Introduction

Cyclone® III devices support interfacing to both DDR2 and DDR SDRAM devices and modules. Altera provides the easy-to-use ALTMEMPHY megafuction to implement robust auto-calibrating interfaces to DDR2 and DDR SDRAM devices. You can configure this megafuction to support either a full-rate or half-rate controller. Tables 1 and 2 display the maximum clock frequencies for DDR2 and DDR SDRAM interfaces supported by Cyclone III devices for half-rate and full-rate controllers.

For detailed information about instantiating the ALTMEMPHY megafuction with half-rate or full-rate controller, refer to the ALTMEMPHY Megafuction User Guide.

Table 1. Cyclone III Maximum Clock Rate Support for External Memory Interfaces with Half-Rate Controller

<table>
<thead>
<tr>
<th>Memory Standard</th>
<th>I/O Standard</th>
<th>Commercial</th>
<th>Industrial</th>
<th>Automotive</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2 SDRAM</td>
<td>SSTL-18</td>
<td>–6 Speed Grade (MHz)</td>
<td>200</td>
<td>167</td>
</tr>
<tr>
<td>DDR SDRAM</td>
<td>SSTL-2</td>
<td>–6 Speed Grade (MHz)</td>
<td>167</td>
<td>150</td>
</tr>
</tbody>
</table>

Notes to Table 1:
(1) Column I/Os refer to Top and Bottom I/Os. Row I/Os refer to Right and Left I/Os. Hybrid mode refer to combination of Column and Row I/Os.
(2) The values apply for interfaces with both modules and components.
(3) The supported operating frequencies listed here are memory interface maximums for the FPGA device family. Your design’s actual achievable performance is based on design- and system-specific factors, as well as static timing analysis of the completed design.
(4) Refer to Table 3 for the required DDR2 memory device speed grade for external memory interface to achieve performance as stated in Table 1.
Table 2. Cyclone III Maximum Clock Rate Support for External Memory Interfaces with Full-Rate Controller

<table>
<thead>
<tr>
<th>Memory Standard</th>
<th>I/O Standard</th>
<th>SSTL-18</th>
<th>Commercial</th>
<th>Industrial</th>
<th>Automotive</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDR2 SDRAM</td>
<td>SSTL-18</td>
<td>Class I</td>
<td>167</td>
<td>167</td>
<td>150</td>
</tr>
<tr>
<td>DDR SDRAM</td>
<td>SSTL-18</td>
<td>Class II</td>
<td>150</td>
<td>150</td>
<td>133</td>
</tr>
</tbody>
</table>

Notes to Table 2:
(1) Column I/Os refer to Top and Bottom I/Os. Row I/Os refer to Right and Left I/Os. Hybrid mode refer to combination of Column and Row I/Os.
(2) The values apply for interfaces with both modules and components.
(3) The supported operating frequencies listed here are memory interface maximums for the FPGA device family. Your design’s actual achievable performance is based on design- and system-specific factors, as well as static timing analysis of the completed design.
(4) Refer to Table 3 for the required DDR2 memory device speed grade for external memory interface to achieve performance as stated in Table 2.

Table 3 displays the required DDR2 memory device speed grade for external memory interface.

Table 3. Required DDR2 Memory Device Speed Grade for External Memory Interface

<table>
<thead>
<tr>
<th>Temperature &amp; Speed Grade</th>
<th>Interface Location</th>
<th>I/O Standard SSTL-18</th>
<th>Memory Speed Grade (MHz)</th>
<th>Interface f_{max} (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C6</td>
<td>Top/Bottom</td>
<td>Class I</td>
<td>267</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class II</td>
<td>333</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Left/Right</td>
<td>Class I/II</td>
<td>200</td>
<td>167</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Class I/II</td>
<td>200</td>
<td>150</td>
</tr>
<tr>
<td>C7</td>
<td>Top/Bottom</td>
<td>Class I/II</td>
<td>200</td>
<td>167</td>
</tr>
<tr>
<td></td>
<td>Left/Right</td>
<td>Class I/II</td>
<td>200</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Class I/II</td>
<td>200</td>
<td>133</td>
</tr>
</tbody>
</table>
This application note describes Altera’s recommended design flow to implement DDR2 SDRAM memory interface using Cyclone III devices. This application note also describes how to use the FPGA design flow to generate an example design featuring a DDR2 SDRAM memory interface that uses the data path provided with the Altera® ALTMEMPHY megafunction. You can also use the same design flow for DDR SDRAM interfaces.

### FPGA Design Flow

Altera recommends the design guidelines described in this section as best practices for successful memory interface implementation in Cyclone III devices. These guidelines are designed to provide a good out-of-the-box experience with external memory interface in Cyclone III devices. **Figure 1** illustrates the design flow required for Cyclone III memory interfaces. Each step is discussed in detail in the following sections.

<table>
<thead>
<tr>
<th>Temperature &amp; Speed Grade</th>
<th>Interface Location</th>
<th>I/O Standard SSTL-18</th>
<th>Memory Speed Grade (MHz)</th>
<th>Interface f(_{\text{max}}) (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C8</td>
<td>Top/Bottom (2)</td>
<td>Class I/II</td>
<td>267</td>
<td>167</td>
</tr>
<tr>
<td></td>
<td>Left/Right</td>
<td>Class I/II</td>
<td>200</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Class I/II</td>
<td>200</td>
<td>125</td>
</tr>
<tr>
<td>I7</td>
<td>Top/Bottom</td>
<td>Class I/II</td>
<td>267</td>
<td>167</td>
</tr>
<tr>
<td></td>
<td>Left/Right</td>
<td>Class I/II</td>
<td>200</td>
<td>150</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Class I/II</td>
<td>200</td>
<td>133</td>
</tr>
<tr>
<td>A7</td>
<td>Top/Bottom</td>
<td>Class I</td>
<td>267</td>
<td>167</td>
</tr>
<tr>
<td></td>
<td>Class II</td>
<td>333</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Left/Right</td>
<td>Class I/II</td>
<td>200</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td>Hybrid</td>
<td>Class I/II</td>
<td>200</td>
<td>125</td>
</tr>
</tbody>
</table>

**Notes to Table 3:**
(1) For DDR2 SDRAM write timing performance on Columns I/O for C8 and A7 devices, 97.5 degree phase offset is required.
(2) For Q240 C8 device 167 MHz performance on Top/Bottom interface, 333 MHz DDR2 memory device is required.
Figure 1. Design Flow for Implementing External Memory Interfaces in Cyclone III Devices

Note to Figure 1:
(1) Although these steps are optional, Altera recommends following these steps.
Step 1: Select Device

Prior to the start of designing any memory interface, determine the required bandwidth of the memory interface. Bandwidth can be expressed as:

\[
\text{Bandwidth} = \text{data width (bits)} \times \text{data transfer rate (1/sec)} \times \text{efficiency}
\]

The efficiency is the percentage of time the data bus is transferring data. It is dependent on the type of memory. For example, in memory interfaces where there are separate write and read ports, the efficiency would be 100% when there is an equal amount of reads and writes on these memory interfaces.

After calculating the bandwidth of the memory interface, determine which memory and FPGA to use.

For information about selecting the different memory types, refer to the Selecting the Right High-Speed Memory Technology for Your System white paper.

In addition, Altera’s FPGA devices support various data widths for different memory interfaces. The memory interface support between density and package combinations is different so you need to determine which FPGA device density and package combination is best suited for your application.

For information about selecting the different memory types, differences between the memory types, and information about the FPGA density and package support for the different memory types, refer to the External Memory Interfaces in Cyclone III Devices chapter in volume 1 of the Cyclone III Device Handbook.
Step 2: Instantiate PHY and Controller in a Quartus II Project

Begin your memory interface design by instantiating the (physical layer) PHY and controller modules. Figure 2 shows a system-level diagram including the top-level example design that the DDR2 SDRAM High Performance Controller MegaCore® function creates for you.

The example design is a fully functional design that can be simulated, synthesized and used in hardware. It contains the PHY, controller, and an example driver. The example driver is a self-test module that issues read and write commands to the controller and checks the read data to produce the pass/fail and test-complete signals.

Figure 2. System-Level Diagram of DDR2 SDRAM Interface

There are two ways you can instantiate the components needed for the external memory interface:

- Use Altera’s DDR2 or DDR SDRAM High-Performance Controller MegaCore function, which builds the controller, instantiates the ALTMEMPHY megafuntion to build the data path or PHY, and instantiates the phase-locked loop (PLL).
- Use Altera’s ALTMEMPHY megafuntion alone to build the PHY and instantiate the PLL. You can create your own controller to work with the ALTMEMPHY megafuntion.
Altera provides an easy and fast way for you to create your memory interface design through the DDR2 or DDR SDRAM High-Performance Controller MegaWizard® Plug-In Manager, which instantiates the ALTMEMPHY megafuction for the data path and the phase-locked loop (PLL). The MegaWizard also generates an example driver for you to test your design. You can also use the example driver as a reference and create your own driver based on the example driver.

When instantiating the PHY as a standalone logic for your memory interface, Altera recommends using the ALTMEMPHY megafuction. This megafuction features a license-free physical interface that you can use with the standard Altera controller or your own custom controller. The ALTMEMPHY megafuction has the auto-calibration feature that automates timing closure for memory interfaces and adjusts timing-over-process, voltage, and temperature (PVT) variations.

There are two ways you can instantiate the ALTMEMPHY megafuction, either through the ALTMEMPHY MegaWizard Plug-In Manager, or using Altera’s DDR2 or DDR SDRAM High-Performance Controller MegaWizard Plug-In Manager, which already includes the ALTMEMPHY megafuction. Even if you plan to use your own controller, Altera recommends that you first create a design using Altera’s high-performance controller and then replace the Altera controller with your own controller. This way, you get an example design that you can simulate and verify.

For detailed information about instantiating the ALTMEMPHY megafuction or the memory controller, refer to the ALTMEMPHY Megafuction User Guide or the DDR and DDR2 SDRAM High-Performance Controller User Guide.

**Step 3: Perform RTL/Functional Simulation (Optional)**

When you instantiate the ALTMEMPHY megafuction using Altera’s DDR2 or DDR SDRAM High-Performance Controller MegaCore, there is an option to generate a simulation model or testbench of the design in either Verilog HDL or VHDL. Even though this step is optional, Altera recommends performing this step.

This IP functional simulation model is a cycle-accurate HDL model file produced by the Quartus® II software. When you instantiate the memory interface using the DDR2 or DDR SDRAM High-Performance Controller, it generates an example design and a testbench, in addition to the ALTMEMPHY megafuction simulation model. The models work with Altera-supported VHDL and Verilog HDL simulators, such as ModelSim.
You can set the Quartus II software to do a simulation of the design using the generated testbench with third-party simulators. This is done through the NativeLink feature of the Quartus II software.

Use these simulation model output files for simulation purposes only and not for synthesis or any other purposes. Using these models for synthesis creates a non-functional design.

For detailed information about generating the simulation model in the Quartus II software and simulating the design, refer to the DDR and DDR2 SDRAM High-Performance Controller User Guide.

Step 4: Add Constraints

The next step in the design process is to add all timing, location, and physical constraints related to the external memory interface. This includes timing, pin locations, I/O standards, and pin loading assignments. The ALTMEMPHY megafuction only supports timing analysis using TimeQuest Timing Analyzer with Synopsys Design Constraints (SDC) assignments. These constraints are derived from the parameters you entered for the ALTMEMPHY megafuction or the High-Performance Controller, based on the DDR2 and DDR SDRAM data sheet and tolerances from the board layout. The ALTMEMPHY megafuction uses TimeQuest timing constraints along with the timing driven fitter to achieve good timing closure.

After instantiating the ALTMEMPHY megafuction, the ALTMEMPHY MegaWizard generates the following files that you need in order to properly constrain the design:

- `<variation_name>_phy_ddr_timing.sdc` to constrain timing.
- `<variation_name>_pin_assignments.tcl` to make I/O standard assignments.

These script files are based on the design name used when instantiating the ALTMEMPHY megafuction. If you plan to use your own top-level design, you must edit the scripts to match your custom top-level design.
Table 4 displays the memory interface pin connections between DDR2 and Cyclone III devices.

<table>
<thead>
<tr>
<th>Interface Pin Description</th>
<th>Pin on Memory Device</th>
<th>Pin on FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write strobes</td>
<td>DQS</td>
<td>DQS</td>
</tr>
<tr>
<td>Read and write data</td>
<td>DQ</td>
<td>DQ</td>
</tr>
<tr>
<td>Data mask</td>
<td>DM</td>
<td>DQ</td>
</tr>
<tr>
<td>Memory clocks</td>
<td>CK, CK#</td>
<td>Any Adjacent User I/O (1),(2)</td>
</tr>
<tr>
<td>Address</td>
<td>A</td>
<td>Any User I/O (2)</td>
</tr>
<tr>
<td>Control signals</td>
<td>CS#, RS#, CAS#, WE#</td>
<td>Any User I/O (2)</td>
</tr>
</tbody>
</table>

Notes to Table 4:
1. Altera recommends that you use differential I/O pair for CK/CK# implementation.
2. Altera recommends that you place these pins in the same I/O bank as the DQ/DQS pins.

For detailed information about creating, generating, and setting the constraints for the design, refer to the ALTMEMPHY Megafunction User Guide.

Step 5: Compile Design and Verify Timing Closure

After you have made the proper constraints to the design, compile the design in the Quartus II software. Upon completion of the compilation, you can generate the timing report by executing the Report DDR function from the Tasks pane of the TimeQuest Timing Analyzer window. Executing this Report DDR task automatically runs the <variation_name>_phy_report_timing.tcl timing margin report script that is generated during the instantiation of the ALTMEMPHY megafunction.

By executing the task, you can get the timing report for different paths, such as write data, read data, command/address, DQS vs CK, mimic, and core (entire interface) timing paths in the design.

For detailed information about the timing analysis and reporting using the ALTMEMPHY megafunction, refer to AN 438: Constraining and Analyzing Timing for External Memory Interfaces in Stratix III and Cyclone III Devices.
Step 6: Adjust Constraints

In the timing report of the design, you can see the worst-case setup and hold margin for the different paths of the design. If the setup and hold margin are unbalanced and you wish to achieve a balanced setup and hold margin, you can adjust the phase setting of the clocks that are used to clock these paths. For example, in the case of the write data margin, the write data is clocked by a write clock, which is –90° with respect to the system clock. The system clock is used to clock the write data strobe. If the report timing script indicates that using a –90° phase setting for the write clock results in more hold time than setup time, you can adjust the write clock to earlier than –90° with respect to the system clock so that there will be less hold margin. Similarly, adjust the write clock to later than –90° with respect to system clock if there is more setup margin.

For detailed information about the clocks used in the ALTMEMPHY megafunction, refer to the ALTMEMPHY Megafunction User Guide.

Step 7: Perform Gate-Level Timing Simulation (Optional)

This optional step allows you to use timing simulation to ensure that your system meets the proper timing requirements needed by each module of the design. Even though this step is optional, Altera recommends performing this step.

Step 8: Determine Board Design Constraints

Once you have closed the timing for the design, examine the board design to determine how different factors can have an effect on the signal integrity, thus affecting the overall timing margin seen at both the memory and the FPGA. For example, the termination scheme used, the drive strength setting on the FPGA, and the loading seen by the driver can directly affect the signal integrity. You need to understand the trade-offs between the different types of termination schemes and the effects of output drive strengths and loading so that you can swiftly navigate through the multiple combinations and choose the best possible settings for your designs.

For detailed information about understanding the different effects on signal integrity design, refer to AN 408: DDR2 Memory Interface Termination, Drive Strength and Loading Design Guidelines.

Step 9: Perform Board-Level Simulations

After determining the system requirements for the board design and finalizing the right board constraints, run board-level simulation to see if the settings are optimum. With many variables that can affect the signal
integrity of the memory interface, simulating the memory interface provides you with an initial indication of how well the memory interface will perform.

Perform simulations on the data, data strobe, command, and address signals. If the memory interface does not have good signal integrity, you can adjust settings such as drive strength setting, termination scheme, or termination values, to improve the signal integrity (note that changing some of these settings will affect your timing and you may have to go back to the timing closure process if these change). There are various electronic design automation (EDA) simulation tools available to perform board-level simulations.

The input/output buffer information specific (IBIS) models for both the FPGA devices and the DDR2 and DDR SDRAM memory do not support read and write operations together, so you must perform the write and read operations separately for the simulation.

For detailed information about understanding the different effects on signal integrity design, refer to AN 408: DDR2 Memory Interface Termination, Drive Strength and Loading Design Guidelines.

**Step 10: Verify FPGA Functionality**

You can obtain useful information about the memory interface performance with board-level verification using the FPGA prototype. While the focus here is to ensure FPGA functionality in your end system, you can take additional steps to examine margins using oscilloscopes to verify the predicted size of the data-valid window, and the setup and hold margins at the I/O interface.

You can also use Altera’s SignalTap® II Embedded Logic Analyzer to perform system-level verification to correlate the system against your design targets.

For detailed information about using SignalTap II, refer to the Design Debugging Using the SignalTap II Embedded Logic Analyzer chapter in volume 3 of the Quartus II Handbook.
Design Guidelines for Implementing DDR and DDR2 SDRAM Interfaces in Cyclone III Devices

Example
Walkthrough for
167-MHz DDR2 SDRAM Interface

This section provides a walkthrough on how to use the FPGA-External Memory design flow described in the preceding sections to design a 32-bit wide 167-MHz DDR2 SDRAM memory interface targeted for the Cyclone III FPGA development kit. The example design also provides some recommended settings, including the termination scheme and drive strength in order to make your design flow easier. Although the example design is specifically for the DDR2 SDRAM memory interface, the design flow for a DDR SDRAM memory interface is the same.

This example design targets for 167 MHz because the development kit uses an EP3C120F780 device. This device is available in –7 and –8 speed grades only. You can achieve higher clock rate up to 200 MHz for DDR2 SDRAM if you select –6 speed grade devices from the Cyclone III family.

Step 1: Select the Device

Cyclone III devices support various data widths for DDR2 and DDR SDRAM memory interfaces. This walkthrough uses the Cyclone III EP3C120F780C7 device with 32-bit wide DDR2 SDRAM interface up to 167 MHz on the bottom I/O banks. The 32-bit wide interface uses four ×8 groups. For the DDR2 SDRAM memory device, choose Micron’s 512-MB MT47H32M16CC-3 333-MHz DDR2 SDRAM device that is used on the development kit.

Top/bottom DQ groups provide the fastest performance. The Quartus II software automatically uses the top/bottom DQ groups if they are available. Combining top/bottom DQ groups with left/right DQ groups for a single interface is not recommended and may result in a worse performance.

For more information about the DQ/DQS bus groups for different Cyclone III densities, packages, and sides of the device, refer to the External Memory Interfaces in Cyclone III Devices chapter in volume 1 of the Cyclone III Device Handbook.
Example Walkthrough for 167-MHz DDR2 SDRAM Interface

Step 2: Instantiate PHY and Controller in a Quartus II Project

Create a project in the Quartus II software targeting the EP3C120F780C7 device, as shown in Figure 3.

Figure 3. Creating Quartus II Project Targeting the EP3C120F780C7 Device

For detailed information about creating a Quartus II project, refer to the DDR and DDR2 SDRAM High-Performance Controller User Guide or the Quartus II Help.

After creating a Quartus II project, instantiate the DDR2 SDRAM controller. This example uses the DDR2 SDRAM High-Performance Controller, which instantiates the ALTMEMPHY megafuction automatically. Select the DDR2 SDRAM High-Performance Controller in the Interfaces section of the MegaWizard Plug-In Manager. Name the controller “DDR2”, as shown in Figure 4.

The subsequent files mentioned in this document that are generated by the MegaWizard and also other project files will have DDR2 as the prefix for the file names.
Design Guidelines for Implementing DDR and DDR2 SDRAM Interfaces in Cyclone III Devices

Figure 4. Invoking the DDR2 SDRAM High-Performance Controller using the MegaWizard Plug-In Manager
The rest of this subsection specifies the memory settings. Select the Cyclone III device with –7 speed grade. Then set the PLL reference clock frequency to 125 MHz and the memory clock frequency to 166.667 MHz. The 125 MHz PLL reference clock is provided by the on-board oscillator. Select the Micron MT47H32M16CC-3 333-MHz device. This 512-MB DDR2 device has a 16-bit data width. Figure 5 shows the memory settings panel.

The MegaWizard uses the default parameters for the memory when you instantiate the controller. The default parameters are based on the memory datasheets. You can customize your memory presets by modifying the parameters. Click the Modify parameters button to modify the memory attributes, memory initialization options, or the
memory timing parameters in the **Preset Editor** menu, as shown in Figure 6. In this design example, modify the memory interface DQ width to 32 to suit the targeted memory width on the development kit. This will customize your memory presets.

**Figure 6. Editing the Memory Presets to Create a Custom Memory**

![Image](image.png)

Figure 7 shows the **PHY Settings** tab. In the **PHY Settings** tab, you need to input the **Board Skew** under **Board Timing Parameters**. The specified skew is across all memory interface signal types including data, strobe, clock, address and command, and is used to generate the PHY timing constraints for all paths. The default value is set to 20 ps. You need to update this number based on your board specifications because this number is used to calculate the overall system timing margin.
Example Walkthrough for 167-MHz DDR2 SDRAM Interface

The DDR2 SDRAM device uses CK and CK# to clock the command and address signals into the memory. The controller names the CK and CK# signals as mem_clk and mem_clk_n, respectively. The skew between the CK or CK# and the DDR2 SDRAM-generated DQS signal is specified as tDQSC in the DDR2 SDRAM data sheet.

The DDR2 SDRAM has a write requirement (tDQSS) that states the positive edge of the DQS signal on writes must be within ± 25% (± 90°) of the positive edge of the DDR2 SDRAM clock input. tDQSS is defined as the time between the DQS latching edge to its associated clock edge. The controller generates the mem_clk and mem_clk_n signals using the DDR registers in the input/output element (IOE) to match with the DQS signal and reduce any variations across process, voltage, and temperature. The positive edge of the DDR2 SDRAM clock, mem_clk, is aligned with the DQS write to satisfy tDQSS.
The settings in the Auto-Calibration Simulation Options section are for RTL simulation only and are not applicable for gate-level simulation. Figure 8 shows the Controller Settings panel.
Choose your local interface setting in the Controller Settings panel. Turn on the Enable error detection and correction logic option if you want to use error code correction (ECC). If you have your own refresh requirements, then turn on Enable user-controlled refresh. Next, select the Local Interface Protocol for the memory interface. The default interface is the Avalon Memory-Mapped Interface that allows you to easily connect to other Avalon® Memory-Mapped peripherals.
Figure 9 shows the EDA panel. The MegaWizard can generate the simulation model for simulating the memory controller in either Verilog HDL or VHDL.

![Figure 9. DDR2 SDRAM High-Performance Controller EDA](image)

In the **Timing and Resource Estimation**, you can choose to generate a netlist if you are synthesizing your design with a third-party EDA synthesis tool.
Figure 10 shows the Summary panel.

For detailed step-by-step instructions about configuring the DDR2 SDRAM High-Performance Controller, refer to the DDR and DDR2 SDRAM High-Performance Controller User Guide.

Click Finish to generate the controller.
Step 3: Perform RTL/Functional Simulation (Optional)

After instantiating the DDR2 SDRAM High-Performance Controller, it generates an example design and driver for testing the memory interface. Figure 11 shows a system-level diagram of the example design that the DDR2 SDRAM High-Performance Controller MegaWizard creates for you.

Figure 11. DDR2 SDRAM High-Performance Controller System-Level Diagram

Note to Figure 11:
(1) The ALTMEMPHY megafunction automatically generates the PLL. The PLL is part of the ALTMEMPHY megafunction.

For more information about the different files generated by the DDR2 SDRAM High-Performance Controller, refer to the DDR and DDR2 SDRAM High-Performance Controller User Guide.

You can simulate the memory interface with the MegaWizard Plug-In Manager-generated IP functional simulation model. You should use this model in conjunction with your own driver or the testbench generated by the MegaWizard that issues read and write operations. The memory model file is also automatically generated by the MegaWizard in the testbench.

Use the functional simulation model with any Altera-supported VHDL or Verilog HDL simulator. This walkthrough uses the Quartus II NativeLink feature to run Altera-ModelSim® software edition to perform the simulation.
For more information about how to set up the simulation in Quartus II software using NativeLink, refer to the DDR and DDR2 SDRAM High-Performance Controller User Guide.

Figure 12 shows an Altera ModelSim RTL simulation.

**Step 4: Add Constraints**

When you create your memory controller, the MegaWizard generates the following constraint files for timing constraint and pin assignment.

- DDR2_phy_ddr_timing.sdc
- DDR2_pin_assignments.tcl
The DDR2_phy_ddr_timing.sdc file is used for constraining the clock and input/output delay in the ALTMEMPHY megafunction. Enable the TimeQuest Timing Analyzer before compiling your design. To enable the TimeQuest Timing Analyzer, perform the following steps:

1. On the Assignments menu, click Settings. The Settings dialog box appears.
2. From the Category list, click Timing Analysis Settings and select Use TimeQuest Timing Analyzer during compilation.
3. Click OK.

Next, to add the timing constraints, perform the following steps:

2. Specify the SDC file and click Add (Figure 13).
3. Click OK.
The DDR2_pin_assignments.tcl file is used for making the I/O standard assignment to the memory interface pins. To run the Tcl file, on the Tools menu, click Tcl Scripts. Select the Tcl file and click Run, as shown in Figure 14. Upon execution of the file, the information is added into your Quartus II assignment. Alternatively, you can also use the Tcl Console to run the Tcl file.
After running the DDR2_pin_assignments.tcl file, you can assign the I/O locations based on your board design in the Assignment Editor or Pin Planner. In this design example, assign the I/O locations based on the Cyclone III FPGA development kit board. The 32-bit interface is located at the bottom I/O banks of the device.

**Step 5: Compile Design and Verify Timing Closure**

Before compiling the design, set the top level entity of the project to the example design created by the High-Performance Controller in the previous section. To do so, perform the following steps:

1. On the File menu, click Open.
2. Browse to `<variation name>_example_top` and click Open.
3. On the Project menu, click Set as Top-Level Entity.

After setting the design example as the top level entity, set the Quartus II software to ensure the remaining unconstrained paths are routed with the highest speed and efficiency. To do this, perform the following steps:
Example Walkthrough for 167-MHz DDR2 SDRAM Interface

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.

2. From the **Category** list, click **Analysis & Synthesis Settings**.

3. Define the **Optimization Technique** setting. The default setting is **Balanced**, as shown in Figure 15.

**Figure 15. Default Setting for Optimization Technique**

![](image)

4. Next, from the **Category** list, click **Fitter Settings** and set the **Fitter effort**. The default setting is **Auto Fit**, as shown in Figure 16.
Design Guidelines for Implementing DDR and DDR2 SDRAM Interfaces in Cyclone III Devices

Figure 16. Default Setting for Filter Effort

You are now ready to compile your design. To do so, perform the following steps:

1. On the Processing menu, click Start Compilation to compile the design.

2. After compilation is successful, on the Tools menu, select TimeQuest Timing Analyzer.

3. Generate the timing margin report for your memory interface design by executing the Report DDR function from the Tasks pane of the TimeQuest Timing Analyzer window, as shown in Figure 17.
Executing the **Report DDR** task automatically runs the 
<variation_name>_phy_report_timing.tcl timing margin report script 
generated by the MegaWizard Plug-In Manager when the megafunction 
variation was created.

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**.

![TimeQuest Timing Analyzer Window](image)

**Figure 17. TimeQuest Timing Analyzer Window**

Figure 18 shows the output of the Report DDR task. The Report pane 
contains a new folder titled **DDR** with detailed timing information on the 
most critical paths, and a timing margin summary similar to the one 
reported on the TimeQuest Console.

![Figure 18](image)
The report timing script provides information about the following margins and paths:

- Address/command setup and hold margin
- Half-rate address/command setup and hold margin
- Core setup and hold margin
- Core reset/removal setup and hold margin
- DQS vs CK setup and hold margin
- Mimic path
- Write setup and hold margin
- Read capture setup and hold margin
For more information about the timing analysis, refer to *AN 438: Constraining and Analyzing Timing for External Memory Interfaces in Stratix III and Cyclone III Devices*.

**Step 6: Adjust Constraints**

Even though the MegaWizard generates the controller with the correct settings and you do not see any timing violation, you can still manually adjust some of the constraints in order to get the best timing that suits your system. The timing margin report shows the address and command datapath with a 2.668 ns setup and 1.601 ns hold margin. Adjusting the clock that is regulating the address and command output registers can make the setup and hold time more balanced by decreasing the setup margin and increasing the hold margin on the address and command datapath. To find out which clock is clocking the address and command registers, click on the address/command report in the Report panel in TimeQuest timing analyzer and select the path for the setup or hold time for the address and command datapath as Path Summary (Figure 19) or in Waveform View (Figure 20).
Figure 19. Path Summary for the Address and Command Datapath

Path Summary

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Node</td>
<td>...init_mem_phy_instDDR2_phy_at_mem_phy_ci_inst[0]phy_i...auto_generated[1][1][2]</td>
</tr>
<tr>
<td>To Node</td>
<td>mem_ci_n0(0)</td>
</tr>
<tr>
<td>Launch Clock</td>
<td>...phy_instDDR2_phy_at_mem_phy_ci_inst[0]phy_i...auto_generated[1][1][2] (INVERTED)</td>
</tr>
<tr>
<td>Latch Clock</td>
<td>...phy_instDDR2_phy_at_mem_phy_ci_inst[0]phy_i...auto_generated[1][1][2] (INVERTED)</td>
</tr>
<tr>
<td>Data Arrival Time</td>
<td>5813</td>
</tr>
<tr>
<td>Data Requested Time</td>
<td>0.281</td>
</tr>
<tr>
<td>Slack</td>
<td>2.288</td>
</tr>
</tbody>
</table>
The report indicates that \( \text{clk2} \) of the PLL is clocking the address and command registers. Go to the PLL megafunction and change the phase setting of \( \text{clk2} \). For this design, the initial phase setting of \( \text{clk2} \) is set to –90° with reference to the system clock. By adjusting \( \text{clk2} \) to later than –90° (meaning launching the address and command later), the setup margin is decreased and the hold margin is increased.

After modifying the \( \text{clk2} \) phase setting to –55°, recompile the design for the new PLL setting to take effect. Run the report timing script again. Figure 21 shows the timing margin reported in the Quartus II software after adjusting the phase setting of \( \text{clk2} \). The address and command datapath now has a more balanced 2.117 ns setup and 2.128 ns hold margin.
Step 7: Determine Board Design Constraints

The Cyclone III FPGA supports the series on-chip termination (OCT) to improve signal integrity and simplify the board design. The Cyclone III device supports both OCT with or without calibration. In addition, you can choose the resistance to be either 25 $\Omega$ or 50 $\Omega$, depending on the I/O standards you are using for the memory interface and the termination scheme.

Apart from that, the DDR2 SDRAM supports the dynamic parallel on-die termination (ODT) feature that you can turn on when the FPGA is writing to the DDR2 SDRAM memory and turn off when the FPGA is reading from the DDR2 SDRAM memory. The ODT features are available in settings of 150 $\Omega$, 75 $\Omega$, and 50 $\Omega$. The 50-$\Omega$ setting is only available in DDR2 SDRAM with operating frequencies greater than 267 MHz.
Refer to the respective memory data sheet for additional information about the available settings for the ODT and the output driver impedance features, and the timing requirements for driving the ODT pin in DDR2 SDRAM.

For this design walkthrough, which is targeted for the Cyclone III FPGA development kit, drive strength setting is used together with Class I termination. Using Class I termination allows a higher maximum DDR2 SDRAM clock frequency and reduces the number of external resistors required on the board. You can adjust the current strength for the output pin of the Cyclone III device according to your board setup. If the current strength is too high, you might see excessive overshoot and undershoot of your signal. Use the oscilloscope to check the signal overshoot and undershoot.

Figure 22 shows the setup for write operation to the DDR2 SDRAM memory with Class I termination together with the drive strength setting of the Cyclone III FPGA device. In this setup, the driver’s (FPGA) output impedance matches that of the transmission line resulting in optimal signal transmission to the DDR2 SDRAM memory. On the receiver (DDR2 SDRAM memory) side, it is properly terminated with matching impedance to the transmission line, through the external pull-up resistor to $V_{TT}$ to eliminate any ringing or reflection.

**Figure 22. Write Operation to DDR2 SDRAM Memory with Class I Termination**

![Diagram of write operation to DDR2 SDRAM memory with Class I termination and drive strength setting.](image)

**Figure 23** shows the setup for read operation from the DDR2 SDRAM memory.

**Figure 23**

![Diagram of read operation from DDR2 SDRAM memory.](image)
If you choose not to use the drive strength setting, you can use the OCT feature of the Cyclone III FPGA device instead.

Finally, the loading seen by the FPGA during writes to the memory is different between a system using dual-inline memory modules (DIMMs) versus a system using components. The additional loading from the DIMM connector can reduce the edge rates of the signals arriving at the memory, thus affecting available timing margin.

For more information about the Cyclone III OCT, refer to the Cyclone III Device I/O Features chapter in volume 1 of the Cyclone III Device Handbook.

For detailed information about understanding the different effects on signal integrity design, refer to AN 408: DDR2 Memory Interface Termination, Drive Strength & Loading Design Guidelines.

Conclusion

Cyclone III devices have dedicated circuitry to interface with DDR2 SDRAM at speeds up to 200 MHz with comfortable and consistent margins. The advanced clocking features available in Cyclone III devices allow a high-performance, versatile interface to DDR2 and DDR SDRAM. For applications requiring lower power consumption and the greater memory bandwidth offered by DDR2 and DDR SDRAM, Altera offers a complete memory solution for Cyclone III FPGA devices.
References

References

- MT47H32M16 DDR2 SDRAM Data Sheet, Micron Technology, Inc.

Referenced Documents

This application note references the following documents:

- ALTMEMPHY Megafuction User Guide
- AN 408: DDR2 Memory Interface Termination, Drive Strength and Loading Design Guidelines
- AN 438: Constraining and Analyzing Timing for External Memory Interfaces in Stratix III and Cyclone III Devices
- AN 462: Implementing Multiple Memory Interfaces Using the ALTMEMPHY Megafunction
- Cyclone III Device I/O Features chapter in volume 1 of the Cyclone III Device Handbook
- DDR and DDR2 SDRAM Controller Compiler User Guide
- DDR and DDR2 SDRAM High-Performance Controller User Guide
- Design Debugging Using the SignalTap II Embedded Logic Analyzer chapter in volume 3 of the Quartus II Handbook
- External Memory Interfaces in Cyclone III Devices chapter in volume 1 of the Cyclone III Device Handbook
- Selecting the Right High-Speed Memory Technology for Your System white paper
- The Quartus II TimeQuest Timing Analyzer chapter in volume 3 of the Quartus II Handbook.
Table 5 shows the revision history for this document.

<table>
<thead>
<tr>
<th>Date and Document Version</th>
<th>Changes Made</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2008 v2.1</td>
<td>● Replaced Table 1 and Table 2.</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>● Added Table 3.</td>
<td></td>
</tr>
<tr>
<td>December 2007 v2.0</td>
<td>● Updated “Introduction” section.</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>● Added new Table 2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Updated Figure 1.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Updated “FPGA Design Flow” section including changes to Steps 1, 2, 3, 4, 5, 6 and added new Table 4 and Step 7.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Removed old Figure 2.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Updated “Example Walkthrough for 167-MHz DDR2 SDRAM Interface” section including changes to Steps 1, 2, 3, 4, 5, 6, 7.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Updated Figures 3, 4, 5, 6, 7, 8, 13, 14, 19, 21.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Added new Figures 8, 9, 10, 11, 15, 16, 20, 22.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Removed old Figures 9 and 21.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Added “Referenced Documents” section.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>● Added “Design Checklist”.</td>
<td></td>
</tr>
<tr>
<td>March 2007 v1.0</td>
<td>Initial Release.</td>
<td>—</td>
</tr>
</tbody>
</table>
Design Checklist

This section contains a design checklist that you can use when implementing DDR2 and DDR2 SDRAM memory interfaces in Cyclone III devices. Use the checklist to verify that you have followed the guidelines for each stage of your design.

<table>
<thead>
<tr>
<th>Select Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Done</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instantiate PHY and Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Done</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Functional Simulation
### Design Guidelines for Implementing DDR and DDR2 SDRAM Interfaces in Cyclone III Devices

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Done</strong></td>
<td><strong>N/A</strong></td>
<td></td>
</tr>
<tr>
<td><strong>4</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you simulated your design using the RTL functional model?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Use the ALTMEMPHY megafucntion functional simulation model in conjunction with your own driver logic/testbench, and a memory model to ensure proper read/write transactions to the memory.

### Timing Closure

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Done</strong></td>
<td><strong>N/A</strong></td>
<td></td>
</tr>
<tr>
<td><strong>5</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you added constraints to the PHY and the rest of your design?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The ALTMEMPHY megafucntion is constrained when you use the generated `.sdc` file and `.tcl` files. However, you may need to adjust these settings to best fit your memory interface configuration.

Add pin assignment constraints and pin loading constraints to your design. Ensure that generic pin names used in the constraint script are modified to match your top-level pin names. Note that the loading on memory interface pins is dependent on your board topology (memory components, single DIMM, multiple DIMMs, single rank DIMM, and so on).

| **6** |   |   |
| Have you compiled your design and verified timing closure using all available models? |

Run the `<variation_name>_report_timing.tcl` file to generate a custom timing report for each of your ALTMEMPHY megafunction instances. Repeat this process using all device timing models (slow 0°C, slow 85°C, fast 0°C).

| **7** |   |   |
| If there are timing violations, have you adjusted your constraints to optimize timing? |

Adjust PLL clock phase-shift settings or appropriate timing/location assignments margins for the various timing paths within the ALTMEMPHY megafunction.

### Gate-Level Simulation

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Done</strong></td>
<td><strong>N/A</strong></td>
<td></td>
</tr>
<tr>
<td><strong>8</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Have you performed a timing simulation on your design?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Board-Level Considerations

Have you selected the termination scheme and drive-strength settings for all the memory interface signals on the memory side and the FPGA side?

Ensure that appropriate termination and drive strength settings are applied on all the memory interface signals, and verified using board-level simulations.

Cyclone III devices support on-chip termination. Altera recommends the calibrated parallel OCT 50 setting for unidirectional input signals from the memory, and the calibrated series OCT 50 setting for unidirectional output signals to the memory. When using the OCT feature on the Stratix III device, the programmable drive-strength feature is unavailable. If there are multiple loads on certain FPGA output pins (for example, if the address bus is shared across many memory devices), use of maximum drive-strength setting may be preferred over the series OCT setting. Use board-level simulations to pick the optimal setting for best signal integrity.

On the memory side, Altera recommends the use of external parallel termination on input signals to the memory (write data, address, command, and clock signals).

Have you performed board-level simulations to ensure electrical and timing margins for your memory interface?

Ensure you have a sufficient eye opening using simulations. Be sure to use the latest FPGA and memory IBIS models, board trace characteristics, drive strength and termination settings in your simulation.

Any timing uncertainties at the board level that you calculate using such simulations must be used to adjust the input timing constraints to ensure the accuracy of the Quartus II timing margin reports.

System Verification

Have you verified functionality of the memory interface in the system?
Design Guidelines for Implementing DDR and DDR2 SDRAM Interfaces in Cyclone III Devices