack.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\220model.vhd</code></td>
</tr>
<tr>
<td>sgate</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\sgate_pack.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\sgate.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\&lt;device name&gt;_atoms.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\&lt;device name&gt;_components.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_primitives_components.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_syn_attributes.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_primitives.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_europa_support_lib.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_standard_functions.vhd</code></td>
</tr>
<tr>
<td>altera</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\altera_insim_components.vhd</code></td>
</tr>
<tr>
<td>hssi</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\&lt;device name&gt;_hssi_components.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS ROOTDIR&gt;eda\sim_lib\&lt;device name&gt;_hssi_atoms.vhd</code> (1)</td>
</tr>
<tr>
<td>sim_fileset</td>
<td><code>&lt;QUARTUS ROOTDIR&gt;libraries\vhdl\altera\altera_europa_support_lib.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;\&lt;variation name&gt;\sim\&lt;variation name&gt;_vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;\&lt;variation name\&gt;\sim\alt_mem_ddrx_mm_st_converter\*.v*.sv</code></td>
</tr>
<tr>
<td></td>
<td><code>.vho*.vhd</code> (3)</td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;\&lt;variation name&gt;\sim\altera_mem_if_nextgen\&lt;variation name&gt;_controller\*.v*.sv</code></td>
</tr>
<tr>
<td></td>
<td><code>.vho*.vhd</code> (3)</td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;\&lt;variation name&gt;\sim\altera_mem_if\&lt;variation name&gt;_controller_core\*.v*.sv</code></td>
</tr>
<tr>
<td></td>
<td><code>.vho*.vhd</code> (3)</td>
</tr>
<tr>
<td></td>
<td><code>&lt;project directory&gt;\&lt;variation name&gt;\sim\altera_mem_if\&lt;variation name&gt;_controller\*.v*.sv</code></td>
</tr>
<tr>
<td></td>
<td><code>.vho*.vhd</code> (4)</td>
</tr>
</tbody>
</table>

**Note to Table 2–2:**

1. Applicable only for Arria II GX, Arria II GZ, Stratix IV, and Stratix V devices.
2. Applicable only for Stratix V devices.
3. Applicable only for DDR2 and DDR3 SDRAM controllers with UniPHY.
4. Applicable only for QDR II and QDR II+ SRAM, and RLDRAM II controllers with UniPHY.
Abstract PHY

In the Quartus II software version 11.0, 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. 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.

To use the regular simulation models for Verilog HDL language, follow these steps:

1. In a text editor, type the following command to create a Verilog HDL header file.
   ```
   define ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL 0
   ```

2. Name the header file as `uniphy_fast_sim_parameter.vh` and save it in the `<project_directory>\<variation name>_example_design\simulation\<variation name>_example_sim\directory`.

3. Add the following comment line to `<variation name>_example_sim_<variation name>_example_sim_e0_if0_p0_memphy_top.sv`:
   ```
   include "uniphy_fast_sim_parameter.vh"
   ```

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.

Simulation Issues

When you simulate an example design in ModelSim using Nativelink, you may 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'.

# ** 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'.

# ** 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'.

# ** Warning: (vsim-3017)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_alt_ddrx_bank_timer_wrapper.v(1191)
: [TFMPC] - Too few port connections. Expected 127, found 126.

# ** Warning: (vsim-3722)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_alt_ddrx_bank_timer_wrapper.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_wrapper.v(1344)
: [PCDPC] - Port size (5 or 5) does not match connection size (1) for port 'wr_to_rd_to_pch_all'.

# ** Warning: (vsim-8598) Non-positive replication multiplier inside concat. Replication will be ignored

Warning-[OSPA-N] Overriding same parameter again
/p/eda/acad/altera/quartusII/10.1/quartus/eda/sim_lib/synopsys/stratixv_atoms_ncrypt.t.v, 8499

Warning-[ZONNMC] 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
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 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 megafuction cannot be simulated alone. To simulate the ALTMEMPHY megafuction, 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 megafuction 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 .vho 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 port names and capitalization in the memory model match the portnames 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.

The original instantiation may be similar to the following code:

```vh
// << 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),
}```
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 EDA Simulation Tool and click EDA 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 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 Volume 6: Design Flow Tutorials of the External Memory Interface Handbook.
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 3–1). The files are in VHDL93 format.

<table>
<thead>
<tr>
<th>Library</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>altera_mf</code></td>
<td><code>&lt;QUARTUS_ROOTDIR&gt;eda\sim_lib\altera_mf_components.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;QUARTUS_ROOTDIR&gt;eda\sim_lib\altera_mf.vhd</code></td>
</tr>
<tr>
<td><code>lpm</code></td>
<td><code>\eda\sim_lib\220pack.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>\eda\sim_lib\220model.vhd</code></td>
</tr>
<tr>
<td><code>sgate</code></td>
<td><code>eda\sim_lib\sgate_pack.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>eda\sim_lib\sgate.vhd</code></td>
</tr>
<tr>
<td><code>&lt;device name&gt;</code></td>
<td><code>eda\sim_lib\&lt;device name&gt;_atoms.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>eda\sim_lib\&lt;device name&gt;_components.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>eda\sim_lib\&lt;device name&gt;_hssi_atoms.vhd (1)</code></td>
</tr>
<tr>
<td><code>altera</code></td>
<td><code>eda\sim_lib\altera_primitives_components.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>eda\sim_lib\altera_syn_attributes.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>eda\sim_lib\altera_primitives.vhd</code></td>
</tr>
<tr>
<td><code>ALTGXB (1)</code></td>
<td><code>&lt;device name&gt;_mf.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;device name&gt;_mf_components.vhd</code></td>
</tr>
<tr>
<td><code>&lt;device name&gt;_hssi (1)</code></td>
<td><code>&lt;device name&gt;_hssi_components.vhd</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;device name&gt;_hssi_atoms.vhd</code></td>
</tr>
</tbody>
</table>
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

Note for Table 3–1:

(1) Applicable only for Arria II GX and Stratix IV devices.
3. Compile the files into the appropriate library as shown in Table 3–2 on page 3–8.

Table 3–2. Files to Compile—Verilog HDL IP Functional Simulation Models

<table>
<thead>
<tr>
<th>Library</th>
<th>File Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>altera_mf_ver</td>
<td>&lt;QUARTUS_ROOTDIR\eda\sim_lib\altera_mf.v</td>
</tr>
<tr>
<td>lpm_ver</td>
<td>\eda\sim_lib\220model.v</td>
</tr>
<tr>
<td>sgate_ver</td>
<td>eda\sim_lib\sgate.v</td>
</tr>
<tr>
<td>&lt;device name&gt;_ver</td>
<td>eda\sim_lib&lt;device name&gt;_atoms.v</td>
</tr>
<tr>
<td></td>
<td>eda\sim_lib&lt;device name&gt;_hssi_atoms.v (1)</td>
</tr>
<tr>
<td>altera_ver</td>
<td>eda\sim_lib\altera_primitives.v</td>
</tr>
<tr>
<td>ALTGXB_ver (1)</td>
<td>&lt;device name&gt;_mf.v</td>
</tr>
<tr>
<td>&lt;device name&gt;_hssi_ver (1)</td>
<td>&lt;device name&gt;_hssi_atoms.v</td>
</tr>
<tr>
<td>auk_ddr_hp_user_lib</td>
<td>&lt;QUARTUS_ROOTDIR\libraries\vhdl\altera\altera_europa_support_lib.v</td>
</tr>
<tr>
<td></td>
<td>alt_mem_phy_defines.v</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_phy_alt_mem_phy_seq_wrapper.vo</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_auk_ddr_hp_controller_wrapper.vo</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;.v</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_example_top.v</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_phy.v</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_controller_phy.v</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_phy_alt_mem_phy_pll.v</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_phy_alt_mem_phy.v</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_example_driver.v</td>
</tr>
<tr>
<td></td>
<td>&lt;project directory&lt;variation name&gt;_ex_lfsr8.v</td>
</tr>
<tr>
<td>testbench</td>
<td>&lt;variation name&gt;_example_top_tb.v</td>
</tr>
<tr>
<td></td>
<td>testbench&lt;variation name&gt;_mem_model.v</td>
</tr>
</tbody>
</table>

Notes for Table 3–2:

(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. Configure your simulator to use transport delays, a timestep of picoseconds, and to include all the libraries in Table 3–2.

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 Altera PHY interface (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.
This chapter provides additional information about the document and Altera.

**Document Revision History**

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2011</td>
<td>3.0</td>
<td>■ Added the Simulation Overview chapter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Added the Simulation Walkthrough with UniPHY IP chapter.</td>
</tr>
<tr>
<td>December 2010</td>
<td>2.1</td>
<td>Updated for 10.1 release.</td>
</tr>
<tr>
<td>July 2010</td>
<td>2.0</td>
<td>Updated for 10.0 release.</td>
</tr>
<tr>
<td>January 2010</td>
<td>1.1</td>
<td>Corrected minor typos.</td>
</tr>
<tr>
<td>November 2009</td>
<td>1.0</td>
<td>First published.</td>
</tr>
</tbody>
</table>

**How to Contact Altera**

To locate the most up-to-date information about Altera products, refer to the following table.

<table>
<thead>
<tr>
<th>Contact (1)</th>
<th>Contact Method</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical support</td>
<td>Website</td>
<td><a href="http://www.altera.com/support">www.altera.com/support</a></td>
</tr>
<tr>
<td>Technical training</td>
<td>Website</td>
<td><a href="http://www.altera.com/training">www.altera.com/training</a></td>
</tr>
<tr>
<td></td>
<td>Email</td>
<td><a href="mailto:custrain@altera.com">custrain@altera.com</a></td>
</tr>
<tr>
<td>Product literature</td>
<td>Website</td>
<td><a href="http://www.altera.com/literature">www.altera.com/literature</a></td>
</tr>
<tr>
<td>Non-technical support (General)</td>
<td>Email</td>
<td><a href="mailto:nacomp@altera.com">nacomp@altera.com</a></td>
</tr>
<tr>
<td>(Software Licensing)</td>
<td>Email</td>
<td><a href="mailto:authorization@altera.com">authorization@altera.com</a></td>
</tr>
</tbody>
</table>

Note to Table:
(1) You can also contact your local Altera sales office or sales representative.

**Typographic Conventions**

The following table shows the typographic conventions this document uses.

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold Type with Initial Capital Letters</strong></td>
<td>Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, <strong>Save As</strong> dialog box. For GUI elements, capitalization matches the GUI.</td>
</tr>
<tr>
<td><strong>bold type</strong></td>
<td>Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, **qdesigns** directory, D: drive, and <strong>chiptrip.gdf</strong> file.</td>
</tr>
<tr>
<td>Visual Cue</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td><em>Italic Type with Initial Capital Letters</em></td>
<td>Indicate document titles. For example, <em>Stratix IV Design Guidelines</em>.</td>
</tr>
<tr>
<td><em>italic type</em></td>
<td>Indicates variables. For example, $n + 1$.</td>
</tr>
<tr>
<td></td>
<td>Variable names are enclosed in angle brackets (&lt;&gt;). For example, <code>&lt;file name&gt;</code> and <code>&lt;project name&gt;.pof</code> file.</td>
</tr>
<tr>
<td><em>Initial Capital Letters</em></td>
<td>Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.</td>
</tr>
<tr>
<td>&quot;Subheading Title&quot;</td>
<td>Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, &quot;Typographic Conventions.&quot;</td>
</tr>
<tr>
<td><em>Courier type</em></td>
<td>Indicates signal, port, register, bit, block, and primitive names. For example, <em>data1, tdi, and input</em>. The suffix <em>n</em> denotes an active-low signal. For example, <em>resetn</em>.</td>
</tr>
<tr>
<td></td>
<td>Indicates command line commands and anything that must be typed exactly as it appears. For example, <em>c:\qdesigns\tutorial\chiptrip.gdf</em>.</td>
</tr>
<tr>
<td></td>
<td>Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword <em>SUBDESIGN</em>), and logic function names (for example, <em>TRI</em>).</td>
</tr>
<tr>
<td>←</td>
<td>An angled arrow instructs you to press the Enter key.</td>
</tr>
<tr>
<td>1., 2., 3., and a., b., c., and so on</td>
<td>Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.</td>
</tr>
<tr>
<td>•</td>
<td>Bullets indicate a list of items when the sequence of the items is not important.</td>
</tr>
<tr>
<td><img src="hand.png" alt="Hand" /></td>
<td>The hand points to information that requires special attention.</td>
</tr>
<tr>
<td><img src="question_mark.png" alt="Question Mark" /></td>
<td>A question mark directs you to a software help system with related information.</td>
</tr>
<tr>
<td><img src="foot.png" alt="Foot" /></td>
<td>The feet direct you to another document or website with related information.</td>
</tr>
<tr>
<td><img src="caution.png" alt="Caution" /></td>
<td>A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.</td>
</tr>
<tr>
<td><img src="warning.png" alt="Warning" /></td>
<td>A warning calls attention to a condition or possible situation that can cause you injury.</td>
</tr>
<tr>
<td><img src="envelope.png" alt="Envelope" /></td>
<td>The envelope links to the <em>Email Subscription Management Center</em> page of the Altera website, where you can sign up to receive update notifications for Altera documents.</td>
</tr>
</tbody>
</table>
External Memory Interface Handbook Volume 4

Section II. Timing Analysis
Chapter 1. Timing Analysis Methodology

Memory Interface Timing Components .......................................................... 1–2
Source-Synchronous Paths ................................................................. 1–2
Calibrated Paths ................................................................. 1–2
Internal FPGA Timing Paths ........................................................... 1–3
Other FPGA Timing Parameters ......................................................... 1–3
FPGA Timing Paths ................................................................. 1–5
Arria II Device PHY Timing Paths ....................................................... 1–5
Stratix III and Stratix IV PHY Timing Paths ............................................ 1–7
Stratix V Timing paths ............................................................... 1–9
Cyclone III and Cyclone IV PHY Timing Paths ......................................... 1–10
Timing Constraint and Report Files ....................................................... 1–11
ALTMEMPHY Megafunction ............................................................. 1–11
<variation_name>_ddr_timing.sdc ............................................... 1–12
<variation_name>_ddr_timing.tcl ............................................... 1–12
<variation_name>_report_timing.tcl .................................... 1–12
<variation_name>_report_timing_core.tcl ................................ 1–13
<variation_name>_ddr_pins.tcl ............................................... 1–13
UniPHY IP ................................................................. 1–13
<variation_name>.sdc ............................................................. 1–13
<variation_name>_timing.tcl ....................................................... 1–14
<variation_name>_report_timing.tcl .................................... 1–14
<variation_name>_report_timing_core.tcl ................................ 1–14
<variation_name>_pin_map.tcl ....................................................... 1–14
<variation_name>_parameters.tcl .................................................. 1–14
Timing Analysis Description ............................................................... 1–14
Address and Command .............................................................. 1–15
PHY or Core ................................................................. 1–15
PHY or Core Reset .............................................................. 1–15
Read Capture and Write ............................................................. 1–15
Cyclone III and Stratix III .............................................................. 1–15
Read Capture ................................................................. 1–18
Write Capture ................................................................. 1–20
Arria II, Cyclone IV, Stratix IV and Stratix V ........................................... 1–22
Read Capture ................................................................. 1–22
Cyclone IV Devices .............................................................. 1–22
Write ................................................................. 1–22
Read Resynchronization ............................................................. 1–23
Mimic Path ................................................................. 1–23
DQS versus CK—Arria II GX, Cyclone III, and Cyclone IV Devices ................................................................. 1–23
Write Leveling tDQSS .......................................................... 1–24
Write Leveling tDSH/tDSS ....................................................... 1–24
DK versus CK (RLDRAM II with UniPHY) ........................................... 1–24
Bus Turnaround Time .......................................................... 1–24
Timing Report DDR ............................................................. 1–24
Calibration Effect in Timing Analysis .................................................. 1–27
Calibration Emulation for Calibrated Path ............................................ 1–27
Calibration Error or Quantization Error .................................................. 1–28
Calibration Uncertainties ............................................................. 1–28

Section II. Timing Analysis

Contents
Chapter 2. Timing Closure

Common Issues .......................................................... 2–1
  Missing Timing Margin Report ......................................... 2–1
  Incomplete Timing Margin Report ..................................... 2–1
  Read Capture Timing ..................................................... 2–1
  Write Timing ............................................................... 2–2
  Address and Command Timing ........................................ 2–2
  PHY Reset Recovery and Removal .................................... 2–3
  Clock-to-Strobe (for DDR and DDR2 SDRAM Only) ............... 2–3
  Read Resynchronization and Write Leveling Timing (for SDRAM Only) .............................................................. 2–3
  Optimizing Timing ......................................................... 2–4

Chapter 3. Timing Deration Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs

Background ..................................................................... 3–1
  ISI Effects ...................................................................... 3–1
  Calibration Effects ........................................................ 3–2
  Board Effects .................................................................. 3–2
Implementing Multiple Chip Selects in the Designs .................. 3–3
  Timing Deration using the Board Settings ......................... 3–3
    Slew Rates .................................................................. 3–3
    Intersymbol Interference .............................................. 3–4
    Board Skews .................................................................. 3–4
  Timing Deration Using the Excel-Based Calculator ................ 3–5
    Before You Use the Excel-based Calculator for Timing Deration .............................................................. 3–6
    Using the Excel-Based Calculator .................................... 3–6
    Using the Excel-based Calculator for Timing Deration (Without Board Trace Models) .............................. 3–9

Additional Information

Revision History ............................................................. Info–1
How to Contact Altera ...................................................... Info–1
Typographic Conventions ............................................... Info–1
1. Timing Analysis Methodology

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, Cyclone® III, Cyclone IV, 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 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)
- Read resynchronization path timing (applicable for DDR, DDR2, and DDR3 SDRAM in Arria II, Stratix III, Stratix IV, and Stratix V devices)
- Read postamble path timing (applicable for DDR and DDR2 SDRAM in Stratix II devices)
Chapter 1: Timing Analysis Methodology

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 resynchronization 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 \( f_{\text{MAX}} \) 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 Layout Guidelines* section in volume 2 of the *External Memory Interface Handbook* 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, 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 1–1 categorizes all Arria II devices external memory interface timing paths.

<table>
<thead>
<tr>
<th>Timing Path</th>
<th>Circuit Category</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Data (2), (7)</td>
<td>Source-Synchronous</td>
<td>Memory DQ, DQS Pins</td>
<td>DQ Capture Registers in IOE</td>
</tr>
<tr>
<td>Write Data (2), (7)</td>
<td>Source-Synchronous</td>
<td>FPGA DQ, DQS Pins</td>
<td>Memory DQ, DM, and DQS Pins</td>
</tr>
<tr>
<td>Address and command (2)</td>
<td>Source-Synchronous</td>
<td>FPGA CK/CK# and Addr/Cmd P</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Clock-to-Strobe (2)</td>
<td>Source-Synchronous</td>
<td>FPGA CK/CK# and DQS Output Pins</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Read Resynchronization (2), (3)</td>
<td>Calibrated</td>
<td>IOE Capture Registers</td>
<td>IOE Resynchronization Registers</td>
</tr>
<tr>
<td>Read Resynchronization (2), (6)</td>
<td>Calibrated</td>
<td>IOE Capture Registers</td>
<td>Read FIFO in FPGA Core</td>
</tr>
<tr>
<td>PHY IOE-Core Paths (2), (3)</td>
<td>Source-Synchronous</td>
<td>IOE Resynchronization Registers</td>
<td>FIFO in FPGA Core</td>
</tr>
<tr>
<td>PHY and Controller Internal Paths (2)</td>
<td>Internal Clock f_{MAX}</td>
<td>Core Registers</td>
<td>Core Registers</td>
</tr>
<tr>
<td>I/O Toggle Rate (4)</td>
<td>I/O</td>
<td>FPGA Output Pin</td>
<td>Memory Input Pins</td>
</tr>
</tbody>
</table>
**Table 1–1. Arria II Devices External Memory Interface Timing Paths** *(Note 1) (Part 2 of 2)*

<table>
<thead>
<tr>
<th>Timing Path</th>
<th>Circuit Category</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Clock Specifications (Jitter, DCD) <em>(5)</em></td>
<td>I/O</td>
<td>FPGA Output Pin</td>
<td>Memory Input Pins</td>
</tr>
</tbody>
</table>

**Notes to Table 1–1:**

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 1–1 shows the Arria II GX devices input datapath registers and circuit types.

![UniPHY IP interfaces bypass the synchronization registers.](image)

---

**Figure 1–1. Arria II GX Devices Input Data Path Registers and Circuit Types in SDRAM Interface**
Figure 1–2 shows the Arria II GZ devices input datapath registers and circuit types.

**Figure 1–2. 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 1–3 and Table 1–2 are based on Stratix IV devices, but they are applicable to Stratix III devices.

**Figure 1–3** 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.

Table 1–2 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 1–2. Stratix IV External Memory Interface Timing Paths  (Part 1 of 2)

<table>
<thead>
<tr>
<th>Timing Path</th>
<th>Circuit Category</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Data (1)</td>
<td>Source-Synchronous and Calibrated</td>
<td>Memory DQ, DQS Pins</td>
<td>DQ Capture Registers in IOE</td>
</tr>
<tr>
<td>Write Data (1)</td>
<td>Source-Synchronous and Calibrated</td>
<td>FPGA DQ, DQS Pins</td>
<td>Memory DQ, DM, and DQS Pins</td>
</tr>
<tr>
<td>Address and command (1)</td>
<td>Source-Synchronous</td>
<td>FPGA CK/CK# and Addr/Cmd Pins</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Clock-to-Strobe (1)</td>
<td>Source-Synchronous</td>
<td>FPGA CK/CK# and DQS Output Pins</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Read Resynchronization (1), (2)</td>
<td>Calibrated</td>
<td>IOE Capture Registers</td>
<td>IOE Alignment and Resynchronization Registers</td>
</tr>
<tr>
<td>Read Resynchronization (1), (5)</td>
<td>Calibrated</td>
<td>IOE Capture Registers</td>
<td>Read FIFO in FPGA Core</td>
</tr>
</tbody>
</table>
Table 1–2. Stratix IV External Memory Interface Timing Paths (Part 2 of 2)

<table>
<thead>
<tr>
<th>Timing Path</th>
<th>Circuit Category</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHY IOE-Core Paths (1), (2)</td>
<td>Source-Synchronous</td>
<td>IOE Half Data Rate Registers and Half-Rate Resynchronization Clock</td>
<td>FIFO in FPGA Core</td>
</tr>
<tr>
<td>PHY &amp; Controller Internal Paths (1)</td>
<td>Internal Clock fMAX</td>
<td>Core registers</td>
<td>Core registers</td>
</tr>
<tr>
<td>I/O Toggle Rate (3)</td>
<td>I/O – Data sheet</td>
<td>FPGA Output Pin</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Output Clock Specifications (Jitter, DCD) (4)</td>
<td>I/O – Data sheet</td>
<td>FPGA Output Pin</td>
<td>Memory Input Pins</td>
</tr>
</tbody>
</table>

Notes to Table 1–2:
(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.

**Stratix V Timing paths**

Figure 1–4 shows a block diagram of the Stratix V input data path.

**Figure 1–4. Stratix V Input Data Path**
Table 1–3 categorizes all Stratix V devices external memory interface timing paths.

<table>
<thead>
<tr>
<th>Timing Path</th>
<th>Circuit Category</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Data</td>
<td>Source-Synchronous and Calibrated</td>
<td>Memory DQ, DQS Pins</td>
<td>DQ Capture Registers in IOE</td>
</tr>
<tr>
<td>Write Data</td>
<td>Source-Synchronous and Calibrated</td>
<td>FPGA DQ, DM, DQS Pins</td>
<td>Memory DQ, DM, and DQS Pins</td>
</tr>
<tr>
<td>Address and command</td>
<td>Source-Synchronous</td>
<td>FPGA CK/CK# and Addr/Cmd Pins</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Clock-to-Strobe</td>
<td>Source-Synchronous</td>
<td>FPGA CK/CK# and DQS Output Pins</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Read Resynchronization</td>
<td>Source-Synchronous</td>
<td>IOE Capture Registers</td>
<td>Read FIFO in IOE</td>
</tr>
<tr>
<td>PHY &amp; Controller Internal Paths</td>
<td>Internal Clock f_{MAX}</td>
<td>Core Registers</td>
<td>Core Registers</td>
</tr>
<tr>
<td>i/O Toggle Rate (3)</td>
<td>I/O – Data sheet</td>
<td>FPGA Output Pin</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Output Clock Specifications (Jitter, DCD) (4)</td>
<td>I/O – Data sheet</td>
<td>FPGA Output Pin</td>
<td>Memory Input Pins</td>
</tr>
</tbody>
</table>

Notes to Table 1–3:
(1) This table lists the timing paths applicable for an interface between 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 1–4 categorizes 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 1–5 shows the Cyclone III and Cyclone IV input datapath registers and circuit types.

Table 1–4. Cyclone III and Cyclone IV SDRAM External Memory Interface Timing Paths (Note 1) (Part 1 of 2)

<table>
<thead>
<tr>
<th>Timing Path</th>
<th>Circuit Category</th>
<th>Source</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Data</td>
<td>Calibrated</td>
<td>Memory DQ, DQS Pins</td>
<td>FPGA DQ Capture Registers in LEs</td>
</tr>
<tr>
<td>Write Data</td>
<td>Source-Synchronous</td>
<td>FPGA DQ, DQS Pins</td>
<td>Memory DQ, DM, and DQS Pins</td>
</tr>
<tr>
<td>Address and command</td>
<td>Source-Synchronous</td>
<td>FPGA CK/CK# and Addr/Cmd Pins</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>Clock-to-Strobe</td>
<td>Source-Synchronous</td>
<td>FPGA CK/CK# and DQS Output Pins</td>
<td>Memory Input Pins</td>
</tr>
<tr>
<td>PHY Internal Timing</td>
<td>Internal Clock f_{MAX}</td>
<td>LE Half Data Rate Registers</td>
<td>FIFO in FPGA Core</td>
</tr>
<tr>
<td>i/O Toggle Rate (3)</td>
<td>I/O – Data sheet</td>
<td>FPGA Output Pin</td>
<td>Memory Input Pins</td>
</tr>
</tbody>
</table>
The timing constraints differ for the ALTMEMPHY megafunction and the UniPHY IP.

**ALTMEMPHY Megafuction**

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>_ddr_timing.sdc`
- `<variation_name>_ddr_timing.tcl` (except Cyclone III devices)
- `<variation_name>_report_timing.tcl`
- `<variation_name>_report_timing_core.tcl` (except Cyclone III devices)
- `<variation_name>_ddr_pins.tcl`
<variation_name>_ddr_timing.sdc

The Synopsys design constraint (.sdc) file has the name <controller_variation_name>_phy_ddr_timing.sdc when you instantiate the ALTMEMPHY megafuncion in the Altera memory controller, and has the name <phy_variation_name>_ddr_timing.sdc when you instantiate the ALTMEMPHY megafuncion 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 1–13. 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 pre-defined. 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 megafuncion 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 file 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 file.

<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.
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.

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 file. 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 wizards for the controllers with UniPHY generate 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

The .sdc file <variation_name>.sdc is listed in the wizard-generated Quartus II IP (.qip) file. 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 file 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)
Chapter 1: Timing Analysis Methodology

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.

Refer to the scripts listed in the “Timing Constraint and Report Files” section for detailed timing analysis description.

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 calculates 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 *Volume 6: Design Flow Tutorials* of the *External Memory Interface Handbook*.

Timing margins for chip-to-chip data transfers can be defined as:

\[
\text{Margin} = \text{bit period} - \text{transmitter uncertainties} - \text{receiver requirements}
\]

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 tCO 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.

Refer to the DC and Switching Characteristics chapter of the Cyclone III Device Handbook or Stratix III Device Handbook for TCCS and SW specifications.

Figure 1–6 relates this terminology to a timing budget diagram.

**Figure 1–6. 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 1–6 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 \text{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 \(t_{\text{EXT}}\). 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 1–7 illustrates the timing budget for this condition. A timing budget similar to that shown in Figure 1–7 is used for Cyclone III and Stratix III FPGA read and write data timing paths.

**Figure 1–7. Sample Timing Budget with Unbalanced (TCCS and SW) Timing Parameters**

Therefore:

\[
\begin{align*}
\text{Setup margin} & = \text{RSKM}_{\text{SETUP}} = P - \text{TCCS}_{\text{LEAD}} - \text{SW}_{\text{SETUP}} - t_{\text{EXT}} \\
\text{Hold margin} & = \text{RSKM}_{\text{HOLD}} = (\text{TUI} - P) - \text{TCCS}_{\text{LAG}} - \text{SW}_{\text{HOLD}} - t_{\text{EXT}}
\end{align*}
\]

The timing budget illustrated in Figure 1–6 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 1–7 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 $TCCS_{LAG} = t_{QHS}$

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 1–8 shows the timing budget for a read data timing path.

Table 1–5 details a read data timing analysis for a Stratix III –2 speed-grade device interfacing with a 400-MHz DDR2 SDRAM component.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specifications</th>
<th>Value (ps)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Specifications (1)</td>
<td>$t_{HP}$</td>
<td>1250</td>
<td>Average half period as specified by the memory data sheet, $t_{HP} = 1/2 \times t_{CK}$</td>
</tr>
<tr>
<td></td>
<td>$t_{DCD}$</td>
<td>50</td>
<td>Duty cycle distortion = 2% × $t_{CK} = 0.02 \times 2500$ ps</td>
</tr>
<tr>
<td></td>
<td>$t_{DQSQ}$</td>
<td>200</td>
<td>Skew between DQS and DQ from memory</td>
</tr>
<tr>
<td></td>
<td>$t_{QHS}$</td>
<td>300</td>
<td>Data hold skew factor as specified by memory</td>
</tr>
<tr>
<td>FPGA Specifications</td>
<td>$t_{SW_SETUP}$</td>
<td>181</td>
<td>FPGA sampling window specifications for a given configuration (DLL mode, width, location, and so on.)</td>
</tr>
<tr>
<td></td>
<td>$t_{SW_HOLD}$</td>
<td>306</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1: Timing Analysis Methodology

Timing Analysis Description

Table 1–6 details 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 1–5. Read Data Timing Analysis for Stratix III Device with a 400-MHz DDR2 SDRAM (Note 1) (Part 2 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specifications</th>
<th>Value (ps)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Board Specifications</td>
<td>t_EXT</td>
<td>20</td>
<td>Maximum board trace variation allowed between any two signal traces (user specified parameter)</td>
</tr>
<tr>
<td>Timing Calculations</td>
<td>t_DVW</td>
<td>710</td>
<td>t_HP – t_DCD – t_QHS – 2 × t_EXT</td>
</tr>
<tr>
<td></td>
<td>t_DQS_PHASE_DELAY</td>
<td>500</td>
<td>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</td>
</tr>
<tr>
<td>Results</td>
<td>Setup margin</td>
<td>99</td>
<td>RSKM_SETUP = t_DQS_PHASE_DELAY – t_DQS – t_SW_SETUP – t_EXT</td>
</tr>
<tr>
<td></td>
<td>Hold margin</td>
<td>74</td>
<td>RSKM_HOLD = t_HP – t_DCD – t_DQS_PHASE_DELAY – t_QHS – t_SW_HOLD – t_EXT</td>
</tr>
</tbody>
</table>

Notes to Table 1–5:
(1) This sample calculation uses memory timing parameters from a 72-bit wide 256-MB micron MT9HTF3272AY-80E 400-MHz DDR2 SDRAM DIMM.

Table 1–6. Read Data Timing Analysis for a 200-MHz DDR2 SDRAM on a Cyclone III Device (Note 1)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specifications</th>
<th>Value (ps)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Specifications</td>
<td>t_HP</td>
<td>2500</td>
<td>Average half period as specified by the memory data sheet</td>
</tr>
<tr>
<td></td>
<td>t_DCD_TOTAL</td>
<td>250</td>
<td>Duty cycle distortion = 2% × t_CK = 0.02 × 5000 ps</td>
</tr>
<tr>
<td></td>
<td>t_AC</td>
<td>± 500</td>
<td>Data (DQ) output access time for a 267-MHz DDR2 SDRAM component</td>
</tr>
<tr>
<td>FPGA Specifications</td>
<td>t_SW_SETUP</td>
<td>580</td>
<td>FPGA sampling window specification for a given configuration (interface width, location, and so on).</td>
</tr>
<tr>
<td></td>
<td>t_SW_HOLD</td>
<td>550</td>
<td></td>
</tr>
<tr>
<td>Board Specifications</td>
<td>t_EXT</td>
<td>20</td>
<td>Maximum board trace variation allowed between any two signal traces (user specified parameter)</td>
</tr>
<tr>
<td>Timing Calculations</td>
<td>t_DVW</td>
<td>1230</td>
<td>t_HP – t_DCD – 2 × t_AC – 2 × t_EXT</td>
</tr>
<tr>
<td>Results</td>
<td>Total margin</td>
<td>100</td>
<td>t_DVW × t_SW_SETUP × t_SW_HOLD</td>
</tr>
</tbody>
</table>

Notes to Table 1–6:
(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 1–9.

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 1–9 illustrates the timing budget for a write data timing path.

### Table 1–7. Write Data Timing Analysis for 400-MHz DDR2 SDRAM Stratix III Device (Note 1) (Part 1 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specifications</th>
<th>Value (ps)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Specifications</td>
<td>t_{HP}</td>
<td>1250</td>
<td>Average half period as specified by the memory data sheet</td>
</tr>
<tr>
<td></td>
<td>t_{DSA}</td>
<td>250</td>
<td>Memory setup requirement (derated for DQ/DQS edge rates and V_{REF} reference voltage)</td>
</tr>
<tr>
<td></td>
<td>t_{DHA}</td>
<td>250</td>
<td>Memory hold requirement (derated for DQ/DQS edge rates and V_{REF} reference voltage)</td>
</tr>
<tr>
<td>FPGA Specifications</td>
<td>TCCS_{LEAD}</td>
<td>229</td>
<td>FPGA transmitter channel-to-channel skew for a given configuration (PLL setting, location, and width).</td>
</tr>
<tr>
<td></td>
<td>TCCS_{LAG}</td>
<td>246</td>
<td></td>
</tr>
<tr>
<td>Board Specifications</td>
<td>t_{EXT}</td>
<td>20</td>
<td>Maximum board trace variation allowed between any two signal traces (user specified parameter)</td>
</tr>
</tbody>
</table>
Table 1–7. Write Data Timing Analysis for 400-MHz DDR2 SDRAM Stratix III Device (Note 1) (Part 2 of 2)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specifications</th>
<th>Value (ps)</th>
<th>Description</th>
</tr>
</thead>
</table>
| Timing Calculations| \( t_{\text{OUTPUT_CLOCK_OFFSET}} \) | 625        | Output clock phase offset between DQ & DQS output clocks = 90°. \[
\begin{align*}
  t_{\text{OUTPUT_CLOCK_OFFSET}} &= (\text{output clock phase DQ and DQS offset} \times t_{\text{CK}}) / 360° \\
  &= (90° \times 2500) / 360° = 625
\end{align*}
\] |
|                    | \( \text{TX}_{\text{DVWLEAD}} \) | 396        | Transmitter data valid window = \( t_{\text{OUTPUT_CLOCK_OFFSET}} - T_{\text{CCSLEAD}} \) |
|                    | \( \text{TX}_{\text{DVWL AG}} \) | 379        | Transmitter data valid window = \( t_{\text{HP}} - t_{\text{OUTPUT_CLOCK_OFFSET}} - T_{\text{CCSLAG}} \) |
| Results            | Setup margin            | 126        | \( \text{TX}_{\text{DVWLEAD}} - t_{\text{EXT}} - t_{\text{DSA}} \)               |
|                    | Hold margin             | 109        | \( \text{TX}_{\text{DVWL AG}} - t_{\text{EXT}} - t_{\text{DHA}} \)               |

Notes to Table 1–7:
(1) This sample calculation uses memory timing parameters from a 72-bit wide 256-MB micron MT9HTF3272AY-80E 400-MHz DDR2 SDRAM DIMM.

Table 1–8 details 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 1–8. Write Data Timing Analysis for a 200-MHz DDR2 SDRAM Interface on a Cyclone III Device (Note 1)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Specifications</th>
<th>Value (ps)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Specifications</td>
<td>( t_{\text{HP}} )</td>
<td>2500</td>
<td>Average half period as specified by the memory data sheet</td>
</tr>
<tr>
<td></td>
<td>( t_{\text{DCD_TOTAL}} )</td>
<td>250</td>
<td>Total duty cycle distortion = 5% \times t_{\text{CK}} = 0.05 \times 5000</td>
</tr>
<tr>
<td></td>
<td>( t_{\text{DS}} ) (derated)</td>
<td>395</td>
<td>Memory setup requirement from a 267-MHz DDR2 SDRAM component (derated for single-ended DQS and 1 V/ns slew rate)</td>
</tr>
<tr>
<td></td>
<td>( t_{\text{DH}} ) (derated)</td>
<td>335</td>
<td>Memory hold from DDR2 267-MHz component (derated for single-ended DQS and 1 V/ns slew rate)</td>
</tr>
<tr>
<td>FPGA Specifications</td>
<td>( T_{\text{CCSLEAD}} )</td>
<td>790</td>
<td>FPGA TCCS for a given configuration (PLL setting, location, width)</td>
</tr>
<tr>
<td></td>
<td>( T_{\text{CCSLAG}} )</td>
<td>380</td>
<td></td>
</tr>
<tr>
<td>Board Specifications</td>
<td>( t_{\text{EXT}} )</td>
<td>20</td>
<td>Maximum board trace variation allowed between any two signal traces (user specified parameter)</td>
</tr>
<tr>
<td>Timing Calculations</td>
<td>( \text{TX}_{\text{DVWLEAD}} )</td>
<td>460</td>
<td>Transmitter data valid window = ( t_{\text{OUTPUT_CLOCK_OFFSET}} - T_{\text{CCSLEAD}} )</td>
</tr>
<tr>
<td></td>
<td>( \text{TX}_{\text{DVWL AG}} )</td>
<td>870</td>
<td>Transmitter data valid window = ( t_{\text{HP}} - t_{\text{OUTPUT_CLOCK_OFFSET}} - T_{\text{CCSLAG}} )</td>
</tr>
</tbody>
</table>
|                    | \( t_{\text{OUTPUT_CLOCK_OFFSET}} \) | 1250       | Output clock phase offset between DQ/DQS output clocks = 90° \[
\begin{align*}
  t_{\text{OUTPUT_CLOCK_OFFSET}} &= (\text{output clock phase DQ & DQS offset} \times t_{\text{CK}}) / 360° \\
  &= (90° \times 5000) / 360° = 1250
\end{align*}
\] |
| Results            | Setup margin            | 45         | \( \text{TX}_{\text{DVWLEAD}} - t_{\text{EXT}} - t_{\text{DS}} \)               |
|                    | Hold margin             | 265        | \( \text{TX}_{\text{DVWL AG}} - t_{\text{EXT}} - t_{\text{DHA}} - t_{\text{DCD_TOTAL}} \) |

Note to Table 1–8:
(1) For more information on Cyclone III –6 speed-grade device read capture and timing analysis, refer to “Read Capture” on page 1–16.
Arria II, Cyclone IV, 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` file (ALTMEMPHY) or `<phy_variation_name>.sdc` file (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` ensure 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` file (ALTMEMPHY) or `<phy_variation_name>.sdc` file (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 megamodule.

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 (tDQSS) 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 (tDSS/tDSH) 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 tDQSS**

In DDR2 SDRAM (with UniPHY) and DDR3 SDRAM (with ALTMEMPHY and UniPHY) interfaces, write leveling tDQSS 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 \( t_{\text{DSH}}/t_{\text{DSS}} \)

In DDR2 SDRAM (with UniPHY) and DDR3 SDRAM (with ALTMEMPHY and UniPHY) interfaces, write leveling \( t_{\text{DSH}}/t_{\text{DSS}} \) 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. This analysis is only available in the UniPHY IP for Arria II GZ, Stratix IV, and Stratix V devices.

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 timing slacks on specific paths of the DDR SDRAM design for example, such as:

- Read capture
- Read resynchronization
- Mimic, address and command
- Core
- Core reset and removal
- Half-rate address and command
- DQS versus CK
- Write
- Write leveling (\( t_{\text{DQSS}} \))
- Write leveling (\( t_{\text{DSS}}/t_{\text{DSH}} \))
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 1–29. 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 file 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.

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 1–10.

**Figure 1–10. Timing Margin Summary Window Generated by Report DDR Task**

Bus turnaround time shown in Figure 1–10 is available in all UniPHY IPs and devices except in QDR II and QDR II+ SRAM memory protocols and Stratix III devices.
Figure 1–11 and Figure 1–12 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 1–11. Read Capture Margin Summary Window**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Setup Slack</th>
<th>Hold Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>After Calibration Read Capture</td>
<td>0.371</td>
<td>0.323</td>
</tr>
<tr>
<td>Before Calibration Read Capture</td>
<td>0.280</td>
<td>0.273</td>
</tr>
<tr>
<td>Memory Calibration</td>
<td>0.078</td>
<td>0.150</td>
</tr>
<tr>
<td>Deskew Read</td>
<td>0.157</td>
<td>0.044</td>
</tr>
<tr>
<td>Quantization error</td>
<td>-0.050</td>
<td>-0.050</td>
</tr>
<tr>
<td>Calibration uncertainty</td>
<td>-0.093</td>
<td>-0.093</td>
</tr>
</tbody>
</table>

**Figure 1–12. Write Capture Margin Summary Window**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Setup Slack</th>
<th>Hold Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>After Calibration Write</td>
<td>0.279</td>
<td>0.279</td>
</tr>
<tr>
<td>Before Calibration Write</td>
<td>0.027</td>
<td>0.026</td>
</tr>
<tr>
<td>Memory Calibration</td>
<td>0.155</td>
<td>0.113</td>
</tr>
<tr>
<td>Deskew Write and/or more clock pessimism removal</td>
<td>0.216</td>
<td>0.240</td>
</tr>
<tr>
<td>Quantization error</td>
<td>-0.050</td>
<td>-0.050</td>
</tr>
<tr>
<td>Calibration uncertainty</td>
<td>-0.093</td>
<td>-0.093</td>
</tr>
</tbody>
</table>

**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` script. 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
Chapter 1: Timing Analysis Methodology

Calibration Effect in Timing Analysis

Figure 1–13 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 1–13. SDC Assignments Report Window

Figure 1–14 shows the timing analysis results calculated using FPGA timing model before adjustment in the Before Calibration panel.

Figure 1–14. Read and Write Before Calibration

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 `tDQSS` and `tDQSO`. 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 tTCCS and tSW 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 1–15 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 1–15. 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 bi-directional pins with the output enable set to VCC.
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 1–16. In this configuration, the high register connects to VCC and the low register connects to GND.

![Figure 1–16. DDIO Configuration](image)

- CK and CK# pins must be fed by a DDIO_OUT WYSIWYG with datainhi connected to GND and datainlo connected to VCC.
- 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 uninveted.
- 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 1–17, with a signal splitter to generate the n- pin pair and a regional clock network-to-clock to output DDIO block.

**Figure 1–17. DIDO Configuration with Signal Splitter**

Notes to Figure 1–17:
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, VCC, 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, VCC, 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, VCC, 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 1–9 lists I/O standards supported for the write clock and write data signals for each memory type and pin location.

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Placement</th>
<th>Legal I/O Standards for DQS</th>
<th>Legal I/O Standards for DQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR3 SDRAM</td>
<td>Row I/O</td>
<td>Differential 1.5-V SSTL Class I</td>
<td>1.5-V SSTL Class I</td>
</tr>
<tr>
<td>DDR3 SDRAM</td>
<td>Column I/O</td>
<td>Differential 1.5-V SSTL Class I</td>
<td>1.5-V SSTL Class I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential 1.5-V SSTL Class II</td>
<td>1.5-V SSTL Class II</td>
</tr>
<tr>
<td>DDR2 SDRAM</td>
<td>Any</td>
<td>SSTL-18 Class I</td>
<td>SSTL-18 Class I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SSTL-18 Class II</td>
<td>SSTL-18 Class II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential 1.8V SSTL Class I</td>
<td>SSTL-18 Class I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Differential 1.8V SSTL Class II</td>
<td>SSTL-18 Class II</td>
</tr>
<tr>
<td>DDR SDRAM</td>
<td>Any</td>
<td>SSTL-2 Class I</td>
<td>SSTL-2 Class I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SSTL-2 Class II</td>
<td>SSTL-2 Class II</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Placement</th>
<th>Legal I/O Standards for DQS</th>
<th>Legal I/O Standards for DQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>QDRII and QDR II + SRAM</td>
<td>Any</td>
<td>HSTL-1.5 Class I HSTL-1.8 Class I</td>
<td>HSTL-1.5 Class I HSTL-1.8 Class I</td>
</tr>
<tr>
<td>RLDRAM II</td>
<td>Any</td>
<td>HSTL-1.5 Class I HSTL-1.8 Class I</td>
<td>HSTL-1.5 Class I HSTL-1.8 Class I</td>
</tr>
</tbody>
</table>
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.
This chapter 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.

Use the Pin Planner feature in the Quartus II software to accurately specify the board trace model information for all your address and command and memory clock output pins. 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 stobes 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 (tDQSCCK, tDSS, tDSH) and board skew (tEXT) 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 2–1).

Figure 2–1. 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 on 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 2–2).

Figure 2–2. Physical Synthesis Optimizations
This chapter explains two timing deration methodologies for multiple chip-select DDR2 and DDR3 SDRAM designs. 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 web site.

Timing deration in this chapter 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 chapter assumes you know how to obtain data on PCB simulations for timing deration from HyperLynx or any other PCB simulator.

Background

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) improve signal integrity (and hence potentially clock rates and overall memory size) by electrically buffering the signals with a register, at a cost of an extra clock of increased latency. In addition, most RDIMMs come with error correction coding (ECC) as standard.

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 3–1). Figure 3–1 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 Synopsis design constraints file (.sdc) file.

Board trace models in the Quartus II software can account for the effects for a single rank, but cannot analyze effects caused by multiple chip selects.
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\text{DS}, t\text{DH}, t\text{IS}, and t\text{IH}).

Figure 3–1. 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, Stratix IV, and Stratix V devices, refer to the Board Settings page in the ALTMEMPHY- or UniPHY-based controller parameter editors.
Implementing Multiple Chip Selects in the 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 tDS, tDH, tIS, tIH values, because the settings on the **Board Settings** page account for multiple chip select slew rate deration.

Timing Deration using the Board Settings

When you target Arria II GX, Arria II GZ, 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 wizard 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 tDS, tDH, tIS, tIH 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.

**Intersymbol Interference**

ISI parameters are similarly autopopulated 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 file and they are used during timing analysis.

**Board Skews**

Table 3–1 describes the types of board skew.

### Table 3–1. Board Skews (Part 1 of 2)

<table>
<thead>
<tr>
<th>Board Skew</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altmemphy</td>
<td>UniPHY</td>
</tr>
<tr>
<td>Minimum CK/DQS skew to DIMM</td>
<td>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.</td>
</tr>
<tr>
<td>Maximum CK/DQS skew to DIMM</td>
<td>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.</td>
</tr>
<tr>
<td>Maximum skew between DIMMs</td>
<td>Maximum delay difference between DIMMs/devices</td>
</tr>
<tr>
<td>Maximum skew within DQS group</td>
<td>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.</td>
</tr>
</tbody>
</table>
Implementing Multiple Chip Selects in the Designs

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 derive 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.

Table 3–1. Board Skews (Part 2 of 2)

<table>
<thead>
<tr>
<th>Board Skew</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum skew between DQS groups</td>
<td>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.</td>
</tr>
<tr>
<td>Address and command to CK skew</td>
<td>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.</td>
</tr>
<tr>
<td>—</td>
<td>The largest skew between the Address/Command signals.</td>
</tr>
</tbody>
</table>
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 3–9.

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 3–2).

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.
4. Enter the Report DDR results from Quartus II timing analysis into the Excel-based calculator (Figure 3–3).

**Figure 3–3. Calculator**

![Calculator Image]

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 3–2 suggests 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 3–2. Data Rates and Stimulus Patterns**

<table>
<thead>
<tr>
<th>Memory Interface</th>
<th>CLK and DQS Toggling Pattern (MHz)</th>
<th>DQ PRBS Pattern (MHz)</th>
<th>Address and Command PRBS Pattern (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2 SDRAM (with a half-rate controller)</td>
<td>400</td>
<td>400</td>
<td>100</td>
</tr>
<tr>
<td>DDR2 SDRAM (with a full-rate controller)</td>
<td>300</td>
<td>300</td>
<td>150</td>
</tr>
<tr>
<td>DDR3 SDRAM (with a half-rate controller)</td>
<td>533</td>
<td>533</td>
<td>133</td>
</tr>
</tbody>
</table>

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 3–4).
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 3–4. ISI and Slew Rate Values

<table>
<thead>
<tr>
<th>Intersymbol Interference</th>
<th>Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Command eye reduction on the setup</td>
<td>0.013</td>
</tr>
<tr>
<td>Address Command eye reduction on the hold</td>
<td>0.013</td>
</tr>
<tr>
<td>DG eye reduction</td>
<td>0.048</td>
</tr>
<tr>
<td>Variation in DQS arrival time</td>
<td>0.003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Slew Rates Deration</th>
<th>V(\text{LIS})</th>
</tr>
</thead>
<tbody>
<tr>
<td>DG</td>
<td>1</td>
</tr>
<tr>
<td>DQS (differential)</td>
<td>2</td>
</tr>
<tr>
<td>Address/Command</td>
<td>1</td>
</tr>
<tr>
<td>CK (differential)</td>
<td>2</td>
</tr>
<tr>
<td>extra IDQ</td>
<td>0</td>
</tr>
<tr>
<td>extra IDH</td>
<td>0</td>
</tr>
<tr>
<td>extra ND</td>
<td>0</td>
</tr>
<tr>
<td>extra NH</td>
<td>0</td>
</tr>
</tbody>
</table>

7. Enter the topology slew rates into the slew rate deration section. The calculator calculates the extra t\(\text{DS}, \text{DH}, \text{IS}, \text{IH}\) 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 3–5).

Figure 3–5. Board Skew Values

<table>
<thead>
<tr>
<th>Board Skews</th>
<th>Time (in ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum skew between DQMMs</td>
<td>0.06</td>
</tr>
<tr>
<td>Minimum CK/DQS to one DQMM</td>
<td>0.01</td>
</tr>
<tr>
<td>Maximum CK/DQS to other DQMM</td>
<td>0.03</td>
</tr>
</tbody>
</table>

The Excel-based calculator then outputs the final derated numbers for your multiple chip select design.

Figure 3–6. Derated Setup and Hold Values

<table>
<thead>
<tr>
<th>Final Multi Chip Select Results</th>
<th>Setup Slack</th>
<th>Hold Slack</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Command</td>
<td>0.01</td>
<td>0.184</td>
</tr>
<tr>
<td>Wait State Address/Command</td>
<td>0.326</td>
<td>0.187</td>
</tr>
<tr>
<td>phy</td>
<td>0.166</td>
<td>0.026</td>
</tr>
<tr>
<td>phy/NoDly</td>
<td>0.176</td>
<td>0.293</td>
</tr>
<tr>
<td>Read/Write</td>
<td>0.919</td>
<td>0.390</td>
</tr>
<tr>
<td>Read/Write</td>
<td>0.179</td>
<td>0.119</td>
</tr>
<tr>
<td>Write</td>
<td>-0.039</td>
<td>-0.013</td>
</tr>
<tr>
<td>Write</td>
<td>0.126</td>
<td>0.070</td>
</tr>
<tr>
<td>Write</td>
<td>0.048</td>
<td>0.112</td>
</tr>
</tbody>
</table>

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.
This chapter provides additional information about the document and Altera.

**Revision History**

The following table shows the revision history for this section.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2011</td>
<td>3.0</td>
<td>Updated for 11.0 release.</td>
</tr>
<tr>
<td>December 2010</td>
<td>2.1</td>
<td>Added Arria II GZ and Stratix V, updated board skew table.</td>
</tr>
<tr>
<td>July 2010</td>
<td>2.0</td>
<td>Added information about UniPHY-based IP and controllers.</td>
</tr>
<tr>
<td>January 2010</td>
<td>1.2</td>
<td>Corrected minor typos.</td>
</tr>
<tr>
<td>December 2009</td>
<td>1.1</td>
<td>Added Timing Deration chapter.</td>
</tr>
<tr>
<td>November 2009</td>
<td>1.0</td>
<td>First published.</td>
</tr>
</tbody>
</table>

**How to Contact Altera**

To locate the most up-to-date information about Altera products, refer to the following table.

<table>
<thead>
<tr>
<th>Contact (1)</th>
<th>Contact Method</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical support</td>
<td>Website</td>
<td><a href="http://www.altera.com/support">www.altera.com/support</a></td>
</tr>
<tr>
<td>Technical training</td>
<td>Website</td>
<td><a href="http://www.altera.com/training">www.altera.com/training</a></td>
</tr>
<tr>
<td>Product literature</td>
<td>Website</td>
<td><a href="http://www.altera.com/literature">www.altera.com/literature</a></td>
</tr>
<tr>
<td>Non-technical support</td>
<td>Email</td>
<td><a href="mailto:nacomp@altera.com">nacomp@altera.com</a></td>
</tr>
<tr>
<td>(General)</td>
<td>Email</td>
<td><a href="mailto:authorization@altera.com">authorization@altera.com</a></td>
</tr>
<tr>
<td>(Software Licensing)</td>
<td>Email</td>
<td><a href="mailto:authorization@altera.com">authorization@altera.com</a></td>
</tr>
</tbody>
</table>

**Note to Table:**
(1) You can also contact your local Altera sales office or sales representative.

**Typographic Conventions**

The following table shows the typographic conventions this document uses.

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold Type with Initial Capital Letters</strong></td>
<td>Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, <strong>Save As</strong> dialog box. For GUI elements, capitalization matches the GUI.</td>
</tr>
<tr>
<td><strong>bold type</strong></td>
<td>Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, <code>\qdesigns</code> directory, D: drive, and <code>chiptrip.gdf</code> file.</td>
</tr>
</tbody>
</table>
### Visual Cue | Meaning
---|---
**Italic Type with Initial Capital Letters** | Indicate document titles. For example, *Stratix IV Design Guidelines*.
*italic type* | Indicates variables. For example, \( n + 1 \).
Variable names are enclosed in angle brackets (<>). For example, `<file name>` and `<project name>.pof` file.
Initial Capital Letters | Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.
“Subheading Title” | Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”
Courier type | Indicates signal, port, register, bit, block, and primitive names. For example, `data1`, `tdi`, and `input`. The suffix `n` denotes an active-low signal. For example, `resetn`.
Indicates command line commands and anything that must be typed exactly as it appears. For example, `c:\qdesigns\tutorial\chiptrip.gdf`.
Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword `SUBDESIGN`), and logic function names (for example, `TRI`).
| An angled arrow instructs you to press the Enter key.
1., 2., 3., and a., b., c., and so on | Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
| Bullets indicate a list of items when the sequence of the items is not important.
| The hand points to information that requires special attention.
| The feet direct you to another document or website with related information.
| A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
| A warning calls attention to a condition or possible situation that can cause you injury.
| The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.
Chapter 2. Debugging Hardware

Debugging Checklist .................................................. 2–1
Quartus II Resource and Planning Issue .......................... 2–2
  Quartus II Resource and Planning Issue Characteristics ... 2–2
Resource Issue Evaluation ........................................... 2–2
  Dedicated IOE DQS Group Resources and Pins ............. 2–2
  Dedicated DLL Resources ........................................ 2–3
Specific PLL Resources ............................................... 2–3
Specific Global, Regional and Dual-Regional Clock Net Resources ... 2–4
Planning Issue Evaluation ........................................... 2–4
Performance, Efficiency, and Bottleneck Issues ................. 2–5
  Performance Issues .............................................. 2–5
  Bottleneck and Efficiency Issues ............................... 2–5
Functional Issues ..................................................... 2–6
  Functional Issue Characteristics ............................... 2–6
  Functional Issue Evaluation .................................... 2–6
  Correct Combination of the Quartus II Software and ModelSim-Altera Device Models ... 2–7
  Altera IP Memory Model ....................................... 2–7
  Vendor Memory Model ......................................... 2–7
  Out of PC Memory Issues .................................... 2–8
  Transcript Window Messages .................................. 2–8
Simulation ............................................................. 2–9
  Modifying the Example Driver to Replicate the Failure ... 2–10
Timing Issues ............................................................ 2–10
  Timing Issue Characteristics ................................. 2–11
  Timing Issue Evaluation ....................................... 2–11
  FPGA Timing Issues .......................................... 2–11
  External Memory Interface Timing Issues .................... 2–12
Hardware Debugging in the Laboratory ......................... 2–13
  Create a Simplified Design that Demonstrates the Same Issue ... 2–13
  Measure Power Distribution Network ....................... 2–13
  Measure Signal Integrity and Setup and Hold Margin ...... 2–13
  Vary Voltage .................................................... 2–14
Use Freezer Spray and Heat Gun .................................. 2–14
  Operate at a Lower Speed ................................... 2–14
Find Out if the Issue Exists in Previous Versions of Software ... 2–14
Find Out if the Issue Exists in the Current Version of Software ... 2–14
Try A Different PCB ............................................ 2–15
Try Other Configurations ......................................... 2–15
Catagorizing Hardware Issues .................................... 2–15
Signal Integrity Issues ............................................... 2–16
  Characteristics ............................................... 2–16
  Evaluating Signal Integrity Issues ......................... 2–16
Hardware and Calibration Issues ............................... 2–18
  Hardware and Calibration Issue Characteristics ........... 2–18
  Evaluating Hardware and Calibration Issues ................ 2–18

Evaluating Hardware and Calibration Issues .................... 2–18
Hardware and Calibration Issue Characteristics ................ 2–18
Evaluating Hardware and Calibration Issues .................... 2–18

June 2011  Altera Corporation
External Memory Interface Handbook Volume 4
Section III. Debugging
Chapter 3. ALTMEMPHY Calibration Stages

Intermittent Issues ................................................................. 2–21
Intermittent Issue Evaluation .................................................... 2–21
Monitoring Signals with the SignalTap II Logic Analyzer .................... 2–22
DDR, DDR2, and DDR3 ALTMEMPHY Designs ................................. 2–22
UniPHY Designs ................................................................. 2–24

Chapter 4. Debug Toolkit for DDR2 and DDR3 SDRAM Controllers with ALTMEMPHY IP

Debug Toolkit Overview .......................................................... 4–1
Install the Debug Toolkit ......................................................... 4–2
Modify the Example Top-Level File to use the Debug Toolkit .................. 4–2
Verify the Design ................................................................. 4–3
Regenerate the IP ................................................................. 4–4
Instantiate the JTAG Avalon-MM port in to the Example-Top Level Project .... 4–4
Add Additional Signals .......................................................... 4–5
Add alt_jtagavalon.v to your Quartus II Project Settings Files List ............... 4–7
Recompile your Quartus II Test Design ....................................... 4–7
Program Hardware with Debug Enabled .sof ................................ 4–7
Use the Debug Toolkit .......................................................... 4–8
Interpret the Results ............................................................ 4–9
Calibration Successful ......................................................... 4–9
Calibration Fails ................................................................. 4–12
Save the Calibration Results .................................................. 4–13
Understand the Checksum and Failure Code ................................ 4–15

Chapter 5. UniPHY Calibration Stages

Overview ................................................................. 5–1
Calibration Stages ........................................................... 5–2
Section III. Debugging

Assumptions ................................................................. 5–2
Memory Initialization ....................................................... 5–2
Stage 1: Read Calibration Part One—DQS Enable Calibration and DQ/DQS Centering ............... 5–2
  Guaranteed Write ......................................................... 5–3
  DQS Enable Phase Calibration ..................................... 5–4
  Centering DQ/DQS ....................................................... 5–6
Stage 2: Write Calibration Part One ...................................... 5–7
Stage 3: Read Calibration Part Two—Read Latency Minimization ........................................... 5–8
  Read Latency Tuning ..................................................... 5–9
Stage 4: Write Calibration Part Two—DQ/DQS Centering ...................................................... 5–9
Stage 5: Diagnostic Test ....................................................... 5–9
Calibration Signals .......................................................... 5–9

Chapter 6. DDR2 and DDR3 SDRAM Controllers with UniPHY EMIF Toolkit

Feature Description .......................................................... 6–1
Using the External Memory Interface Toolkit ........................................... 6–1
Enabling Communication with the Controller via the CSR Port ......................... 6–2
Launching the External Memory Interface Debug Toolkit ........................................ 6–4
Specifying Project Settings ................................................ 6–4
Viewing Information About Your External Memory Interface .................................. 6–5
Interpreting Results and Troubleshooting ...................................................... 6–6
  Calibration Successful ................................................... 6–6
  Calibration Failed ......................................................... 6–6

Additional Information

Document Revision History ................................................ Info–1
How to Contact Altera ....................................................... Info–1
Typographic Conventions .................................................. Info–2
1. Verifying Functionality using the SignalTap II Logic Analyzer

The SignalTap® II logic analyzer shows read and write activity in the system.

For more information on 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 (\(<variation name>_example_top|<variation name>_inst|<variation name>_controller_phy:<variation name>_controller_phy_inst|phy_clk|phy_clk\) 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.
This chapter describes the process of debugging hardware and the tools to debug any external memory interface. The concepts contained 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.

Use this document with the respective IP user guide and other sections in the External Memory Interface Handbook.

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.

### 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.

<table>
<thead>
<tr>
<th>Check</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>☐</td>
<td>Try a different fit.</td>
</tr>
<tr>
<td>☐</td>
<td>Check IP parameters at the operating frequency ($t_{MRD}$, $t_{WTR}$ for example).</td>
</tr>
<tr>
<td>☐</td>
<td>Ensure you have constrained your design with proper timing deration and have closed timing.</td>
</tr>
<tr>
<td>☐</td>
<td>Simulate the design. If it fails in simulation, it will fail in hardware.</td>
</tr>
<tr>
<td>☐</td>
<td>Analyze timing.</td>
</tr>
<tr>
<td>☐</td>
<td>Place and assign $R_{UP}$ and $R_{DN}$ (OCT).</td>
</tr>
<tr>
<td>☐</td>
<td>Measure the power distribution network (PDN).</td>
</tr>
<tr>
<td>☐</td>
<td>Measure signal integrity.</td>
</tr>
<tr>
<td>☐</td>
<td>Measure setup and hold timing.</td>
</tr>
<tr>
<td>☐</td>
<td>Measure FPGA voltages.</td>
</tr>
<tr>
<td>☐</td>
<td>Vary voltages.</td>
</tr>
<tr>
<td>☐</td>
<td>Heat and cool the PCB.</td>
</tr>
<tr>
<td>☐</td>
<td>Operate at a lower or higher frequency.</td>
</tr>
<tr>
<td>☐</td>
<td>Check board timing and trace Information.</td>
</tr>
<tr>
<td>☐</td>
<td>Check LVDS and clock sources, I/O voltages and termination.</td>
</tr>
</tbody>
</table>
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.

**Quartus II Resource and Planning Issue**

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.

**Quartus II Resource and Planning Issue Characteristics**

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 Volume 6: Design Flow Tutorials of the External Memory Interface Handbook.

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.

Performance, Efficiency, and Bottleneck Issues

This topic describes the performance \( f_{\text{MAX}} \), 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 \( f_{\text{MAX}} \) 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 Timing Analysis section in volume 4 of the External Memory Interface Handbook.

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 \( f_{\text{MAX}} \) 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 practice 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 vendor's 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.

#### Functional Issue Characteristics

Functional issues 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
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 Jede specification. After you resolve functional issues, you can start testing hardware.

For more information about simulating, refer to the Simulation section in volume 4 of the External Memory Interface Handbook.

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.
Derating 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 constraints, refer to Volume 3: Implementing Altera Memory Interface IP of the External Memory Interface Handbook.

Hardware Debugging in the Laboratory

Before starting to debug, confirm the design followed the Altera recommended design flow.

Refer to the Recommended Design Flow section 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 at a lower speeds:

- tMRD
- tWTR
- 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.

### 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 bi-directional. 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, RUP or RDN 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 RUP and RDN 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 RUP and RDN pins in either the projects HDL port list, or in the assignment editor (termination_blk0~_rup_pad/ termination_blk0~_rdn_pad).
- Connect the RUP and RDN pins to the correct resistors and pull-up and pull-down voltage in the schematic or PCB.
- Contain the RUP and RDN 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 \texttt{ctl\_cal\_fail} signal.
- Hardware issues result in read and write failures, which typically results in the design asserting the pass not fail (\texttt{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.

Refer to Chapter 4, Debug Toolkit for DDR2 and DDR3 SDRAM Controllers with ALTMEMPHY IP.
Refer to the respective user guide for information on which signals you should use during debugging. 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:

```vhdl
//w ire [7:0] Probe_sig;
wire [5:0] SourceSig;
PhaseCount PhaseCounter ( 
  .resetn (1'b1),
  .clock (pll_ref_clk),
  .step (SourceSig[5]),
  .updown (SourceSig[4]),
  .offset (ProbeSig)
);
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]),
 .phasetep (Source_sig[5]),
 .phasedown (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 Chapter 4, Debug Toolkit for DDR2 and DDR3 SDRAM Controllers with ALTMEMPHY IP.

Resynchronization Timing Margin
Observe the size and margins, available for resynchronization using the debug toolkit or system console.

Refer to Chapter 4, Debug Toolkit for DDR2 and DDR3 SDRAM Controllers with ALTMEMPHY IP.

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 the debug toolkit or system console.

Refer to Chapter 4, Debug Toolkit for DDR2 and DDR3 SDRAM Controllers with ALTMEMPHY IP

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.

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: Implementing Altera Memory Interface IP 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
- Phasecountersel * -altpll_component
- Phasetopdown -altpll_component
- Phasetop -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`
- `qvl_d_wr_address*`
- `qvl_d_rd_address*`
3. **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. **Advertise 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 3–1 on page 3–2 shows a flow chart of the calibration stages for the ALTMEMPHY IP.
Figure 3-1. Calibration Stages

1. **Enter Calibration**
   - Initialize PHY

2. **Initialize PHY**
   - Initialize DRAM Power-Up Sequence

3. **Program Mode Registers for Calibration**
   - All CS MR prog?
     - yes
     - no

4. **Write Header Information in the IRAM**
   - s_cal
     - Start of Calibration

5. **s_cal**
   - Calibrate?
     - yes
     - no

6. **Calibrate**
   - Write Burst Training Pattern
   - BTP Write to all CS?
     - yes
     - no

7. **Write More Training Patterns**
   - All CS written?
     - yes
     - no

8. **Initialize Read Resynchronization Phase Calculation**

9. **Calculate Read Resynchronization Phase**

10. **Calculate Read Data Valid Window**

11. **Initialize Read Resynchronization Phase Calculation**

12. **Calculate Read Resynchronization Phase**

13. **All CS Sweep?**
    - yes
    - no

14. **Calculate Read Resynchronization Phase**

15. **Advertise Write Latency**

16. **Calculate Read Latency**

17. **Write Datapath Setup**

18. **ac_H setting ok?**
    - yes
    - no

19. **Calibrate Postamble**

20. **Setup the Mimic Window**

21. **Write User Mode Register Settings**

22. **All CS MR prog?**
    - yes
    - no

23. **s_operational**
    - Calibration successful

24. **Tracking Due?**
    - yes
    - no

25. **Perform Tracking**

26. **Calibration Error Detected**

27. **s_non_operational**
    - Calibration not successful

**Legend**:
- Initialization
- Load Training Patterns
- Setup Read Datapath
- Setup Write Datapath
- Setup Tracking
- User Mode
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.

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.

Refer to Chapter 4, Debug Toolkit for DDR2 and DDR3 SDRAM Controllers with ALTMEMPHY IP

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. The ALTMEMPHY IP requires this pattern to match the characterization behavior for non-DQS 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 clock 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 3–2) while driving a count of frequency controller clock rate to the DQ pins (Figure 3–2), 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 3–2. Description of Write Pattern**

<table>
<thead>
<tr>
<th>controller</th>
<th>clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>address</td>
<td>48</td>
</tr>
<tr>
<td>command</td>
<td>NOP</td>
</tr>
<tr>
<td>write data</td>
<td>-two</td>
</tr>
</tbody>
</table>

**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.

\[ \text{The read data from the DRAM is not important here. The count is performed between the issue of } doing_{rd} \text{ and } rdata_{valid} \text{ returning.} \]

**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 3–2) 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 3–3 shows the calibration timing diagram.

**Figure 3–3. Calibration Timing**

![Calibration Timing Diagram](Note_to_Figure_3-3.png)

**Note to Figure 3–3:**
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 oscillator (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.
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 Chapter 2, Debugging Hardware of the External Memory Interfaces 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.
- Provides possible causes for calibration failures.
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 design flow, refer to the Recommended Design Flow section 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_addr cmd_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 RUP/RDN 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:
- \texttt{ctl_rlat} *DT-phy\_inst
- \texttt{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
- \texttt{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 4–1 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 \texttt{<variation\_name>\_alt\_mem\_phy\_seq.vhd} file.

All signals are active high.

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flag_done_timeout</td>
<td>Calibration stage timeout failure, memory did not respond.</td>
</tr>
<tr>
<td>Flag_ack_timeout</td>
<td>Sequencer failed to respond.</td>
</tr>
<tr>
<td>state.s_phy_initialise</td>
<td>PHY initialization Stage: wait for DLL lock and init_done.</td>
</tr>
<tr>
<td>state.s_init_dram</td>
<td>DRAM initialization stage: reset sequence.</td>
</tr>
<tr>
<td>State.s_prog_cal_mrs</td>
<td>DRAM initialization stage: programming mode registers (once per chip select).</td>
</tr>
<tr>
<td>state.s_write_ihi</td>
<td>Write internal RAM header initialization.</td>
</tr>
<tr>
<td>state.s_cal</td>
<td>Calibration required stage.</td>
</tr>
<tr>
<td>state.s_write_btp</td>
<td>Write block training pattern stage: 00001111.</td>
</tr>
<tr>
<td>state.s_write_mtp</td>
<td>Write memory training patterns: 00110101.</td>
</tr>
<tr>
<td>state.s_rrp_reset</td>
<td>Read resynchronization phase reset: PLL initial condition.</td>
</tr>
<tr>
<td>state.s_rrp_sweep</td>
<td>Read resynchronization phase sweep: sweep PLL phases per chip select.</td>
</tr>
<tr>
<td>state.s_read_mtp</td>
<td>Read memory training patterns to find correct alignment.</td>
</tr>
<tr>
<td>State.s_rrp_seek</td>
<td>Read resynchronization phase setup stage: set PLL to center of valid window.</td>
</tr>
<tr>
<td>state.s_rdv</td>
<td>Read data valid stage.</td>
</tr>
<tr>
<td>state.s_poa</td>
<td>Postamble calibration stage.</td>
</tr>
<tr>
<td>state.s_was</td>
<td>Write datapath setup: write data to DRAM so that latency can be determined.</td>
</tr>
<tr>
<td>state.s_adv_rd_lat</td>
<td>Advertise read latency stage.</td>
</tr>
</tbody>
</table>
Modify the Example Top-Level File to use the Debug Toolkit

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.* 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.

<table>
<thead>
<tr>
<th>Port Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state.s_adv_wr_lat</td>
<td>Advertise write latency stage.</td>
</tr>
<tr>
<td>state.s_tracking_setup</td>
<td>Tracking setup stage (first pass to setup mimic window).</td>
</tr>
<tr>
<td>state.s_prep_customer_mr_setup</td>
<td>Set custom mode register settings (admin).</td>
</tr>
<tr>
<td>state.s_tracking</td>
<td>Tracking stage (mimic path tracking in user mode).</td>
</tr>
<tr>
<td>state.s_operational</td>
<td>Calibration success: user mode.</td>
</tr>
<tr>
<td>state.s_non_operational</td>
<td>Calibration failed or tracking failed in user mode.</td>
</tr>
<tr>
<td>state.s_reset</td>
<td>Reset stage.</td>
</tr>
<tr>
<td>dgrb_ctrl.command_err</td>
<td>Error in the data gather read bias block.</td>
</tr>
<tr>
<td>dgrb_ctrl.command_result[7..0]</td>
<td>Data gather read block (DGRB) error code.</td>
</tr>
<tr>
<td>dgrb_ctrl.command_err</td>
<td>Error in the data gather write bias block.</td>
</tr>
<tr>
<td>dgwb_ctrl.command_result[7..0]</td>
<td>Data gather write block (DGWB) error code.</td>
</tr>
<tr>
<td>admin_ctrl.command_err</td>
<td>Error in the admin (DRAM initialization and control) block.</td>
</tr>
<tr>
<td>admin_ctrl.command_result[7..0]</td>
<td>Admin block error code.</td>
</tr>
</tbody>
</table>
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 4–1).

3. Navigate down the hierarchy and click on the Avalon-MM JTAG node (Figure 4–2).

   If you encounter connection problems, Altera recommends that you have only a single USB-Blaster™ download cable programming adaptor connected to your PC.

4. If you receive a prompt stating the following message, verify you have the latest debug toolkit, and click **Yes** (Figure 4–3).
Chapter 4: Debug Toolkit for DDR2 and DDR3 SDRAM Controllers with ALTMEMPHY IP

Interpret the Results

This topic discusses:

- Calibration Successful
- Calibration Fails

Calibration Successful

If calibration is successful, you see the following screen (Figure 4–4).

Figure 4–4. Calibration Successful
For optimum operation of the debug toolkit, ensure that you turn on **Enable Debug** (Figure 4–5).

**Figure 4–5. Enable Debug**

Click **internal RAM** to display the calibration memory results (Figure 4–6).

This setting is not typically used.

**Figure 4–6. Calibration Memory Results**

The debug toolkit can dynamically alter the PLL clock phases (Figure 4–7).
This setting is not typically used.

Figure 4–7. 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 4–7 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 4–7, 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 4–8) 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

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 4–9), to show the exact calibration configuration of the generated IP.

---

**Figure 4–8. Visualization**

![Visualization](image1.png)

**Figure 4–9. PHY Parameterization**

![PHY Parameterization](image2.png)
Calibration Fails

If calibration fails, you see the following screen (Figure 4–10).

![Calibration Fails](image)

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 4–15.

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 4–11).

**Figure 4–11. Export**
In the **Save** dialog box (Figure 4–12), 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 4–12. 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 4–2 shows the codes that correspond to the indicated calibration stages.

**Table 4–2. Calibration Stages**

<table>
<thead>
<tr>
<th>Stage</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter calibration state</td>
<td>0</td>
</tr>
<tr>
<td>PHY initialization</td>
<td>1</td>
</tr>
<tr>
<td>DRAM initialization</td>
<td>2</td>
</tr>
<tr>
<td>Writing header information in the internal RAM</td>
<td>3</td>
</tr>
<tr>
<td>Writing the burst training pattern</td>
<td>4</td>
</tr>
<tr>
<td>Writing more training patterns</td>
<td>5</td>
</tr>
<tr>
<td>Testing more training pattern writes</td>
<td>6</td>
</tr>
<tr>
<td>Read resynchronization phase calibration—reset stage</td>
<td>7</td>
</tr>
<tr>
<td>Read resynchronization phase calibration—sweep stage</td>
<td>8</td>
</tr>
<tr>
<td>Read resynchronization phase calibration—seek stage</td>
<td>9</td>
</tr>
<tr>
<td>Read data valid window calibration</td>
<td>10</td>
</tr>
<tr>
<td>Postamble calibration</td>
<td>11</td>
</tr>
<tr>
<td>Calibration of the write datapath (including finding write latency)</td>
<td>12</td>
</tr>
<tr>
<td>Output of read latency</td>
<td>13</td>
</tr>
<tr>
<td>Output of write latency</td>
<td>14</td>
</tr>
<tr>
<td>Writing of customer mode register settings</td>
<td>15</td>
</tr>
<tr>
<td>Tracking</td>
<td>16</td>
</tr>
</tbody>
</table>

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” on page 3–1.
This chapter describes the calibration stages performed by the DDR2 and DDR3 SDRAM, QDRII and QDRII+ SRAM, and RLDRAM II Controllers with UniPHY. This information is useful in debugging calibration failures. The chapter includes an overview of calibration, explanation of the calibration stages, and a list of generated calibration signals. The information in this chapter applies only to the Nios® II-based sequencer used in the DDR2 and DDR3 SDRAM Controllers with UniPHY versions 10.0 and later, and in the QDRII and QDRII+ SRAM and RLDRAM II Controllers with UniPHY version 11.0. The information in this chapter applies to the Stratix III, Stratix IV, and Stratix V device families.

For QDRII and QDRII+ SRAM and RLDRAM II Controllers with UniPHY version 11.0, you have the option to select either the RTL-based sequencer or the Nios® II-based sequencer.

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 5–1 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 5–1. 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. Read calibration part two—Read latency minimization
4. Write calibration part two—DQ/DQS centering
5. Diagnostic test

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 SDRAM is powered up according to JEDEC initialization specifications. All ranks power up simultaneously. Once powered, the SDRAM 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 3 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
DQS enable calibration and DQ/DQS centering consists of the following actions:

- Guaranteed Write
- DQS Enable Phase Calibration
- DQ/DQS Centering

Figure 5–2 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 5–2. Read Data Path Calibration Model**

![Read Data Path Calibration Model Diagram]

---

**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.

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 5–3 illustrates a guaranteed write of zeros.

**Figure 5–3. Guaranteed Write of Zeros**

![Guaranteed Write of Zeros Diagram]
For DQ[0], the guaranteed write performs the following operations:

- **DDR2 SDRAM**
  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 2, column 0
  d. Writes a full burst of ones to bank 2, column 1

- **DDR3 SDRAM**
  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

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 5–4.

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 5–4. Back to Back Reads on pin DQ[0]**

### DQS Enable Phase Calibration

* Only applicable for DDR2 and DDR3 SDRAM Controllers with UniPHY.

DQS enable phase 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.
The VFIFO generates the DQS enable signal by shifting the controller-generated read data enable signal, \( rdata\_en \), by a number of full-rate clock cycles. The sequencer determines DQS enable signal phases by sweeping the DQS enable signal with coarse delays of \( 360^\circ \) (VFIFO latency) and fine delays of \( 45^\circ \) (phase). Fine delays are VT-compensated.

For each VFIFO latency, the sequencer sweeps through all phases until a read test fails, or until the sweep completes all available phases. For each VFIFO latency and phase setting, the sequencer issues back-to-back reads from column 0 of bank 0 and bank 1, and column 1 of bank 0 and bank 1, as shown in Figure 5–4. Two full bursts are read and compared with the reference data for each phase and delay setting. Figure 5–5 shows the steps taken in DQS enable calibration.

![Figure 5–5. DQS Enable Phase Calibration Steps](image)

Once the sequencer identifies a range of working phases, it center-aligns the falling edge of the DQS enable signal within a valid phase range with respect to the DQS signal. 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 5–6 shows the DQS enable signal and phase relationships. The goal of VFIFO latency and phase sweeping 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 5–6. DQS Enable Signal and Phase Relationships**

<table>
<thead>
<tr>
<th>Row 1</th>
<th>DQS + 90</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 2</td>
<td>dq_enable (inside I/O)</td>
</tr>
<tr>
<td>Row 3</td>
<td>dq_enable aligned</td>
</tr>
<tr>
<td>Row 4</td>
<td>dq_enable (inside I/O)</td>
</tr>
<tr>
<td>Row 5</td>
<td>dq_enable aligned</td>
</tr>
</tbody>
</table>

The following points describe each row of Figure 5–6:

- **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 phase 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 phase 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 phases, beginning from the initial DQS enable of Row 4. The first phase 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 phase 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. In reality, 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 5–7 illustrates a four DQ/DQS group per-bit-deskew and centering.

**Figure 5–7. 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 5–7, 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 5–7, 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.

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 5–8 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 5–8. Write Data Path](image)

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°. 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.

Coarse delays are called *phases*, and fine delays are called *delays*; phases are PVT compensated, delays are not.

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: 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.

Stage 4: 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 5: Diagnostic Test

The diagnostic test is the final stage of calibration before the sequencer passes control to the memory controller. At this stage, the sequencer writes and reads more complex patterns to and from the memory device. The goals of this stage are to verify that calibration is successful and robust, and to estimate the read and write margins under more noisy conditions.

The sequencer writes and reads a sequential 8-bit burst pattern from 00000001 to 11111111 on columns 0 and 1 of bank 0 and column 0 and 1 of bank 1.

Calibration Signals

Table 5–1 lists signals produced by the calibration process.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>afi_cal_fail</td>
<td>Asserts high if calibration fails.</td>
</tr>
<tr>
<td>afi_cal_success</td>
<td>Asserts high if calibration is successful.</td>
</tr>
</tbody>
</table>

Table 5–1. Calibration Signals (Part 1 of 2)
Table 5–1. Calibration Signals (Part 2 of 2)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>afi_cal_debug_info</td>
<td>A 32-bit signal containing additional data relating to the success or failure of calibration. Only 16 bits are used.</td>
</tr>
<tr>
<td></td>
<td>■ In cases of calibration success:</td>
</tr>
<tr>
<td></td>
<td>■ DEBUG_INFO[7:0] is a binary representation of the read margin and DVW width</td>
</tr>
<tr>
<td></td>
<td>■ DEBUG_INFO[15:8] is a binary representation of the write margin and DVW width</td>
</tr>
<tr>
<td></td>
<td>■ In cases of calibration failure:(1)</td>
</tr>
<tr>
<td></td>
<td>■ DEBUG_INFO[7:0] is a binary representation of the stage at which failure occurred</td>
</tr>
<tr>
<td></td>
<td>■ DEBUG_INFO[15:8] is a binary representation of the group that was being calibrated at the time of failure</td>
</tr>
</tbody>
</table>

Notes for Table 5–1:
(1) In cases of calibration failure, DEBUG_INFO values are valid only if failure occurred during the Read Calibration, Write Leveling, or Write Per-bit deskew and Centering stages of calibration. DEBUG_INFO values are not valid if failure occurred during Read Latency Tuning.
This chapter describes the UniPHY External Memory Interface Toolkit. It explains how to enable, launch, and run the toolkit, and provides a guide for interpreting results and troubleshooting. This toolkit is a Tcl-based 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.

This toolkit supports only the DDR2 and DDR3 SDRAM Controllers with UniPHY.

**Feature Description**

The External Memory Interface Toolkit 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.

**Using the External Memory Interface Toolkit**

Using the external memory interface toolkit to analyze your external memory interface involves the following steps:

1. (Optional) Generating your IP core with the CSR port enabled and with the CSR communication interface type properly set.
2. Launching the debug toolkit.
3. Specifying project settings.
4. Using the toolkit to view information about your interface.
5. Interpreting results and troubleshooting your interface.

The following sections discuss each of the above steps in detail.
Enabling Communication with the Controller via the CSR Port

Optionally, you can enable communication between the EMIF toolkit and the memory controller through the Configuration and Status Register (CSR) port on the controller.

You do not have to enable the CSR port in order to use the toolkit’s report-generation functions; however, enabling the CSR port provides the following additional capabilities:

- allows the toolkit to verify memory device operation by determining whether the controller receives DQS edges from the memory device
- allows the toolkit to issue soft resets to the memory interface
- allows the toolkit to monitor PLL locked status

Before the toolkit can communicate with the controller through the CSR port, you must enable the CSR port from the Controller Settings tab of your memory interface parameter editor, as described in the following steps:

1. Open the DDR2 or DDR3 SDRAM Controller with UniPHY parameter editor from the MegaWizard Plug-in Manager, SOPC Builder, or Qsys.
2. Under Advanced Controller Features on the Controller Settings tab, specify the following options:
   a. Turn on Enable Configuration and Status Register Interface.
   b. Set CSR port host interface to INTERNAL_JTAG.
3. Regenerate the IP core.
4. Compile your design in the Quartus II software.
5. Program the Altera FPGA device using the programming file from your project.

Figure 6–1 illustrates the external memory interface components with the optional CSR port and JTAG Avalon master configured.

Figure 6–1. External Memory Interface with CSR Port Configured
Table 6–1 shows the UniPHY and High Performance Controller II CSR address map.

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit</th>
<th>Name</th>
<th>Default</th>
<th>Access</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x001</td>
<td>15:0</td>
<td>Reserved</td>
<td>0</td>
<td>—</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td></td>
<td>31:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x002</td>
<td>15:0</td>
<td>Reserved</td>
<td>0</td>
<td>—</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td></td>
<td>31:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>0</td>
<td>SOFT_RESET</td>
<td>—</td>
<td>W0</td>
<td>Initiates a soft reset of the memory interface. This bit is automatically</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>deasserted when reset is completed.</td>
</tr>
<tr>
<td></td>
<td>23:1</td>
<td>Reserved</td>
<td>0</td>
<td>—</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>AFI_CAL_SUCCESS</td>
<td>—</td>
<td>RO</td>
<td>Reports the value of the UniPHY afi_cal_success output. Writing to this bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>has no effect.</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>AFI_CAL_FAIL</td>
<td>—</td>
<td>RO</td>
<td>Reports the value of the UniPHY afi_cal_fail output. Writing to this bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>has no effect.</td>
</tr>
<tr>
<td></td>
<td>31:26</td>
<td>Reserved</td>
<td>0</td>
<td>—</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>0x005</td>
<td>7:0</td>
<td>FOM_IN</td>
<td>—</td>
<td>RO</td>
<td>The figure of merit (1) for read as calculated by the sequencer. Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>applicable if AFI_CAL_SUCCESS is 1.</td>
</tr>
<tr>
<td></td>
<td>15:8</td>
<td>Reserved</td>
<td>0</td>
<td>—</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td></td>
<td>23:16</td>
<td>FOM_OUT</td>
<td>—</td>
<td>RO</td>
<td>The figure of merit (1) for write as calculated by the sequencer. Only</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>applicable if AFI_CAL_SUCCESS is 1.</td>
</tr>
<tr>
<td></td>
<td>31:24</td>
<td>Reserved</td>
<td>0</td>
<td>—</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>0x006</td>
<td>7:0</td>
<td>INIT_FAILING_STAGE</td>
<td>—</td>
<td>RO</td>
<td>Initial failing error stage of calibration. Only applicable if AFI_CAL_FAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is 1.</td>
</tr>
<tr>
<td></td>
<td>15:8</td>
<td>Reserved</td>
<td>0</td>
<td>—</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td></td>
<td>23:16</td>
<td>INIT_FAILING_GROUP</td>
<td>—</td>
<td>RO</td>
<td>Initial failing error group of calibration. Only applicable if AFI_CAL_FAIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>is 1.</td>
</tr>
<tr>
<td></td>
<td>31:24</td>
<td>Reserved</td>
<td>0</td>
<td>—</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>0x007</td>
<td>31:0</td>
<td>DQS_DETECT</td>
<td>—</td>
<td>RO</td>
<td>Indicates whether DQS edges have been identified for each group. Each bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>corresponds to 1 DQS group.</td>
</tr>
</tbody>
</table>

Note to Table 6–1:

(1) The figure of merit (FOM) is a measure of the health of the read (or write) interface; it is calculated as the sum over all groups of the minimum margin on DQ plus the margin on DQS, divided by 2.
Launching the External Memory Interface Debug Toolkit

To launch the debug toolkit from the Quartus II software, perform the following steps:

1. Program the Altera FPGA device using the programming file from your project.
2. To open the External Memory Interface Toolkit in the System Console, click External Memory Interface Toolkit on the Tools menu in the Quartus II software.
3. On the File menu in the System Console window, select Load Design to load your Quartus II Project File (.qpf) into the EMIF toolkit. When your project is loaded, your design folder appears under design_instances in the System Explorer tree.
4. In the System Explorer tree, right-click the <instance_name>.jdi file in your design folder, and click Link design instance to device to link the design instance to the target device.
5. Under Hardware Setup, click Apply Linked Design to load the project. Your interface connection now appears under New Memory Interface Connection.
6. Under Hardware Setup on the External Memory Interface Toolkit tab, select your hardware and device. The Hardware list shows all the detected connections between your board and the PC; the Device list shows all the devices on your board, detected by the selected connection.
7. If you have more than one interface, select the interface you want from the Memory Interface list, under New Memory Interface Connection.
   Once you have selected your interface, the New Memory Interface Connection group box displays the UniPHY instance name, and—if you have set up communication with the CSR port—the CSR instance name and PLL status.
8. Under New Memory Interface Connection, click Establish Connection to create a tab for your external memory interface.
   You can optionally repeat this step to create tabs for any other external memory interfaces in your design.
9. If you turn on the Efficiency Monitor and Protocol Checker option in your design, under New Efficiency Monitor Connection, select the efficiency monitor instance from the Efficiency Monitor list.
10. Click Establish Connection to create a tab for the efficiency monitor and protocol checker.

Specifying Project Settings

Before rerunning calibration or generating reports, you must establish project settings to allow for correct calibration and margining results. To establish the project settings, perform the following steps:

1. On the Project Settings tab, ensure the Settings Type field is set to Device Settings.
2. Under Project FPGA Settings, select values for the FPGA Family and FPGA Speedgrade.
3. Under Project PLL Settings, type your memory interface frequency.
4. Click Update Project Settings to save the information for your project in memory.

**Viewing Information About Your External Memory Interface**

The following steps explain how to use the External Memory Interface Toolkit to view information about your interface.

1. When you establish a connection for your memory interface, a memory interface tab appears in the External Memory Interface Toolkit parameter editor.

2. In the memory interface tab, go to Memory Interface Status and Control tab and click Generate Calibration Report to generate detailed calibration information for your interface.

3. To view the calibration report, on the Reports tab, under Report Type, select Calibration report per DQ group or Calibration report.

The Calibration report per DQ group lists read and write data valid windows and calibration status for each group and rank.

The Calibration report is comprehensive, and includes the following information:

- Group and rank mask status
- Calibration status
- Margins observed during calibration, per DQ group
- DQ and DQS I/O settings

4. To view read and write margining and data valid window information, click Generate margining report.


The Margining report lists each DQ pin and DQS group margin observed following calibration. Any reduced margins on DQ pins can be indicative of possible problems with the PCB layout of the memory interface.

The DVW report is a graphical representation of the Margining report, and can help you visualize the range of the data valid window per DQ pin. The black line in the report represents the point within the window to which the DQ pin is calibrated.

6. Click Save Report to save all generated reports in HTML format on your computer.

   ![Note] You cannot save the DVW report.

7. On the Summary tab, select Connection Summary to review the interface and its connection properties.

8. If you have turned on the Efficiency Monitor and Protocol Checker option in your design, and established the efficiency monitor connection, an efficiency monitor tab appears in the External Memory Interface Toolkit parameter editor.
9. In the efficiency monitor tab, use the Efficiency Monitor Controls parameters to do the following tasks:
   ■ Start or stop the efficiency monitor
   ■ Reset the efficiency monitor
   ■ Reset the protocol checker
   ■ Read the efficiency monitor data
10. On the Summary tab, you get the following details:
   ■ Interface and efficiency monitor connection properties
   ■ Efficiency monitor properties summary
   ■ Efficiency monitor statistics summary
   ■ Protocol checker summary
11. Select Result Summary to display read and write latency values, as well as the DQS Captured Status. (If your design does not contain a CSR port, or you have not enabled the CSR port for use, the DQS Captured Status is not available.)

For information about using the CSR port, refer to “Enabling Communication with the Controller via the CSR Port” on page 6–2.

Interpreting Results and Troubleshooting

This section provides information on how to interpret the results returned by the toolkit.

Calibration Successful
When both calibration and post-calibration tests complete successfully, you can use the debug toolkit to retrieve information about your memory interface and to identify areas of reduced margin. DQ pins with small margins tend to be the pins most susceptible to the effects of data corruption, board delays, and noise.

Calibration Failed
In the event of calibration failure, you should check the DQS Capture Status in the Summary results to verify that the memory interface is functioning. If DQS edges are not detected, it is likely that the memory interface is not functioning. If DQS edges are detected, refer to Figure 6–2 to assist in troubleshooting your design.
Figure 6–2 shows a flowchart of debugging tips to assist in resolving calibration failure.

**Figure 6–2. Debugging Tips**

For detailed information about each calibration stage, refer to Chapter 5, “UniPHY Calibration Stages”. 
This chapter provides additional information about the document and Altera.

Document Revision History

The following table shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2011</td>
<td>3.0</td>
<td>Removed leveling information from ALTMEMPHY Calibration Stages chapters.</td>
</tr>
<tr>
<td>December 2010</td>
<td>2.1</td>
<td>■ Added new chapter: UniPHY Calibration Stages</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Added new chapter: DDR2 and DDR3 SDRAM Controllers with UniPHY EMIF Toolkit</td>
</tr>
<tr>
<td>July 2010</td>
<td>2.0</td>
<td>Updated for 10.0 release.</td>
</tr>
<tr>
<td>January 2010</td>
<td>1.2</td>
<td>Corrected minor typos.</td>
</tr>
<tr>
<td>December 2009</td>
<td>1.1</td>
<td>Added Debug Toolkit for DDR2 and DDR3 SDRAM High-Performance Controllers chapter and ALTMEMPHY Calibration Stages chapter.</td>
</tr>
<tr>
<td>November 2009</td>
<td>1.0</td>
<td>First published.</td>
</tr>
</tbody>
</table>

How to Contact Altera

To locate the most up-to-date information about Altera products, refer to the following table.

<table>
<thead>
<tr>
<th>Contact (1)</th>
<th>Contact Method</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technical support</td>
<td>Website</td>
<td><a href="http://www.altera.com/support">www.altera.com/support</a></td>
</tr>
<tr>
<td>Technical training</td>
<td>Website</td>
<td><a href="http://www.altera.com/training">www.altera.com/training</a></td>
</tr>
<tr>
<td></td>
<td>Email</td>
<td><a href="mailto:custrain@altera.com">custrain@altera.com</a></td>
</tr>
<tr>
<td>Product literature</td>
<td>Website</td>
<td><a href="http://www.altera.com/literature">www.altera.com/literature</a></td>
</tr>
<tr>
<td>Non-technical support (General)</td>
<td>Email</td>
<td><a href="mailto:nacomp@altera.com">nacomp@altera.com</a></td>
</tr>
<tr>
<td>(Software Licensing)</td>
<td>Email</td>
<td><a href="mailto:authorization@altera.com">authorization@altera.com</a></td>
</tr>
</tbody>
</table>

Note to Table:

(1) You can also contact your local Altera sales office or sales representative.
# Typographic Conventions

The following table shows the typographic conventions this document uses.

<table>
<thead>
<tr>
<th>Visual Cue</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bold Type with Initial Capital Letters</strong></td>
<td>Indicate command names, dialog box titles, dialog box options, and other GUI labels. For example, Save As dialog box. For GUI elements, capitalization matches the GUI.</td>
</tr>
<tr>
<td><strong>bold type</strong></td>
<td>Indicates directory names, project names, disk drive names, file names, file name extensions, software utility names, and GUI labels. For example, \qdesigns directory, D: drive, and chiptrip.gdf file.</td>
</tr>
<tr>
<td><strong>Italic Type with Initial Capital Letters</strong></td>
<td>Indicate document titles. For example, Stratix IV Design Guidelines.</td>
</tr>
<tr>
<td><strong>italic type</strong></td>
<td>Indicates variables. For example, ( n + 1 ). Variable names are enclosed in angle brackets (&lt;&gt;). For example, &lt;file name&gt; and &lt;project name&gt;.pdf file.</td>
</tr>
<tr>
<td><strong>Initial Capital Letters</strong></td>
<td>Indicate keyboard keys and menu names. For example, the Delete key and the Options menu.</td>
</tr>
<tr>
<td><strong>“Subheading Title”</strong></td>
<td>Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”</td>
</tr>
<tr>
<td><strong>Courier type</strong></td>
<td>Indicates signal, port, register, bit, block, and primitive names. For example, data1, tdi, and input. The suffix ( n ) denotes an active-low signal. For example, resetn. Indicates command line commands and anything that must be typed exactly as it appears. For example, c:\qdesigns\tutorial\chiptrip.gdf. Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword SUBDESIGN), and logic function names (for example, TRI).</td>
</tr>
<tr>
<td>➡️ ➡️ ➡️</td>
<td>An angled arrow instructs you to press the Enter key.</td>
</tr>
<tr>
<td>1., 2., 3., and a., b., c., and so on</td>
<td>Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.</td>
</tr>
<tr>
<td>■ ■ ■</td>
<td>Bullets indicate a list of items when the sequence of the items is not important.</td>
</tr>
<tr>
<td><img src="hand.png" alt="hand" /></td>
<td>The hand points to information that requires special attention.</td>
</tr>
<tr>
<td>![question mark](question mark.png)</td>
<td>A question mark directs you to a software help system with related information.</td>
</tr>
<tr>
<td><img src="feet.png" alt="feet" /></td>
<td>The feet direct you to another document or website with related information.</td>
</tr>
<tr>
<td><img src="caution.png" alt="caution" /></td>
<td>A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.</td>
</tr>
<tr>
<td><img src="warning.png" alt="warning" /></td>
<td>A warning calls attention to a condition or possible situation that can cause you injury.</td>
</tr>
<tr>
<td><img src="email.png" alt="email" /></td>
<td>The envelope links to the Email Subscription Management Center page of the Altera website, where you can sign up to receive update notifications for Altera documents.</td>
</tr>
</tbody>
</table>