Test DDR or DDR2 SDRAM Interfaces on Hardware

Using the Example Driver

June 2006 ver 1.2 Application Note 380

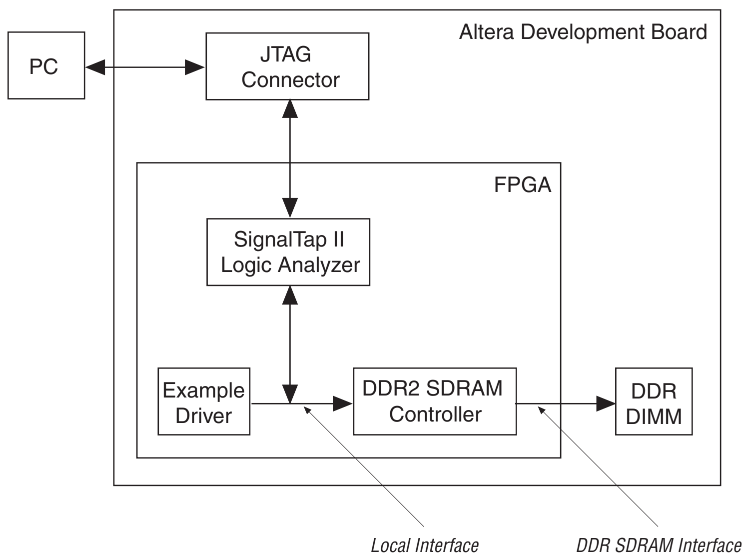
Introduction

This application note describes how to test DDR or DDR2 SDRAM interfaces on Altera® development boards using the Altera DDR or DDR2 SDRAM Controller MegaCore® function-generated example driver. The example driver—a stand-alone synthesizable circuit—demonstrates the DDR or DDR2 SDRAM interface. You can use these instructions to quickly build a DDR or DDR2 SDRAM interface on one of the Altera boards and see it working; or use the same principles to establish whether the DDR or DDR2 SDRAM interface on your board is working as expected, independently of any other circuit.

☞ This application note describes a DDR2 SDRAM Controller example driver, but is applicable to the Altera DDR SDRAM Controller.

Figure 1 shows the example system block diagram.

Figure 1. Example System Block Diagram



This application note details the following topics that help you build a stand-alone synthesizable circuit that demonstrates the DDR2 SDRAM interface:

Overview

A PC running the Quartus® II software downloads the device programming file and monitors the activity on the DDR2 SDRAM Controller local interface. The Quartus II SignalTap® II utility captures the activity on the DDR2 SDRAM Controller local interface via the JTAG connector.

The driver is a self-checking test generator for the DDR2 SDRAM controller. The driver uses a state machine to write data patterns to a range of column addresses, within a range of row addresses in all memory banks. The driver then reads back the data from the same locations, and checks that the data matches. The `pnf` (pass not fail) output transitions low if any read data fails the comparison. There is also a `pnf_per_byte` output, which shows the comparison on a per byte basis. The `test_complete` output transitions high for a clock cycle at the end of the write then read sequence. After this transition the test restarts from the beginning and repeats indefinitely.

For more information on `pnf_per_byte`, refer to "Appendix A: Interpret the pnf_per_byte Output" on page 24.

The data patterns are generated with an 8-bit linear feedback shift register (LFSR) per byte—each LFSR has a different initialization seed.

The application note requires the following hardware and software:

- Cyclone™ II PCI Development Board, available in the PCI High-Speed Development Kit, Cyclone II Edition
- DDR2 SDRAM Controller MegaCore function
- Quartus II software

> The principles in this application note are the same for any Altera development board.

Set Up the Quartus II Project

To set up the Quartus II project, follow these steps:

1. Follow the instructions in the *PCI High-Speed Development Kit, Cyclone II Edition, Getting Started User Guide* to correctly install your Cyclone II PCI Development Board.

2. Start the Quartus II software and create a new project by choosing **New Project Wizard** (File menu).

3. On page 3 of 5 of the New Project Wizard in the **Family** drop-down box choose **Cyclone II**. In the **Available Devices** list choose **EP2C35F672C6**.

4. Click **Finish**.

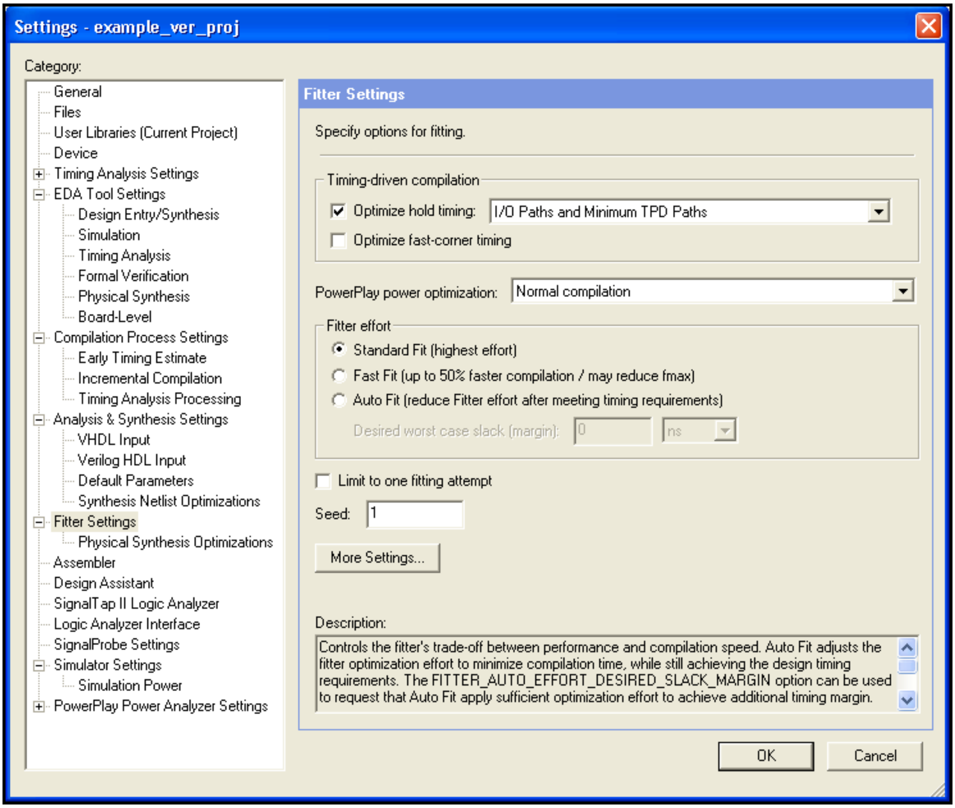
For more information, see the *DDR & DDR2 SDRAM Controller Compiler User Guide*.

Fitter Effort

You must ensure the Quartus II Fitter Effort is set to standard, for the best timing placements—timing placements are essential for a DDR2 SDRAM interface.

✓ To set the fitter effort, choose **Settings** > **Fitter Settings** > **Fitter effort** = **Standard Fit (highest effort)** (Assignments menu), see Figure 2.

Figure 2. Fitter Settings

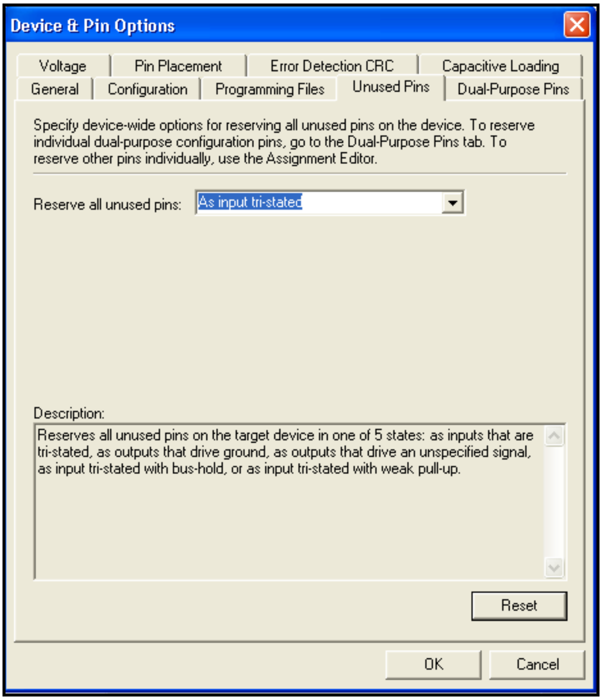


Unused Pins

You must ensure other unused pins on the device are tri-stated inputs (because the unused pins still attach to various devices on the development board), by following these steps:

1. Choose **Assignments** > **Settings** > **Device** > **Device & Pin Options** > **Unused Pins**, and for **Reserve all unused pins** select **As inputs, tri-stated** (see Figure 3). Click **OK**, and click **OK**.

Figure 3. Device & Pin Options



Generate a DDR2 SDRAM Controller MegaCore Function

To generate a DDR2 SDRAM Controller MegaCore Function, follow these steps:

1. Choose **MegaWizard Plug-in Manager** (Tools menu), select **Create a new custom megafunction variation** and click **Next**.

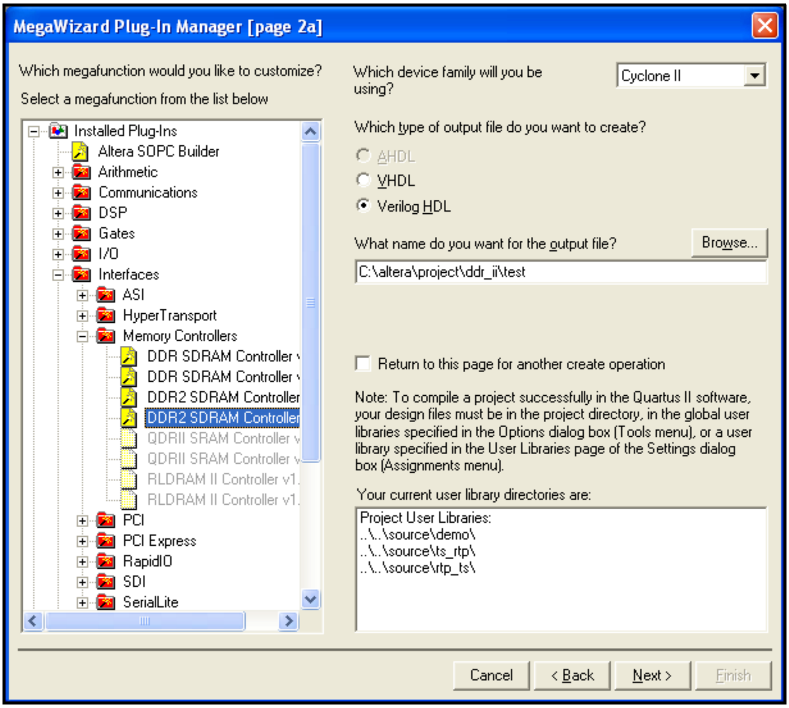
2. In the Device drop-down box choose **Cyclone II**. For the output file type select **VHDL** or **Verilog HDL**, and enter a name, for example, **test**.

☞ The <*variation name*> must be a different name from the project name and the top-level design entity name.

3. Choose **DDR2 SDRAM Controller** <*version*> in the **Interfaces > Memory Controllers** directory.

4. Click **Next** (see Figure 4).

Figure 4. Select the Megafunction



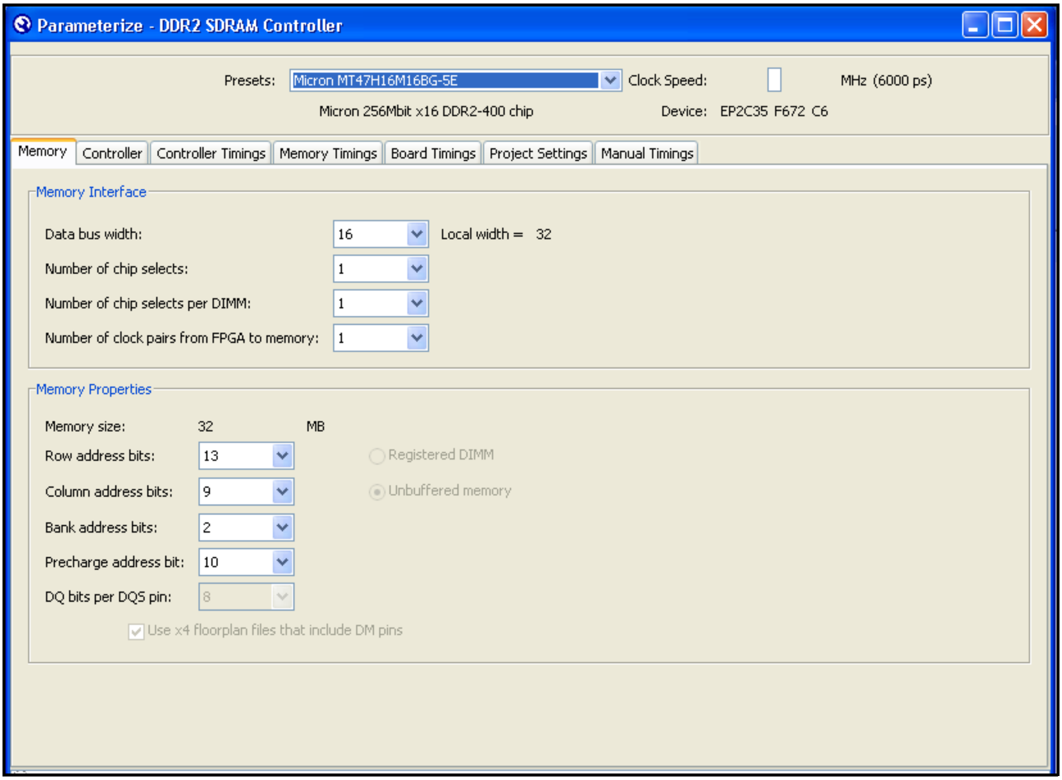
Parameterize the DDR2 SDRAM Controller

To parameterize the DDR2 SDRAM Controller, follow these steps:

1. Click **Step 1: Parameterize**.

2. In the **Presets** list, choose **Micron MT47H16M16BG-5E** (see Figure 5), which selects the correct settings on each tab for this device.

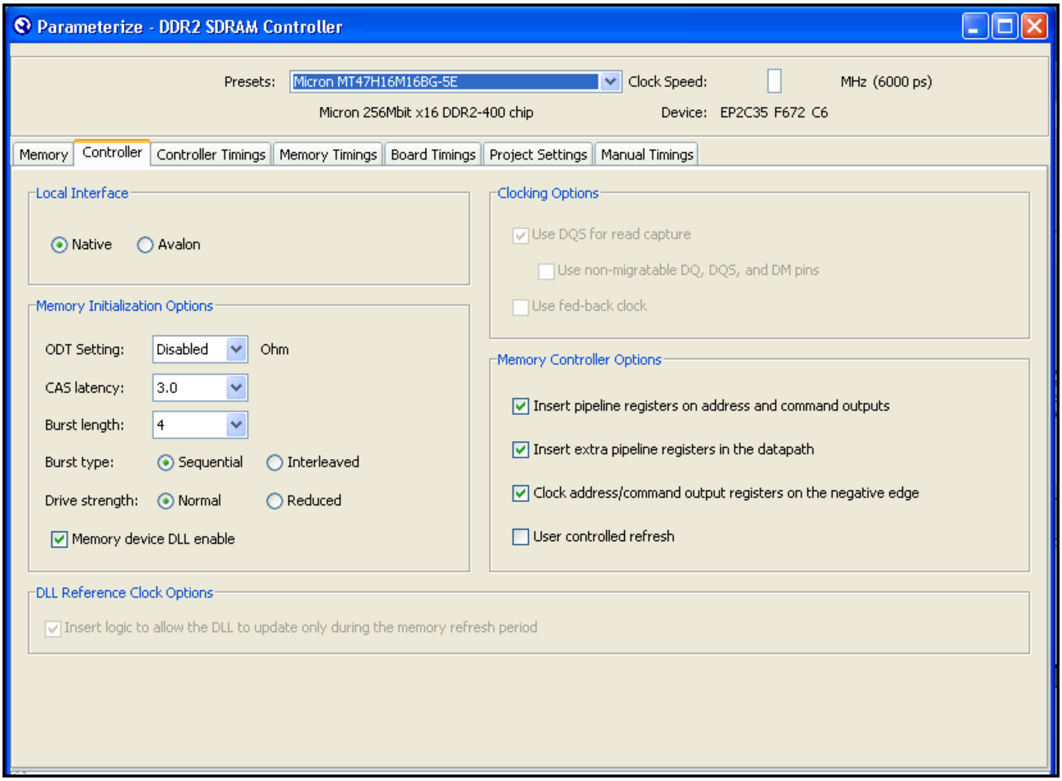
Figure 5. Choose Memory Device



3. Click the **Controller Settings** tab.

4. Turn on **Insert extra pipeline registers on address and command outputs** (see Figure 6), which inserts an extra pipeline stage between the DDR2 SDRAM Controller and the input-output element (IOE) register to improve f_{MAX}. Do not change any other settings.

Figure 6. Controller Settings

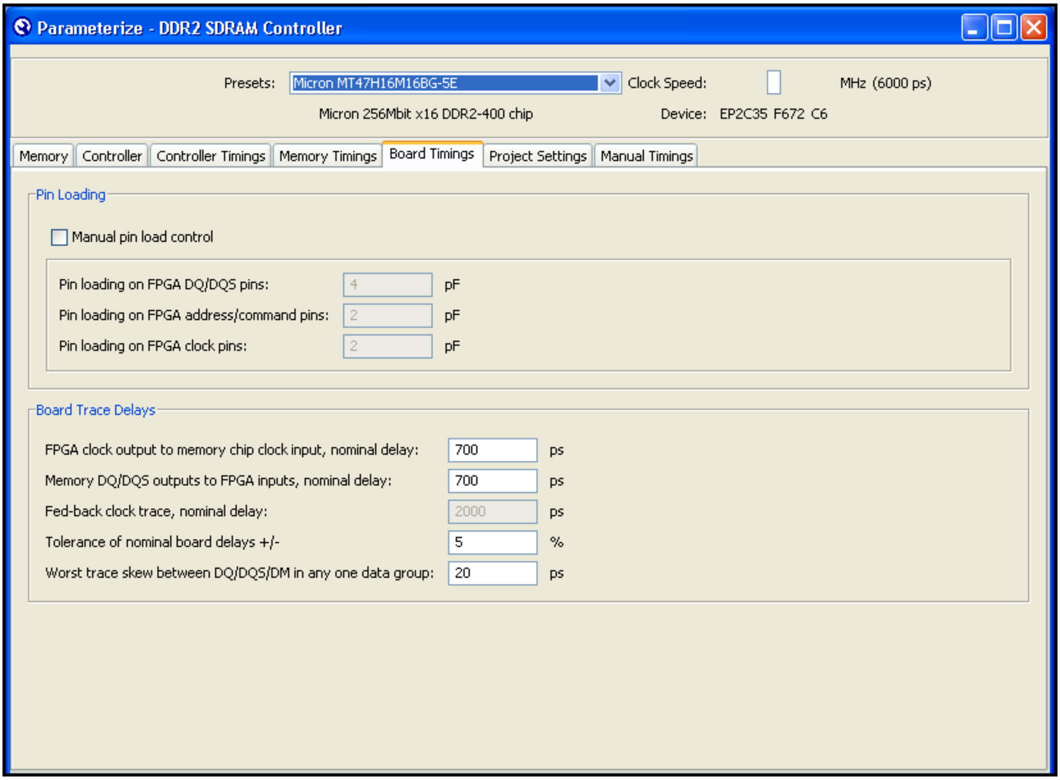


5. Click the **Board Timings** tab.

6. For the **FPGA Clock output to memory chip clock input, nominal delay**, enter **700ps**; for the **Memory DQ/DQS outputs to FPGA inputs, nominal delay**, enter **700ps** (see Figure 7 on page 9).

These settings are for the Cyclone II PCI Development Board. For other Altera board settings, see "Appendix B. Useful Development Board Information" on page 26.

Figure 7. Board Timings



7. Click **Finish** on the Parameterize - DDR SDRAM Controller window.

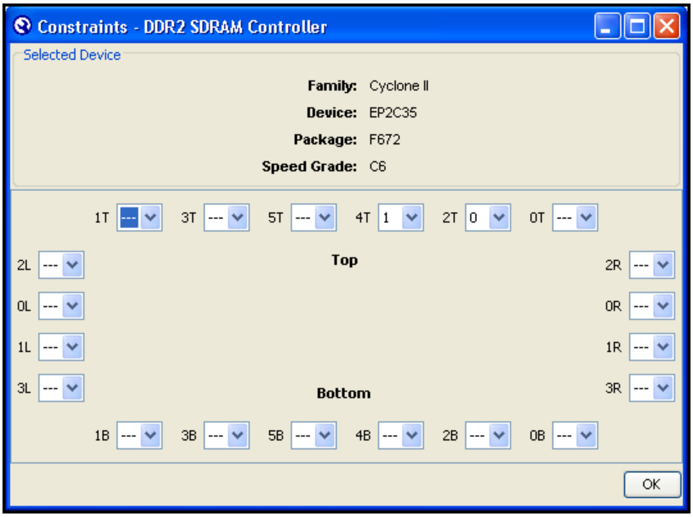
Choose DQS Group Placement for the DDR2 SDRAM Controller

To choose DQS group placement for the DDR2 SDRAM Controller, follow these steps:

1. Click **Step 2: Constraints**.

2. Set 2T = 0 and 4T = 1 (settings for the Cyclone II PCI Development board). See Figure 8.

Figure 8. Constraints



Assignments made here must match your board layout—which is design dependant. The IP Toolbench-generated constraints set up the pin assignments, LogicLock™ regions, IO standards, and other constraints. Therefore, these groupings must match the pin out on the board.

For the settings for other Altera boards, run the appropriate reference board's constraints Tcl file, which is in the **\lib** directory of the MegaCore function. See "Select the Board Pin Outs" on page 16.

3. Click **OK** on the Constraints - DDR2 SDRAM Controller window.

Set Up Simulation

To set up simulation, follow these steps:

This application note explains the steps to simulate a Verilog HDL design. Follow the same steps to simulate a VHDL design.

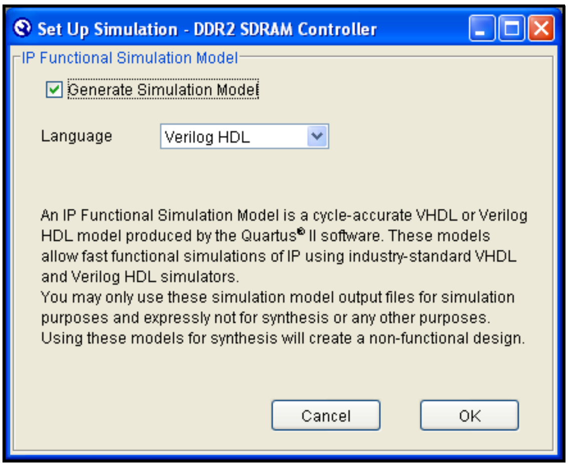
To generate an IP functional simulation model for your MegaCore function, follow these steps:

1. Click **Step 3: Set Up Simulation** in IP Toolbench (see Figure 9).

Figure 9. IP Toolbench—Set Up Simulation

2. Turn on **Generate Simulation Model** (see Figure 10).

Figure 10. Generate Simulation Model



3. Choose the language in the **Language** list.

   To use the IP Toolbench-generated testbench, choose the same language that you chose for your variation.

4. Click **OK**.

For instructions and tips on functional simulation, see "Appendix C. Perform Functional Simulation" on page 27.

Generate the DDR2 SDRAM Controller

To generate the DDR2 SDRAM Controller, follow these steps:

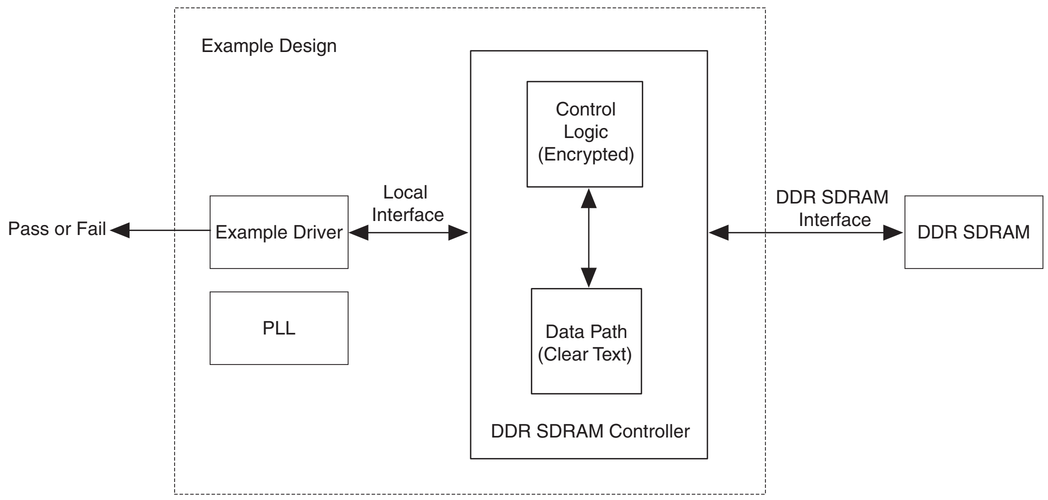
1. Click **Step 4: Generate** on the DDR2 SDRAM Controller window.

2. When the **MegaCore Function Generation Successful** message appears, click **Exit** on the Generation - DDR2 SDRAM Controller window.

Increase the Example Driver Address Range

Figure 11 shows a system-level diagram including the example instance that the DDR2 SDRAM Controller MegaCore function creates for you.

Figure 11. DDR SDRAM Controller System-Level Diagram



To test more of the memory, you can increase the example driver address range, if not go to "Edit the PLL" on page 14. To increase the address range of the example driver, follow these steps:

1. In the Quartus II software, choose **Open** (File menu) and choose <*variation name*>**_driver.vhd** or **.v**, in this example choose **test_example_driver.vhd** or **.v**.

2. Search for the following line in VHDL:

```
MAX_ROW <= std_logic_vector'("0000000000011");
```

   For Verilog HDL, search for the following line:

```
Assign MAX_ROW = 3;
```

3. Change the line to the following VHDL code:

```
MAX_ROW <= to_stdlogicvector("0000000000001" sll
(memory row bits -1));
```

   For Verilog HDL, change the line to the following code:

```
Assign MAX_ROW = 1<<(memory row bits - 1);
```

   ☞ Replace *memory row bits* with your value for the memory row bits on the Memory Settings tab.

4. Search for the following line in VHDL:

```
MAX_COL <= std_logic_vector'("0000010000");
```

   For Verilog HDL, search for the following line:

```
Assign MAX_COL = 16;
```

5. Change the line to the following VHDL code:

```
MAX_COL <= to_stdlogicvector("0000000001" sll
(memory column bits -1));
```

   For Verilog HDL, change the line to the following code:

```
Assign MAX_COL = 1<<(memory column bits - 1);
```

   ☞ Replace *memory column bits* with your value for the memory column bits on the Memory Settings tab.

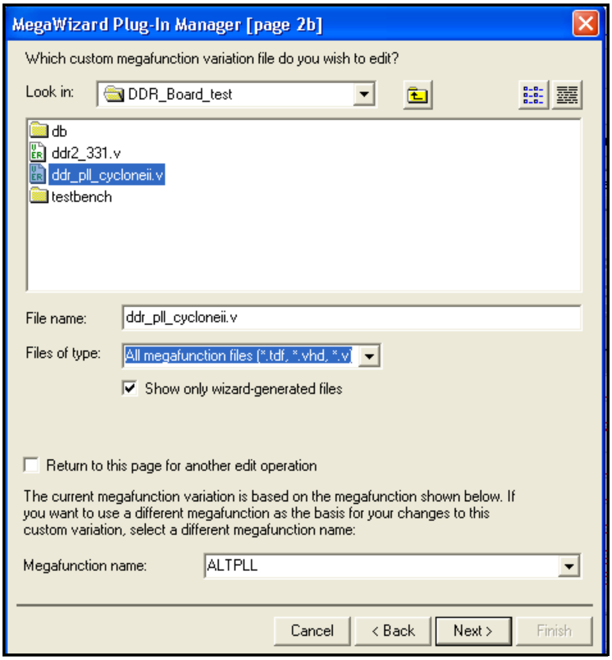
Edit the PLL

The IP Toolbench-generated PLL has an input to output clock ratio of 1:1 and a clock frequency that you entered in IP Toolbench. However, the Cyclone II PCI Development Board uses a 100-MHz input clock. To update the PLL for the design, follow these steps:

1. Choose **MegaWizard Plug-in Manager** (Tools menu), select **Edit an existing custom megafunction variation** and click **Next**.

2. Choose **ddr_pll_cycloneii.vhd** and click **Next** (see Figure 12).

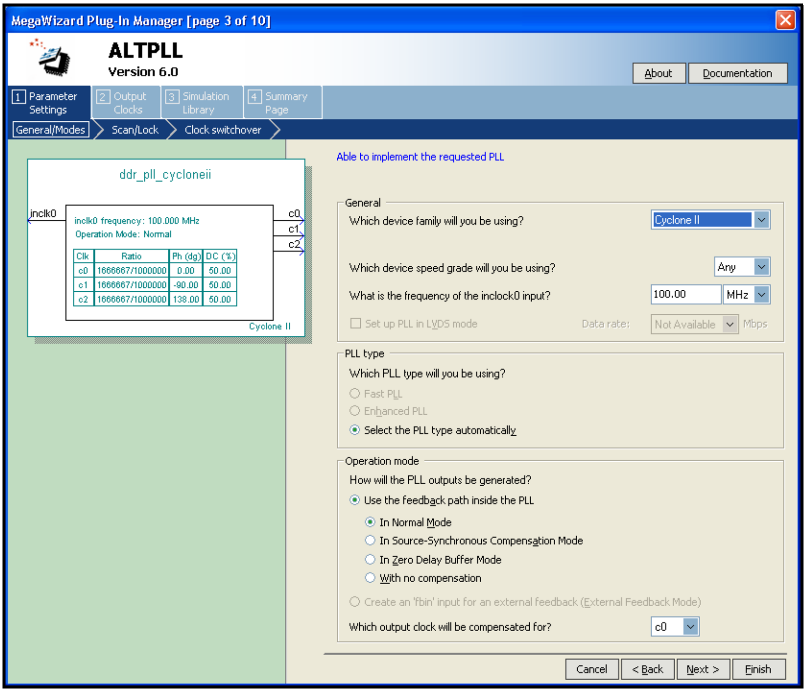
Figure 12. Choose ddr_pll_cycloneii



3. On ALTPLL [page 3 of 10], in **What is the frequency of the inclock0 input?** enter **100 MHz** (see Figure 13). Click **Next** four times.

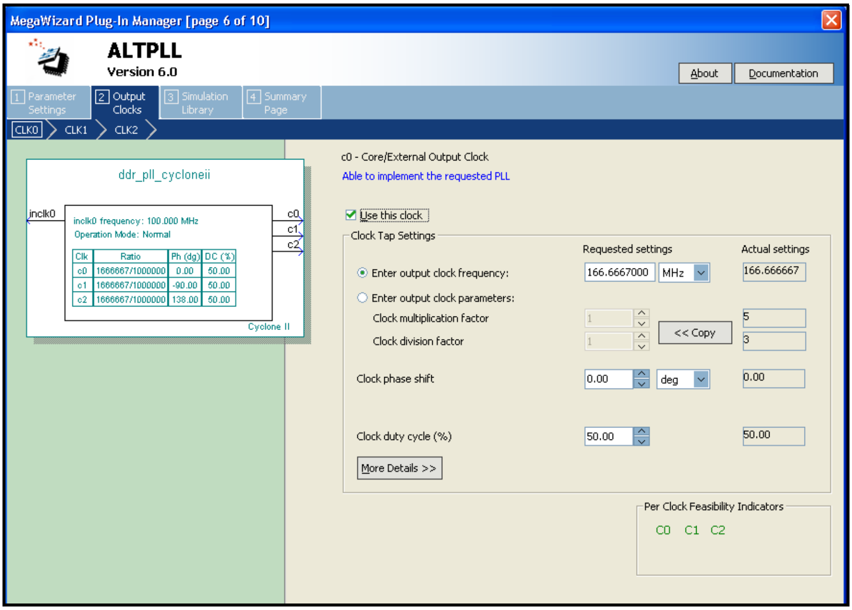
☞ This setting is design dependant, and may be different in your design.

Figure 13. Set the PLL Frequency



4. On ALTPLL [page 7 of 10], for C0 select **Enter output clock Frequency** and in Requested Settings enter **166.7MHz** (see Figure 14). This setting should match your memory clock speed. Ignore the **Cannot Implement the Requested PLL** error message and click **Next**.

Figure 14. Set Clock Speeds for Each PLL Output



5. Repeat for C1, C2. For C1 clock, set the **Clock phase shift** to –90 degrees; for the C2 clock, –135 degrees.

6. Click **Finish**.

☞ In IP Toolbench, if you want to regenerate your design, turn off **Automatically generate the PLL**, so IP Toolbench does not overwrite the changes that you made to the PLL.

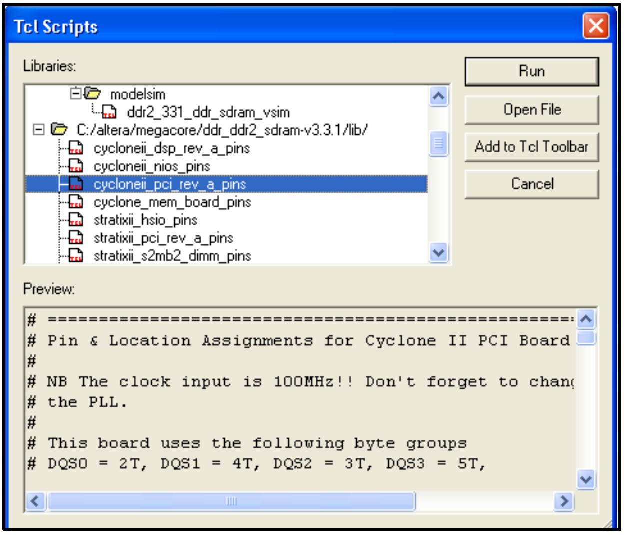
Select the Board Pin Outs

To select the appropriate pin out for the Cyclone II PCI Development Board, follow these steps:

1. Choose **Tcl Scripts** (Tools menu).

2. Choose **cycloneii_pci_rev_a_pins** in the **c:/MegaCore/ddr_ddr2_sdram-**<*version*>**/lib/** directory and click **Run** (see Figure 15).

Figure 15. Pin Out Tcl Script



☞ There is one file for each supported Altera memory development board. For your own board design, manually create one of these files using on of the files as a guide or use the Quartus II Assignment Editor to assign your pins.

When the script is complete, the following message displays:

```
Info: Successfully loaded and ran Tcl Script File
"C:\MegaCore\ddr_ddr2_sdram-
<version>\lib\cycloneii_pci_rev_a_pins.tcl"
```

Compile the Design

Before the Quartus II software compiles the design, it runs the IP Toolbench-generated Tcl constraints script, **auto_add_constraints.tcl**.

✓ Choose **Start Compilation** (Processing menu), which runs the add constraints scripts, compiles the design, and performs timing analysis.

For more information on the constraints script and timing analysis, see the *DDR & DDR2 SDRAM Controller Compiler User Guide*.

When the compilation is complete, the Quartus II processing messages tab displays the post-compilation timing analysis results. The results are also written to the <*variation name*>**_post_summary.txt** file in your project directory.

The results show how much slack you have for each of the various timing requirements—negative slack means that you are not meeting timing.

If the verify timing script reports that your design meets timing, you have successfully generated and implemented your DDR SDRAM Controller.

☞ The verify timing script checks the round trip delay, but it does not check that your FPGA can run at this frequency. You should check the f_{MAX} of your system using the Quartus II timing analysis to ensure that your internal logic runs at the desired speed.

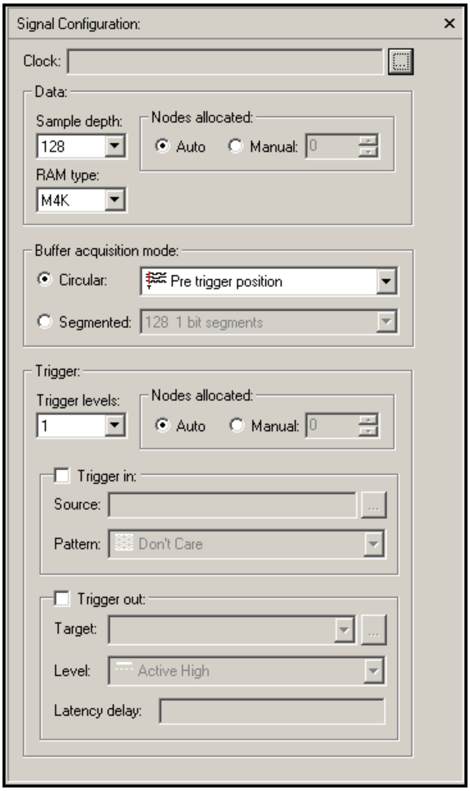
Set Up the SignalTap II Logic Analyzer

To set up your SignalTap II settings to observe your design working on your board, follow these steps:

1. Choose **SignalTap II Logic Analyzer** (Tools menu).

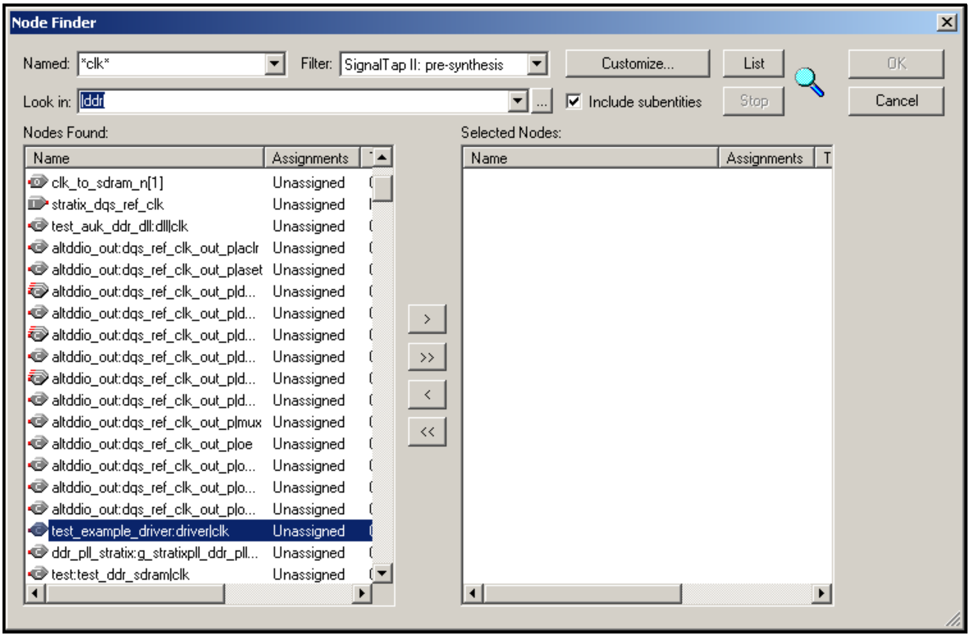
2. In the Signal Configuration window, click the **...** button (see Figure 16).

Figure 16. Signal Configuration Window



3. In the **Named** box enter `*clk*` and click **List** (see Figure 17).

Figure 17. Add Clock



4. Choose `test_example_driver:driver|clk` in the **Nodes Found** list and click > to add to the **Selected Nodes** list.

5. Click **OK**.

6. In the Signal Configuration window, choose the following settings:

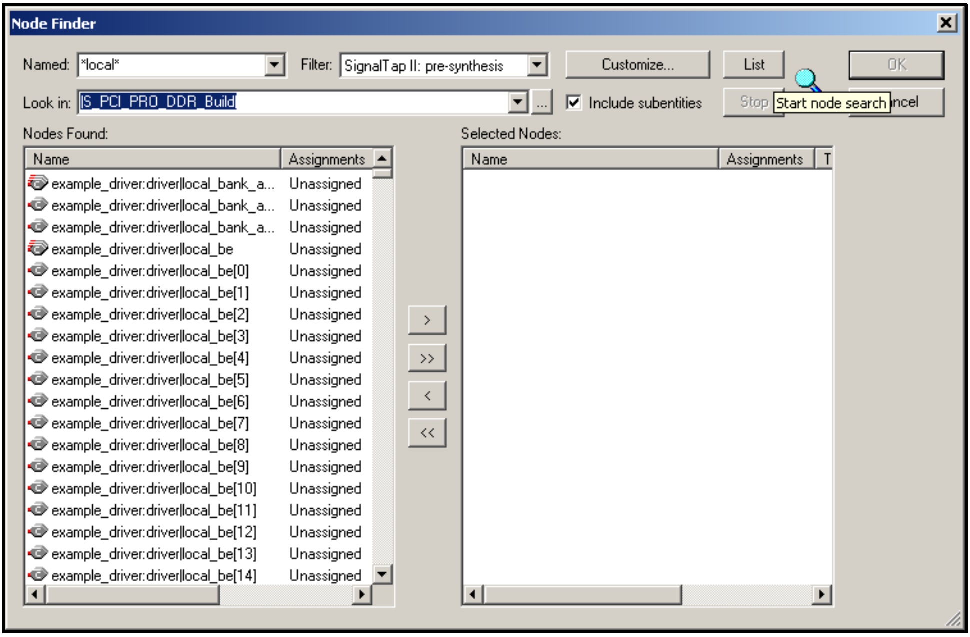
   - In the **Sample depth** box choose **512**
   - In the **RAM type** box choose **M-RAM**
   - In **Buffer acquisition mode** select **Circular: Center trigger position**

7. Choose **Add nodes** (Edit menu).

☞ Do not add any DDR SDRAM interface signals (DQ or DQS), because the additional logic, which the SignalTap II logic analyzer adds, adversely affects your timings.

8. In the **Named** box enter `*local*` and click **List** (see Figure 18).

Figure 18. Add SignalTap II Nodes



9. Choose the following signals in the **Nodes Found** list and click **>** to add to the **Selected Nodes** list:

- `example_driver:driver|local_rdata`
- `example_driver:driver|local_rdata_valid`
- `example_driver:driver|local_read_req`
- `example_driver:driver|local_wdata`
- `example_driver:driver|local_wdata_req`
- `example_driver:driver|local_write_req`

10. In the **Named** box enter `*pnf*` and click **List**.

11. Choose the following signals in the **Nodes Found** list and click **>** to add to the **Selected Nodes** list:

- `pnf`
- `pnf_per_byte`

12. In the **Named** box enter `*test_complete*` and click **List**.

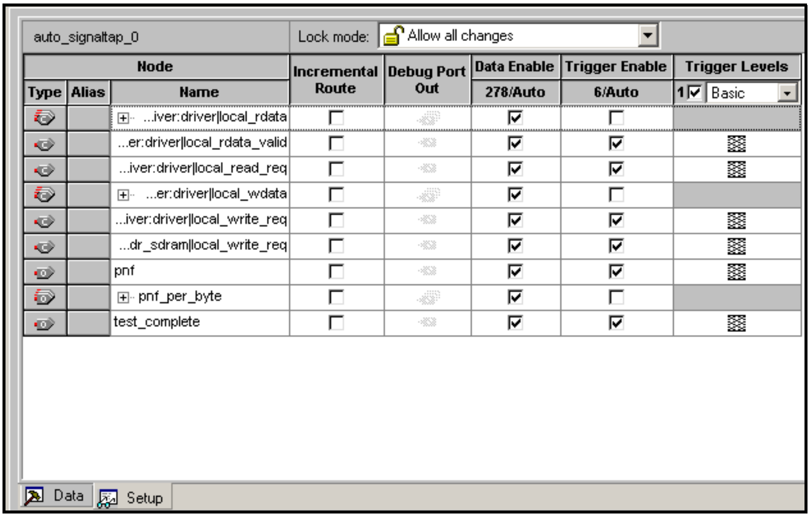
13. Choose the `test_complete` signal in the **Nodes Found** list and click **>** to add to the **Selected Nodes** list.

14. Click **OK** in the Node Finder window.

15. To reduce the SignalTap logic size, turn off **Trigger Enable** on the following signals (see Figure 19):

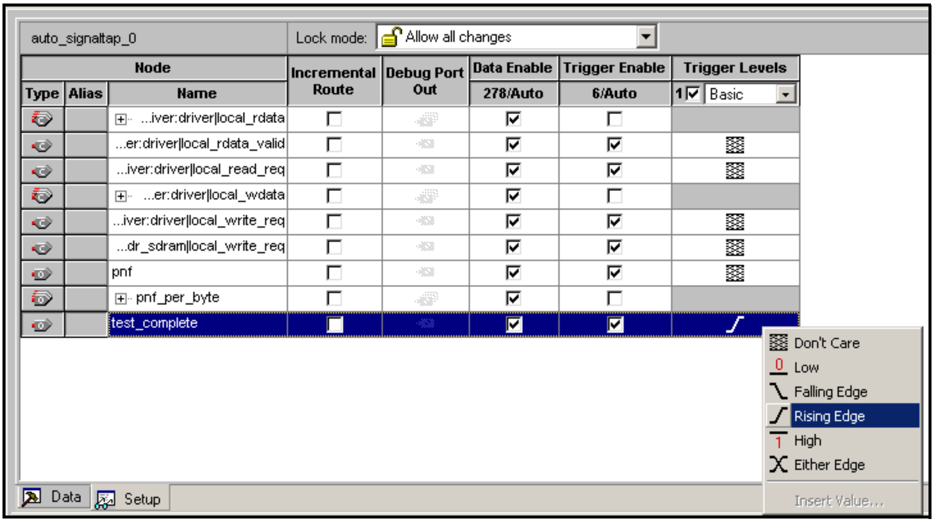
   - `example_driver:driver|local_rdata`
   - `example_driver:driver|local_wdata`
   - `pnf_per_byte`

Figure 19. Trigger Enable



16. Right click on the `test_complete` Trigger Levels cell and set to trigger on a **Rising Edge** (see Figure 20).

Figure 20. Rising Edge



17. Choose **Save** (File menu), and choose **Yes** to the prompt **Do you want to enable SignalTap II File stp1.stp for the current project?**

18. Re-compile the design to add the SignalTap II probes, by choosing **Start Compilation** (Processing menu).

19. When compilation is complete, connect your download cable (for example, ByteBlaster™ II download cable) to the JTAG port on the development board.

20. In the SignalTap II logic analyzer in the JTAG Chain Configuration window:

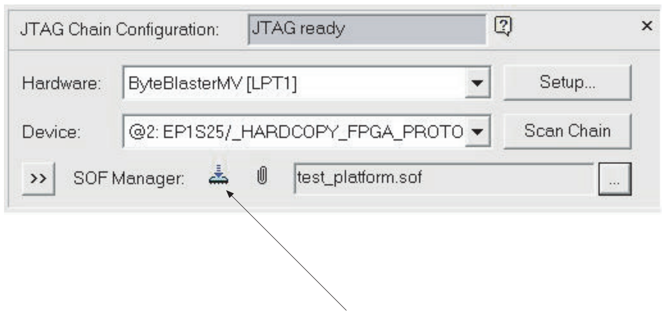
    - In the Hardware list, choose **ByteBlasterII [LPT1]**
    - In the Device list, choose **EP2C35**
    - In the SOF Manager list, choose <*project name*>**.sof**

Program the Device

To program the device, follow these steps:

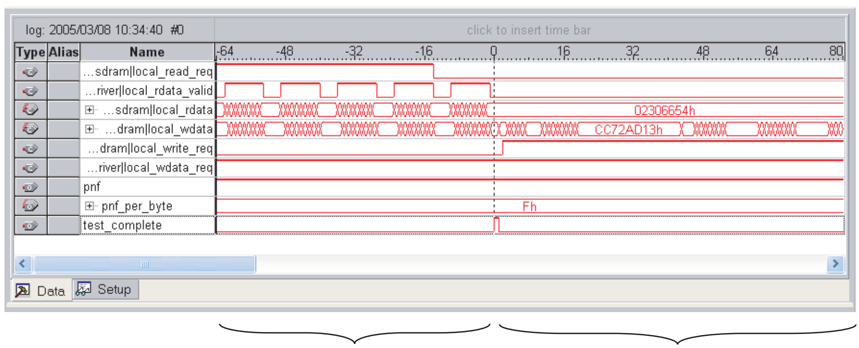
1. Click the **Program Device** icon that is next to SOF Manager (see Figure 21).

Figure 21. Program Device



2. Click **Run Analysis** to run once; click **Autorun Analysis** to run continuously. See Figure 22.

Figure 22. Analysis



Appendix A: Interpret the pnf_per_byte Output

Figure 23 shows an example of how to interpret the `pnf_per_byte` output. This example uses a 24-bit wide data bus—three DQS pins and six `pnf_per_byte` signals. The numbers on the rising and falling edges of the DQS signals represent the `pnf_per_byte[5:0]` bus. For example, if `pnf_per_byte[3]` is zero and all other `pnf_per_byte` outputs are high, there is an error on the data clocked by the `DQS[0]` falling edge.

Figure 23. Interpret the pnf_per_byte Output

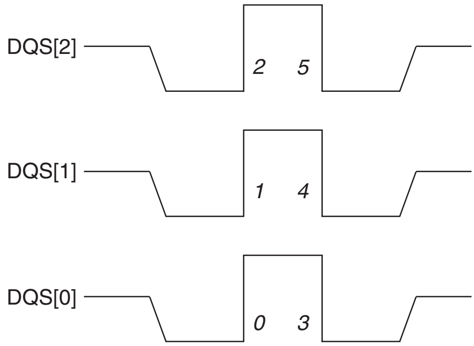


Figure 24 shows example `local_rdata`, `local_wdata`, and `pnf_per_byte` signals for this example.

> The `pnf_per_byte` output is three cycles after `local_rdata`.

Figure 24. Example Signals

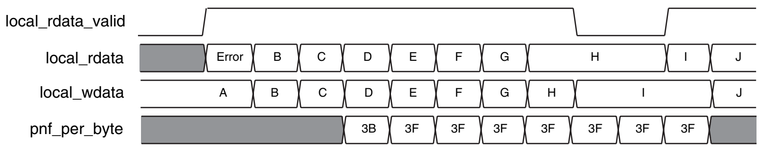


Table 1 shows generally how to interpret any errors on the `pnf_per_byte` signal.

Table 1. Interpret pnf_per_byte Errors

Error in Position	Interpretation
A	If the postamble logic is too late, you can miss capturing A, by not enabling on time.
H	If the postamble logic is too early, you can miss capturing F; by disabling too soon, you see E twice.

Appendix B. Useful Development Board Information

Table 2 shows a summary of the board level design information for various Altera IP evaluation boards.

Table 2. Altera Evaluation Boards *Note (1)*

Board	SDRAM	f_{MAX} (MHz)	FPGA-to-Memory Delay (ps)	Memory-to-FPGA Delay (ps)
Stratix Memory Demonstration Board 1	DDR	200	1,400	1,400
Stratix PCI Development Board	DDR	200	1,200	1,200
Stratix PCI High-Speed Development Board	DDR	167	1,200	1,200
Stratix GX Video Development Board	DDR	–	1,000	1,000
Stratix II Memory Demonstration Board 1	DDR	200	1,400	1,400
Stratix II Memory Demonstration Board 2	DDR2	267	1,400	1,400
Nios II Development Board, Cyclone II Edition	DDR	167	550	550
Cyclone™ DDR Memory Board	DDR	133	500	500
Twister DDR-SDRAM Evaluation Kit	DDR	133	500	500
Cyclone II DSP Development Board	DDR2	167	1,400	1,400
Cyclone II PCI Development Board	DDR2	167	700	700

Note to Table 2:

(1) The figures are for operation at room temperature and have not been verified over the full process, voltage, temperature (PVT) range.

Appendix C. Perform Functional Simulation

The DDR2 SDRAM Controller generates all the controller files with the testbench. The testbench files are located in the *<project name>***\testbench** directory.

The Verilog HDL testbench (*<variation name>***_sim_tb.v**) requires the modifications to match to your particular memory model. To simulate the design, follow these steps:

1. Download the simulation model of the memory type that you selected in the **DDR2 SDRAM Controller - Parametrize** window into the *<project name>***\testbench** directory.

2. Copy any parameter file (if separate from the model) in to the *<project name>***\testbench\modelsim** directory.

3. Open *<variation name>***_sim_tb.v** in a text editor.

4. Locate the line `generic_ddr_sdram_rtl memory_0_0`.

5. Replace `generic_ddr_sdram_rtl` with the *<modelname>*.

6. Ensure all signal names match those used in your model.

7. Repeat for all memory instances in the testbench.

The automatic testbench generation assumes each memory model is a ×8 device—has one DQS per DQ group and each chip is a single ×8 device.

If you have a ×16 device, follow these steps to ensure the testbench DQ, DQS, and DM signals match the model, otherwise go to “Increase the Example Driver Address Range” on page 12.

1. Check the `if def` statements within the parameters file and add the appropriate `define` statements to the top of the memory model. For example, for Micron models set the speed grade and number of DQ to DQS pins.

2. Start the ModelSim simulator and change directory to the **\testbench\modelsim**.

3. Type the following commands

```
set memory_model <model name>
source <variation_name>_ddr_sdram_vsim.tcl
```

For example to use two Micron MT47H16M16BG -5E ×16 DDR devices to make a 32-bit DDR SDRAM interface, follow these steps:

1. In the DDR SDRAM IP Toolbench, select **Micron MT47H16M16BG -5E** in the **Presets:** list.

   If the memory model is not available, add the model to the **memory_types.dat** file in the <*DDR installation directory*>**\ddr_ddr2_sdram-v#\constraints** directory.

2. Click the **Memory** tab and select **32** for the **Data bus width**.

3. Click **Finish**.

4. In IP Toolbench, click **Set Up Simulation** and turn on generation for a Verilog HDL simulation model.

5. Click **Finish**.

6. In IP Toolbench, click **Generate** to generate the custom MegaCore variation.

7. Download the **MT47H16M16BG -5E** Verilog HDL model from the Micron website.

8. Extract the **ddr.v** model to the <*project name* >**\testbench** directory.

9. Rename the **ddr_parameters.vh** file to **ddr_parameters.v** and move it to the <*project name*>**\testbench\modelsim** directory.

10. Open <*variation name*>**_sim_tb.v** in a text editor.

11. The IP Toolbench-generated testbench creates 4 × 8 devices. However, the DDR memory model is a ×16, so you require a 2 × 16 memory model instantiation. Locate the following line:

```
generic_ddr_sdram_rtl memory_0_0 (
```

12. Replace `generic_ddr_sdram_rtl` with the model name, for example `ddr eg`.

```
ddr memory_0_0 (
```

13. Repeat for `memory_0_2`.

14. Delete the instance `memory_0_1` and `memory_0_3` as these are not required.

15. Change the width of DQ, DQS, and DM to correctly match the model. For example change the following code:

```
//generic_ddr_sdram_rtl memory_0_0(
//        .Dq    (mem_dq[8* (0+1) - 1 : 8 * 0]),
//        .Dqs   (mem_dqs[0]),
//        .Addr  (a_delayed[11: 0]),
//        .Ba    (ba_delayed),
//        .Clk   (clk_to_ram),
//        .Clk_n (clk_to_ram_n),
//        .Cke   (cke_delayed[0]),
//        .Cs_n  (cs_n_delayed[0]),
//        .Ras_n (ras_n_delayed),
//        .Cas_n (cas_n_delayed),
//        .We_n  (we_n_delayed),
//        .Dm    (dm_delayed[0])
//    );
```

To the following code:

```
ddr2 memory_0_0 (
     .Dq    (mem_dq[15:0]),//Updated
     .Dqs   (mem_dqs[1:0]), //Updated
     .Addr  (a_delayed[11: 0]),
     .Ba    (ba_delayed),
     .Clk   (clk_to_ram),
     .Clk_n (clk_to_ram_n),
     .Cke   (cke_delayed[0]),
     .Cs_n  (cs_n_delayed[0]),
     .Ras_n (ras_n_delayed),
     .Cas_n (cas_n_delayed),
     .We_n  (we_n_delayed),
     .Dm    (dm_delayed[1:0])  //Updated
 );
```

and:

```
ddr2 memory_0_2 (
   .Dq    (mem_dq[31:16]),//Updated
     .Dqs   (mem_dqs[3:2]),//Updated
     .Addr  (a_delayed[11: 0]),
     .Ba    (ba_delayed),
     .Clk   (clk_to_ram),
     .Clk_n (clk_to_ram_n),
     .Cke   (cke_delayed[0]),
     .Cs_n  (cs_n_delayed[0]),
     .Ras_n (ras_n_delayed),
     .Cas_n (cas_n_delayed),
     .We_n  (we_n_delayed),
     .Dm    (dm_delayed[3:2])//Updated
 );
```

16. Save the testbench.

17. Open the DDR model and set the following `define` statements:

```
`define sg5E
`define x16
```

    These statements ensure the model is behaving as a ×16 device with the correct speed grade.

18. Start the ModelSim simulator and change directory to the *<project>***\testbench\modelsim** directory.

19. At the command prompt type the following command:

```
set memory_model ddr
```

20. On the Tools menu, click **Execute Macro** and select *<variation name>***_ddr_sdram_vsim.tcl**.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
literature@altera.com