Introduction

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

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

Figure 1 shows the block diagram.

Figure 1. Demonstration Overview
A PC running the Quartus® II software downloads the device programming file and monitors the activity on the DDR SDRAM Controller local interface. The Quartus II SignalTap® II utility captures the activity on the DDR SDRAM Controller local interface via the JTAG connector.

The driver is a self-checking test generator for the DDR 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 23.

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:

- DDR SDRAM Controller MegaCore function
- Quartus II software

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

This application note contains the following steps that help you build a stand-alone synthesizable circuit that demonstrates the DDR SDRAM interface:

- “Set Up the Project” on page 3
- “Launch the DDR SDRAM Controller MegaCore Function” on page 5
- “Increase the Example Driver Address Range” on page 11
- “Update Your PLL for the Design” on page 13
- “Compile the Design” on page 16
- “Select the Board Pin Outs” on page 15
- “Set Up the SignalTap II Logic Analyzer” on page 17
- “Program the Device” on page 22
Set Up the Project

To set up the project, follow these steps:

1. Follow the instructions in the *PCI High-Speed Development Kit, Stratix Professional Edition, Getting Started User Guide* to correctly install your Stratix PCI High-Speed 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 **Stratix**. In the **Available Devices** list choose one of the following devices:
   - For the PCI High-Speed Development Kit, Stratix Professional Edition, choose **EP1S60F1020C5**
   - For the PCI High-Speed Development Kit, Stratix Standard Edition, choose **EP1S25F1020C5**

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 DDR 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**

![Fitter Settings](image)

---

**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**

To launch the DDR 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 Stratix. 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 DDR SDRAM Controller <version> in the Interfaces > Memory Controllers directory.
4. Click **Next** (see Figure 4).

**Figure 4. Select the Megafunctio**

![Figure 4](image_url)

Parameterize the DDR SDRAM Controller

To parameterize the DDR SDRAM Controller, follow these steps:

1. Click **Step 1: Parameterize**.
2. In the **Presets** list, choose **Micron MT8VDDT3264HG-335C2** (see Figure 5), which selects the correct settings on each tab for this device.

**Figure 5. Choose Memory Device**

![Parameterize - DDR SDRAM Controller](image)

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 DDR SDRAM Controller and the input-output element (IOE) register to improve fMAX. 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 1200ps; for the Memory DQ/DQS outputs to FPGA inputs, nominal delay, enter 1200ps (see Figure 7 on page 9).

These settings are for the Stratix PCI High-Speed Development Board. For other Altera board settings, see “Appendix B. Useful Development Board Information” on page 25.
7. Click **Finish** on the Parameterize - DDR SDRAM Controller window.

### Choose DQS Group Placement for the DDR SDRAM Controller

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

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

2. Set 0T = 0, 1T = 1, 2T = 2, 3T = 3, 5T = 4, 6T = 5, 7T = 6, 8T = 7 (settings for the Stratix PCI High-Speed Development board). 9T, 4T, and 0B to 9B should all be blank (see Figure 8).

   You may see a 9 in the 4T, which you can ignore.

---

### 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 15.

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

In many design environments, the next step is to simulate the design. This application note does not describe design simulation.

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

The next step is to generate the DDR SDRAM Controller, by following these steps:

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

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

Increase the Example Driver Address Range

Figure 9 shows a system-level diagram including the example instance that the DDR SDRAM Controller MegaCore function creates for you.

Figure 9. DDR SDRAM Controller System-Level Diagram

To test more of the memory, you can increase the example driver address range, if not go to “Update Your PLL for the Design” on page 13. 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("0000000000001");

   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("000000000001" 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.
Update Your PLL for the Design

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 Stratix PCI High-Speed 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 megafuction variation and click Next.

2. Choose ddr_pll_stratix.vhd and click Next (see Figure 10).

Figure 10. Choose ddr_pll_stratix
3. On ALTPLL [page 3 of 17], in **What is the frequency of the inclock0 input?** enter **100 MHz** (see **Figure 11**). Click **Next** four times.

   ![Figure 11. Set the PLL Frequency](image)

   This setting is design dependant, and may be different in your design.
4. On ALTPLL [page 7 of 17], for C0 select **Enter output clock Frequency** and in Requested Settings enter **166.7MHz** (see Figure 12). This setting should match your memory clock speed. Ignore the **Cannot Implement the Requested PLL** error message and click **Next**.

**Figure 12. Set Clock Speeds for Each PLL Output**

5. Repeat for C1, C2, and C3 with the same settings.

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 Stratix PCI High-Speed Development Board, follow these steps:

1. Choose **Tcl Scripts** (Tools menu).
2. Choose <code>stratix_pci_rev_b_pins</code> in the
   <code>c:/MegaCore/ddr_ddr2_sdram-<version>/lib/</code> directory and click Run (see Figure 13).

   **Figure 13. 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 one 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\stratix_pci_rev_b_pins.tcl"
```

**Compile the Design**

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

- 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. The Messages window shows various timing margins for your design (see Figure 14).

**Figure 14. Verify 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 fMAX of your system using the Quartus II timing analysis to ensure that your internal logic runs at the desired speed.

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 15).

Figure 15. Signal Configuration Window
3. In the **Named** box enter *clk* and click **List** (see Figure 16).

**Figure 16. 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 add any DDR SDRAM interface signals (DQ or DQS), because the additional logic, which the SignalTap II logic analyzer adds, adversely effects your timings.
8. In the Named box enter *local* and click List (see Figure 17).

**Figure 17. 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 18):
   - `example_driver:driver|local_rdata`
   - `example_driver:driver|local_wdata`
   - `pnf_per_byte`

**Figure 18. Trigger Enable**
16. Right click on the test_complete Trigger Levels cell and set to trigger on a Rising Edge (see Figure 19).

**Figure 19. 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 EP1S60 (or EP1S25)
   - 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 20).
Appendix A: Interpret the pnf_per_byte Output

Figure 20. Program Device

2. Click Run Analysis to run once; click Autorun Analysis to run continuously. See Figure 21.

Figure 21. Analysis

Appendix A: Interpret the pnf_per_byte Output

Figure 22 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 22. Interpret the pnf_per_byte Output**

The pnf_per_byte output is three cycles after local_rdata.

**Table 1. Interpret pnf_per_byte Errors**

<table>
<thead>
<tr>
<th>Error in Position</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>If the postamble logic is too late, you can miss capturing A, by not enabling on time.</td>
</tr>
<tr>
<td>H</td>
<td>If the postamble logic is too early, you can miss capturing F; by disabling too soon, you see E twice.</td>
</tr>
</tbody>
</table>
Appendix B. Useful Development Board Information

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

<table>
<thead>
<tr>
<th>Board</th>
<th>SDRAM</th>
<th>$f_{\text{MAX}}$ (MHz)</th>
<th>FPGA-to-Memory Delay (ps)</th>
<th>Memory-to-FPGA Delay (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stratix Memory Demonstration Board 1</td>
<td>DDR</td>
<td>200</td>
<td>1,400</td>
<td>1,400</td>
</tr>
<tr>
<td>Stratix PCI Development Board</td>
<td>DDR</td>
<td>200</td>
<td>1,200</td>
<td>1,200</td>
</tr>
<tr>
<td>Stratix PCI High-Speed Development Board</td>
<td>DDR</td>
<td>167</td>
<td>1,200</td>
<td>1,200</td>
</tr>
<tr>
<td>Stratix GX Video Development Board</td>
<td>DDR</td>
<td>–</td>
<td>1,000</td>
<td>1,000</td>
</tr>
<tr>
<td>Stratix II Memory Demonstration Board 1</td>
<td>DDR</td>
<td>200</td>
<td>1,400</td>
<td>1,400</td>
</tr>
<tr>
<td>Stratix II Memory Demonstration Board 2</td>
<td>DDR2</td>
<td>267</td>
<td>1,400</td>
<td>1,400</td>
</tr>
<tr>
<td>Nios II Development Board, Cyclone II Edition</td>
<td>DDR</td>
<td>167</td>
<td>550</td>
<td>550</td>
</tr>
<tr>
<td>Cyclone™ DDR Memory Board</td>
<td>DDR</td>
<td>133</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Twister DDR-SDRAM Evaluation Kit</td>
<td>DDR</td>
<td>133</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Cyclone II DSP Development Board</td>
<td>DDR2</td>
<td>167</td>
<td>1,400</td>
<td>1,400</td>
</tr>
</tbody>
</table>

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.