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

May 2008, ver 2.1 Application Note 445

Introduction

Cyclone® III devices support interfacing to both DDR2 and DDR SDRAM devices and modules. Altera provides the easy-to-use ALTMEMPHY megafunction to implement robust auto-calibrating interfaces to DDR2 and DDR SDRAM devices. You can configure this megafunction 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 megafunction with half-rate or full-rate controller, refer to the *ALTMEMPHY Megafunction User Guide*.

Table 1. Cyclone III Maximum Clock Rate Support for External Memory Interfaces with Half-Rate Controller *Notes (1), (2), (3), (4)*

Memory Standard	I/O Standard	Commercial									Industrial			Automotive		
		–6 Speed Grade (MHz)			–7 Speed Grade (MHz)			–8 Speed Grade (MHz)			–7 Speed Grade (MHz)			–7 Speed Grade (MHz)		
		Column I/O Banks	Row I/O Banks	Hybrid Mode	Column I/O Banks	Row I/O Banks	Hybrid Mode	Column I/O Banks	Row I/O Banks	Hybrid Mode	Column I/O Banks	Row I/O Banks	Hybrid Mode	Column I/O Banks	Row I/O Banks	Hybrid Mode
DDR2 SDRAM	SSTL-18 Class I/II	200	167	150	167	150	133	167	133	125	167	150	133	167	133	125
DDR SDRAM	SSTL-2 Class I/II	167	150	133	150	133	125	133	125	100	150	133	125	133	125	100

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 *Notes (1), (2), (3), (4)*

Memory Standard	I/O Standard	Commercial						Industrial		Automotive	
		–6 Speed Grade (MHz)		–7 Speed Grade (MHz)		–8 Speed Grade (MHz)		–7 Speed Grade (MHz)		–7 Speed Grade (MHz)	
		Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks	Column I/O Banks	Row I/O Banks
DDR2 SDRAM	SSTL-18 Class I/II	167	167	150	150	133	133	150	150	133	133
DDR SDRAM	SSTL-2 Class I/II	167	150	150	133	133	125	150	133	133	125

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 *Note (1)*

Temperature & Speed Grade	Interface location	I/O Standard SSTL-18	Memory Speed Grade (MHz)	Interface f_{max} (MHz)
C6	Top/Bottom	Class I	267	200
		Class II	333	
	Left/Right	Class I/II	200	167
	Hybrid	Class I/II	200	150
C7	Top/Bottom	Class I/II	200	167
	Left/Right	Class I/II	200	150
	Hybrid	Class I/II	200	133

Table 3. Required DDR2 Memory Device Speed Grade for External Memory Interface Note (1)

Temperature & Speed Grade	Interface location	I/O Standard SSTL-18	Memory Speed Grade (MHz)	Interface f_{max} (MHz)
C8	Top/Bottom *(2)*	Class I/II	267	167
	Left/Right	Class I/II	200	133
	Hybrid	Class I/II	200	125
I7	Top/Bottom	Class I/II	267	167
	Left/Right	Class I/II	200	150
	Hybrid	Class I/II	200	133
A7	Top/Bottom	Class I	267	167
		Class II	333	
	Left/Right	Class I/II	200	133
	Hybrid	Class I/II	200	125

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.

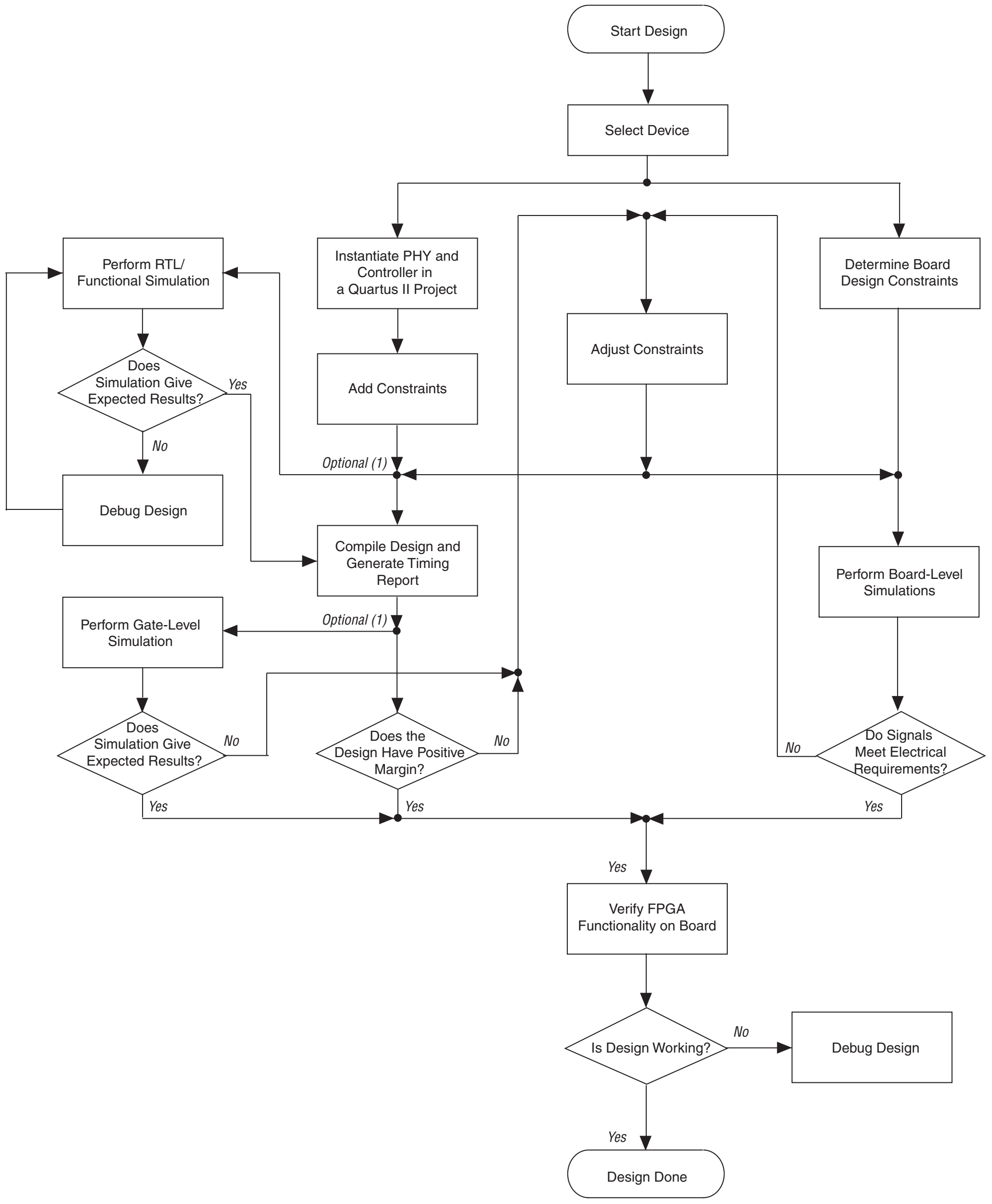
(2) For Q240 C8 device 167 MHz performance on Top/Bottom interface, 333 MHz DDR2 memory device is required.

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.

Figure 1. Design Flow for Implementing External Memory Interfaces in Cyclone III Devices


Start Design
Select Device
Perform RTL/ Functional Simulation
Instantiate PHY and Controller in a Quartus II Project
Determine Board Design Constraints
Adjust Constraints
Does Simulation Give Expected Results?
Yes
Add Constraints
No
Optional (1)
Debug Design
Compile Design and Generate Timing Report
Perform Board-Level Simulations
Perform Gate-Level Simulation
Optional (1)
Does Simulation Give Expected Results?
No
Does the Design Have Positive Margin?
No
No
Do Signals Meet Electrical Requirements?
Yes
Yes
Yes
Yes
Verify FPGA Functionality on Board
Is Design Working?
No
Debug Design
Yes
Design Done

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.

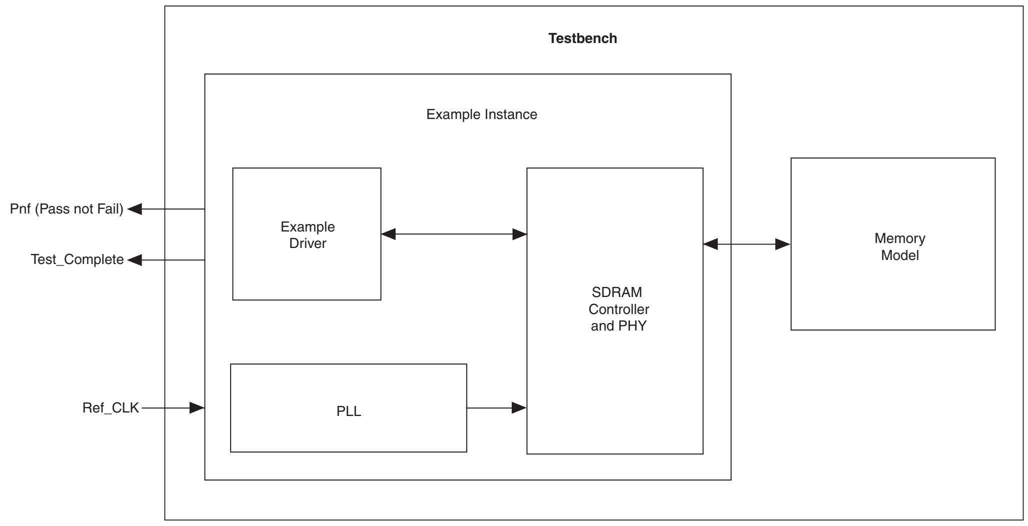
For information about selecting the different memory types, differences between the memory types, and information about the FPGA density and package support for the different memory types, refer to the *External Memory Interfaces in Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device 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 megafunction to build the data path or PHY, and instantiates the phase-locked loop (PLL).
- Use Altera's ALTMEMPHY megafunction alone to build the PHY and instantiate the PLL. You can create your own controller to work with the ALTMEMPHY megafunction.

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 megafunction 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 megafunction. This megafunction features a license-free physical interface that you can use with the standard Altera controller or your own custom controller. The ALTMEMPHY megafunction 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 megafunction, 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 megafunction. 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 megafunction or the memory controller, refer to the *ALTMEMPHY Megafunction 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 megafunction 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 megafunction 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 megafunction 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 megafunction or the High-Performance Controller, based on the DDR2 and DDR SDRAM data sheet and tolerances from the board layout. The ALTMEMPHY megafunction uses TimeQuest timing constraints along with the timing driven fitter to achieve good timing closure.

After instantiating the ALTMEMPHY megafunction, 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 megafunction. 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 4. Memory Interface Pin Connections Between DDR2 and Cyclone III Devices

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

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*.

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.

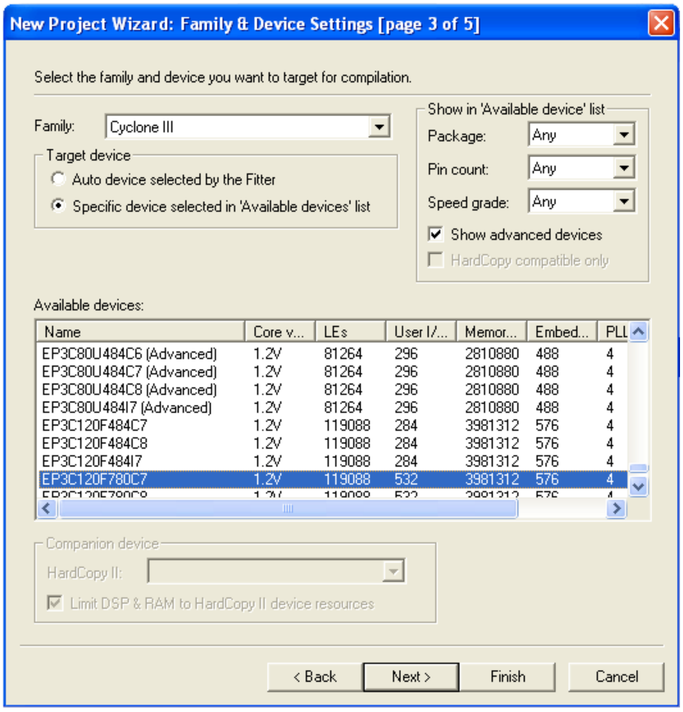
Top/bottom DQ groups provide the fastest performance. The Quartus II software automatically uses the top/bottom DQ groups if they are available. Combining top/bottom DQ groups with left/right DQ groups for a single interface is not recommended and may result in a worse performance.

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.*

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



New Project Wizard: Family & Device Settings [page 3 of 5]

Select the family and device you want to target for compilation.

Family: Cyclone III

Target device
- Auto device selected by the Fitter
- [✓ Specific device selected in 'Available devices' list]

Show in 'Available device' list
- Package: Any
- Pin count: Any
- Speed grade: Any
- [✓] Show advanced devices
- [ ] HardCopy compatible only

Available devices:

Name	Core v...	LEs	User I/...	Memor...	Embed...	PLL
EP3C80U484C6 (Advanced)	1.2V	81264	296	2810880	488	4
EP3C80U484C7 (Advanced)	1.2V	81264	296	2810880	488	4
EP3C80U484C8 (Advanced)	1.2V	81264	296	2810880	488	4
EP3C80U484I7 (Advanced)	1.2V	81264	296	2810880	488	4
EP3C120F484C7	1.2V	119088	284	3981312	576	4
EP3C120F484C8	1.2V	119088	284	3981312	576	4
EP3C120F484I7	1.2V	119088	284	3981312	576	4
EP3C120F780C7	1.2V	119088	532	3981312	576	4

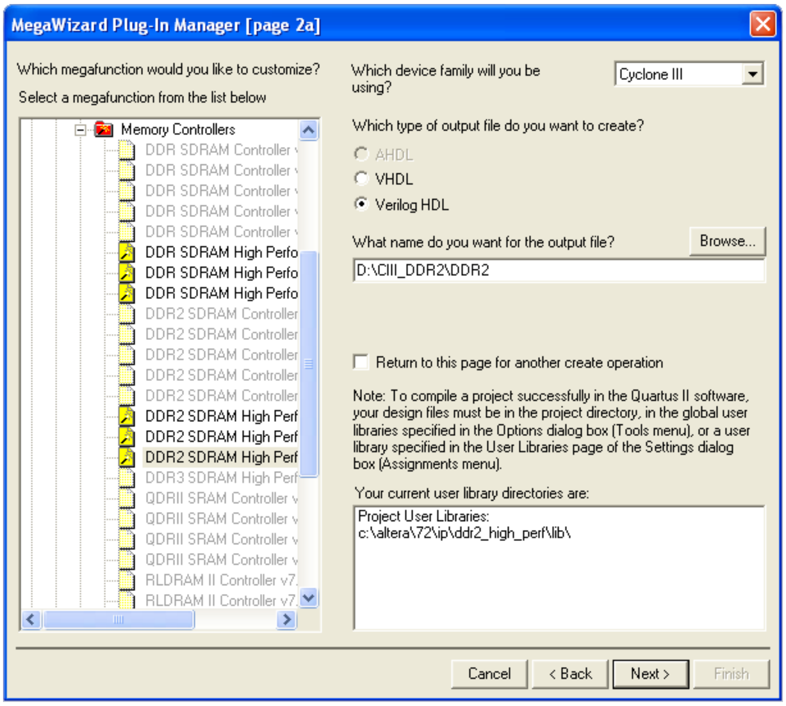
Companion device
- HardCopy II:
- [✓] Limit DSP & RAM to HardCopy II device resources

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 megafunction 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.

Figure 4. Invoking the DDR2 SDRAM High-Performance Controller using the MegaWizard Plug-In Manager



Which device family will you be using? Cyclone III

Which type of output file do you want to create?

AHDL [ ] VHDL [✓ Verilog HDL]

What name do you want for the output file? D:\CIII_DDR2\DDR2

Return to this page for another create operation

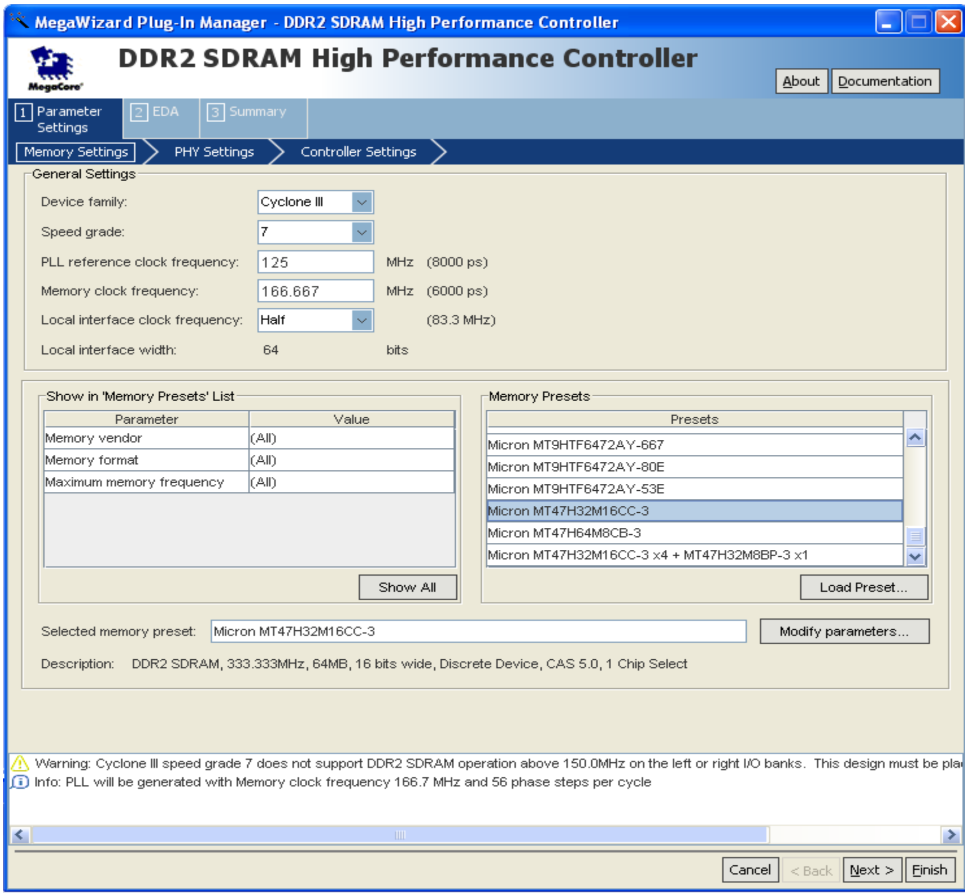
Note: To compile a project successfully in the Quartus II software, your design files must be in the project directory, in the global user libraries specified in the Options dialog box (Tools menu), or a user library specified in the User Libraries page of the Settings dialog box (Assignments menu).

Your current user library directories are:

Project User Libraries:
c:\altera\72\ip\ddr2_high_perf\lib\

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.

Figure 5. Configuring the DDR2 SDRAM High-Performance Controller for the DDR2 Memory Interface



Device family: Cyclone III
Speed grade: 7
PLL reference clock frequency: 125 MHz (8000 ps)
Memory clock frequency: 166.667 MHz (6000 ps)
Local interface clock frequency: Half (83.3 MHz)
Local interface width: 64 bits

Parameter	Value
Memory vendor	(All)
Memory format	(All)
Maximum memory frequency	(All)

Presets
Micron MT9HTF6472AY-667
Micron MT9HTF6472AY-80E
Micron MT9HTF6472AY-53E
Micron MT47H32M16CC-3
Micron MT47H64M8CB-3
Micron MT47H32M16CC-3 x4 + MT47H32M8BP-3 x1

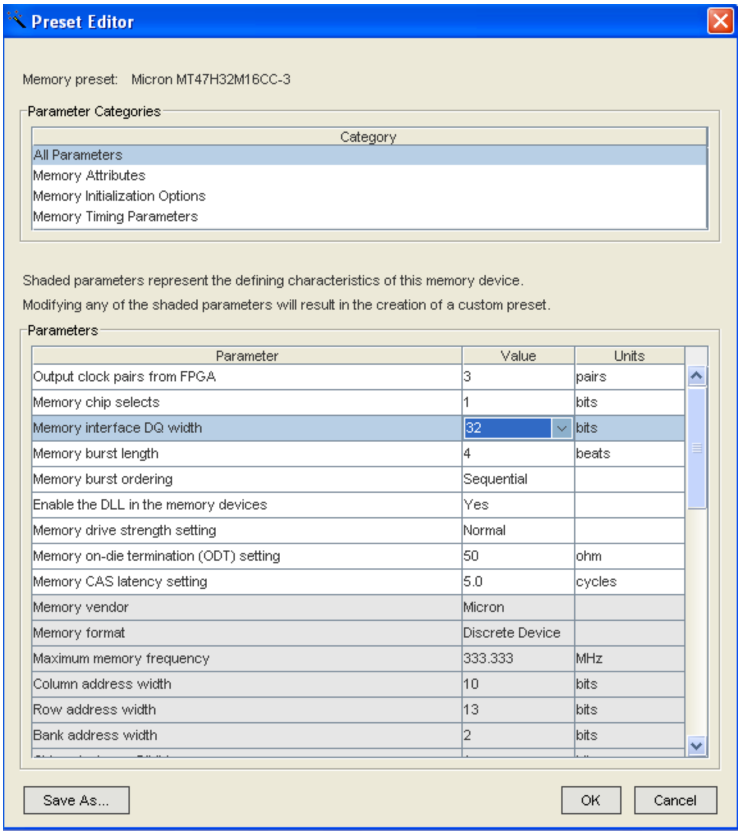
Selected memory preset: Micron MT47H32M16CC-3

Description: DDR2 SDRAM, 333.333MHz, 64MB, 16 bits wide, Discrete Device, CAS 5.0, 1 Chip Select

```
Warning: Cyclone III speed grade 7 does not support DDR2 SDRAM operation above 150.0MHz on the left or right I/O banks.  This design must be pla
Info: PLL will be generated with Memory clock frequency 166.7 MHz and 56 phase steps per cycle
```

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



Memory preset: Micron MT47H32M16CC-3

Parameter Categories

Category
All Parameters
Memory Attributes
Memory Initialization Options
Memory Timing Parameters

Shaded parameters represent the defining characteristics of this memory device.
Modifying any of the shaded parameters will result in the creation of a custom preset.

Parameters

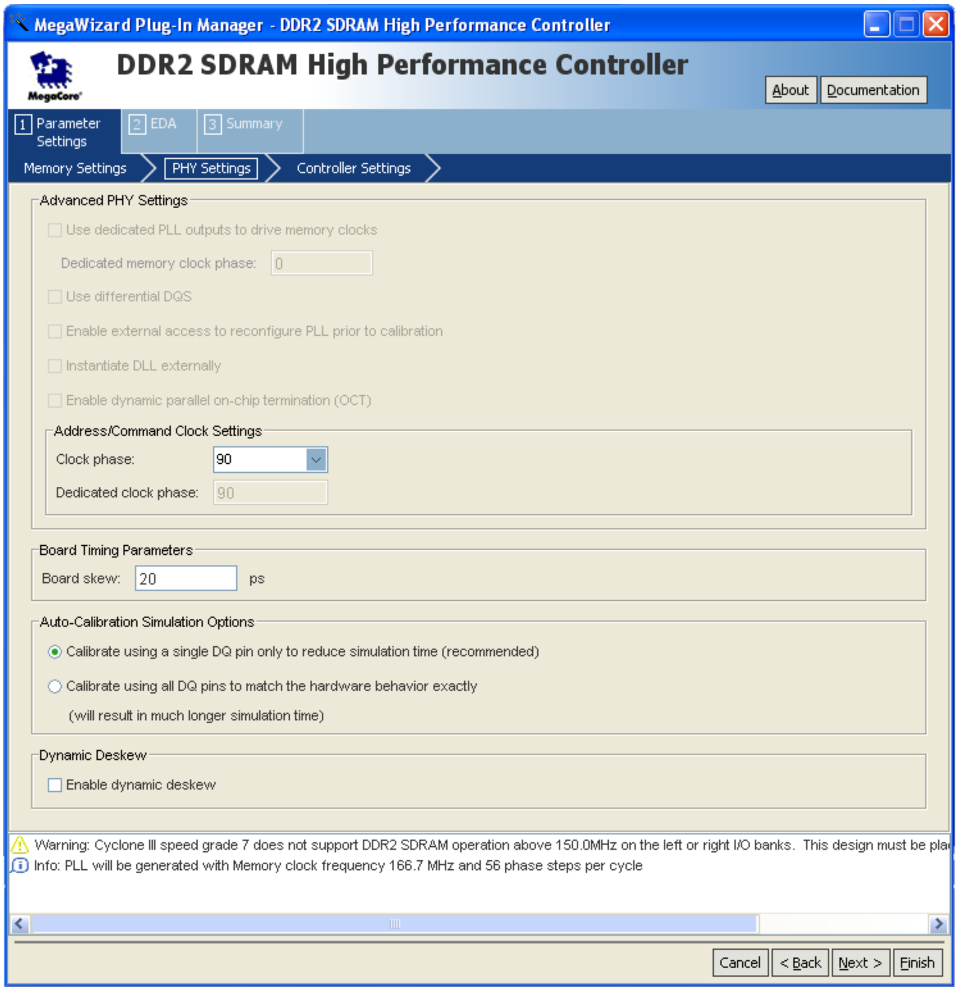
Parameter	Value	Units
Output clock pairs from FPGA	3	pairs
Memory chip selects	1	bits
Memory interface DQ width	32	bits
Memory burst length	4	beats
Memory burst ordering	Sequential	
Enable the DLL in the memory devices	Yes	
Memory drive strength setting	Normal	
Memory on-die termination (ODT) setting	50	ohm
Memory CAS latency setting	5.0	cycles
Memory vendor	Micron	
Memory format	Discrete Device	
Maximum memory frequency	333.333	MHz
Column address width	10	bits
Row address width	13	bits
Bank address width	2	bits

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.

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 t_{DQSCK} in the DDR2 SDRAM data sheet.

The DDR2 SDRAM has a write requirement (t_{DQSS}) that states the positive edge of the `DQS` signal on writes must be within ± 25% (± 90°) of the positive edge of the DDR2 SDRAM clock input. t_{DQSS} 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 t_{DQSS}.

Figure 7. DDR2 SDRAM High-Performance Controller PHY Settings



Advanced PHY Settings

- [ ] Use dedicated PLL outputs to drive memory clocks
- Dedicated memory clock phase: 0
- [ ] Use differential DQS
- [ ] Enable external access to reconfigure PLL prior to calibration
- [ ] Instantiate DLL externally
- [ ] Enable dynamic parallel on-chip termination (OCT)

Address/Command Clock Settings

- Clock phase: 90
- Dedicated clock phase: 90

Board Timing Parameters

- Board skew: 20 ps

Auto-Calibration Simulation Options

- [✓ Calibrate using a single DQ pin only to reduce simulation time (recommended)]
- Calibrate using all DQ pins to match the hardware behavior exactly (will result in much longer simulation time)

Dynamic Deskew

- [ ] Enable dynamic deskew

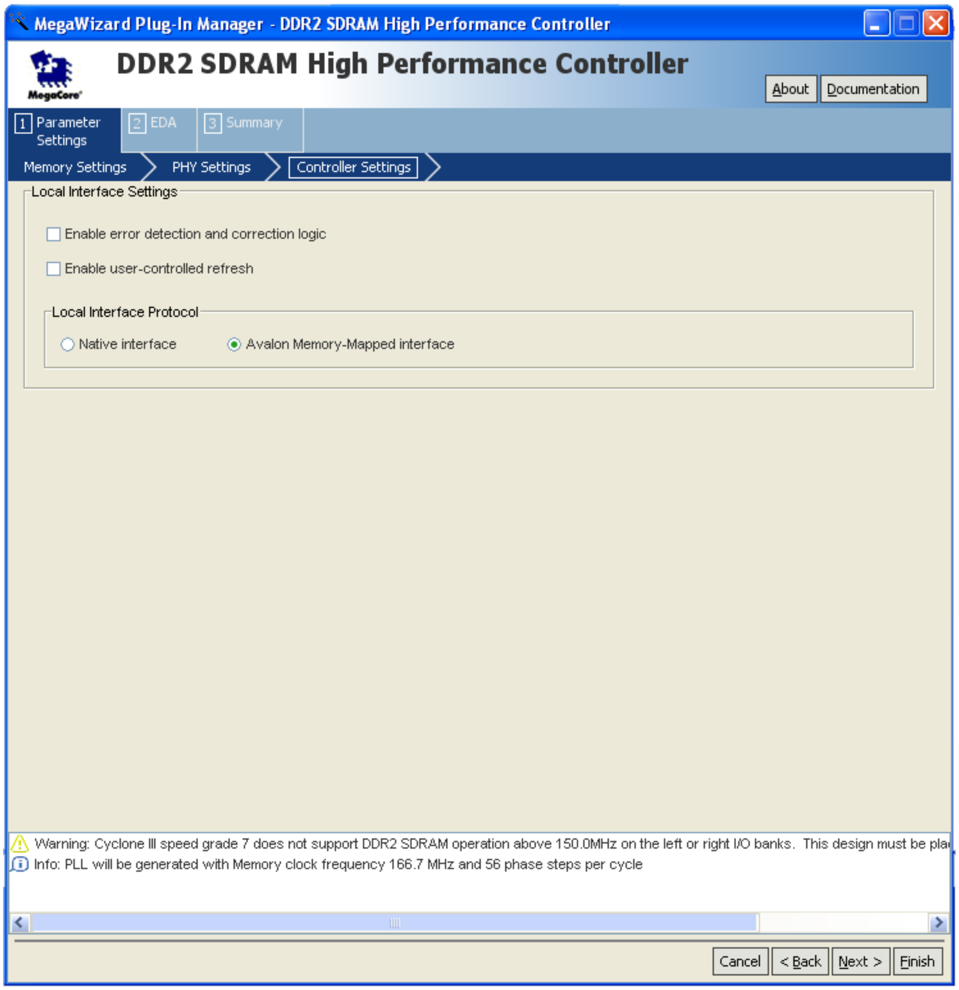
Warning: Cyclone III speed grade 7 does not support DDR2 SDRAM operation above 150.0MHz on the left or right I/O banks. This design must be pla

Info: PLL will be generated with Memory clock frequency 166.7 MHz and 56 phase steps per cycle

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.

Figure 8. DDR2 SDRAM High-Performance Controller Settings



Local Interface Settings

☐ Enable error detection and correction logic

☐ Enable user-controlled refresh

Local Interface Protocol

Native interface [✓ Avalon Memory-Mapped interface]

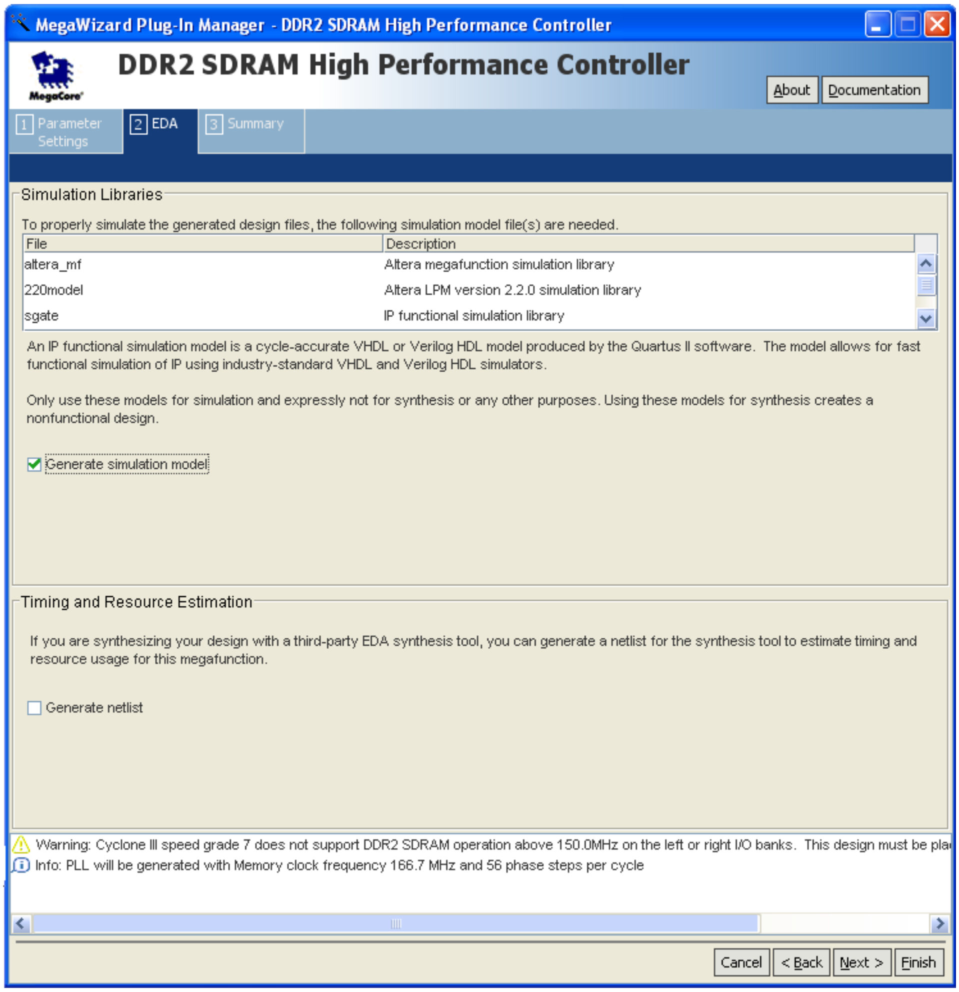
Warning: Cyclone III speed grade 7 does not support DDR2 SDRAM operation above 150.0MHz on the left or right I/O banks. This design must be pla

Info: PLL will be generated with Memory clock frequency 166.7 MHz and 56 phase steps per cycle

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


MegaWizard Plug-In Manager - DDR2 SDRAM High Performance Controller
DDR2 SDRAM High Performance Controller
MegaCore
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
Cancel
< Back
Next >
Finish

Simulation Libraries

To properly simulate the generated design files, the following simulation model file(s) are needed.

File	Description
altera_mf	Altera megafunction simulation library
220model	Altera LPM version 2.2.0 simulation library
sgate	IP functional simulation library

An IP functional simulation model is a cycle-accurate VHDL or Verilog HDL model produced by the Quartus II software. The model allows for fast functional simulation of IP using industry-standard VHDL and Verilog HDL simulators.

Only use these models for simulation and expressly not for synthesis or any other purposes. Using these models for synthesis creates a nonfunctional design.

[✓] Generate simulation model

Timing and Resource Estimation

If you are synthesizing your design with a third-party EDA synthesis tool, you can generate a netlist for the synthesis tool to estimate timing and resource usage for this megafunction.

[ ] Generate netlist

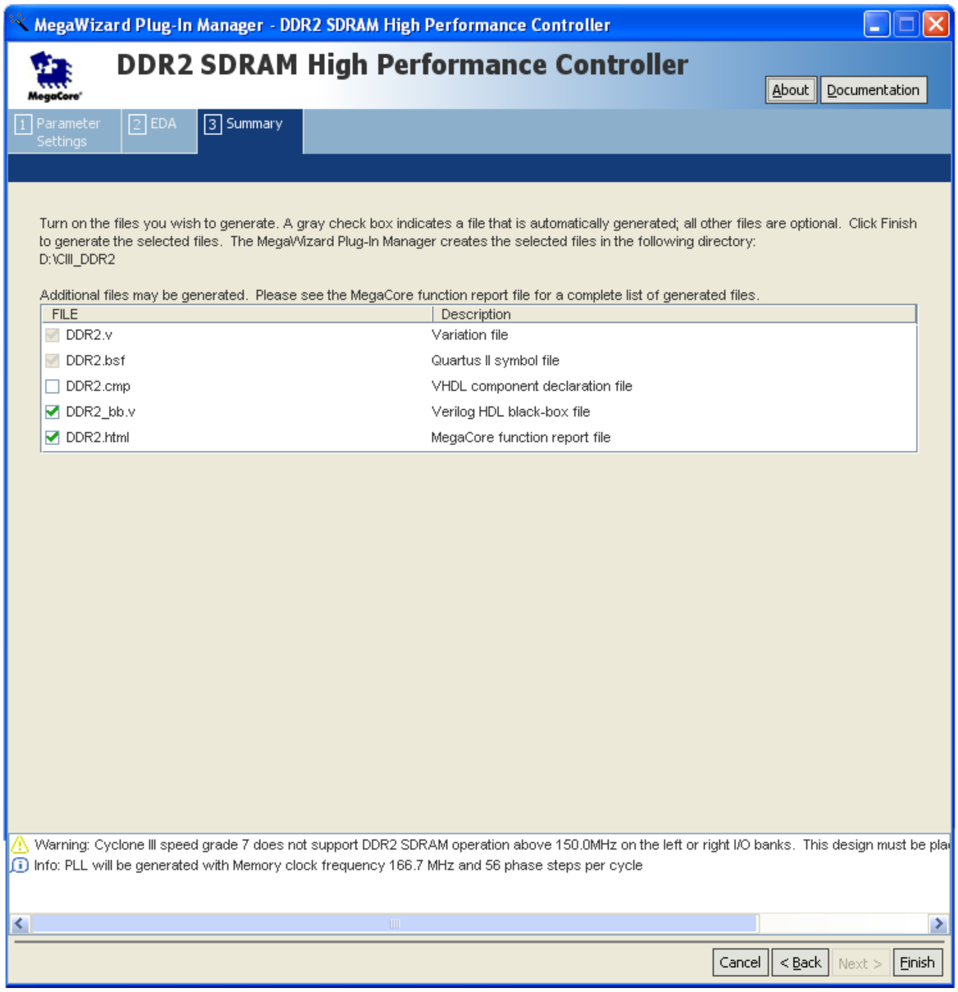
Warning: Cyclone III speed grade 7 does not support DDR2 SDRAM operation above 150.0MHz on the left or right I/O banks. This design must be pla

Info: PLL will be generated with Memory clock frequency 166.7 MHz and 56 phase steps per cycle

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.

Figure 10. Summary


MegaWizard Plug-In Manager - DDR2 SDRAM High Performance Controller
DDR2 SDRAM High Performance Controller
About
Documentation
1 Parameter Settings
2 EDA
3 Summary
Cancel
< Back
Next >
Finish

Turn on the files you wish to generate. A gray check box indicates a file that is automatically generated; all other files are optional. Click Finish to generate the selected files. The MegaWizard Plug-In Manager creates the selected files in the following directory:
D:\CIII_DDR2

Additional files may be generated. Please see the MegaCore function report file for a complete list of generated files.

FILE	Description
✓ DDR2.v	Variation file
✓ DDR2.bsf	Quartus II symbol file
☐ DDR2.cmp	VHDL component declaration file
✓ DDR2_bb.v	Verilog HDL black-box file
✓ DDR2.html	MegaCore function report file

Warning: Cyclone III speed grade 7 does not support DDR2 SDRAM operation above 150.0MHz on the left or right I/O banks. This design must be pla

Info: PLL will be generated with Memory clock frequency 166.7 MHz and 56 phase steps per cycle

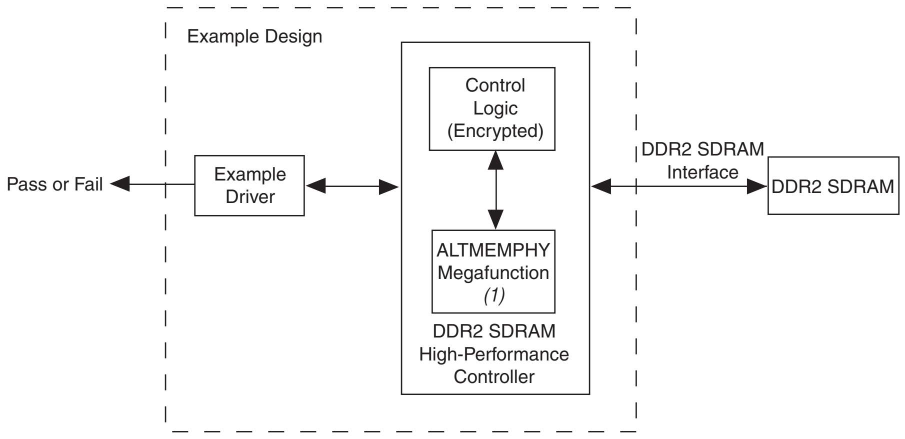
For detailed step-by-step instructions about configuring the DDR2 SDRAM High-Performance Controller, refer to the *DDR and DDR2 SDRAM High-Performance Controller User Guide*.

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


Example Design
Control Logic (Encrypted)
Pass or Fail
Example Driver
DDR2 SDRAM Interface
DDR2 SDRAM
ALTMEMPHY Megafunction (1)
DDR2 SDRAM High-Performance Controller

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*.

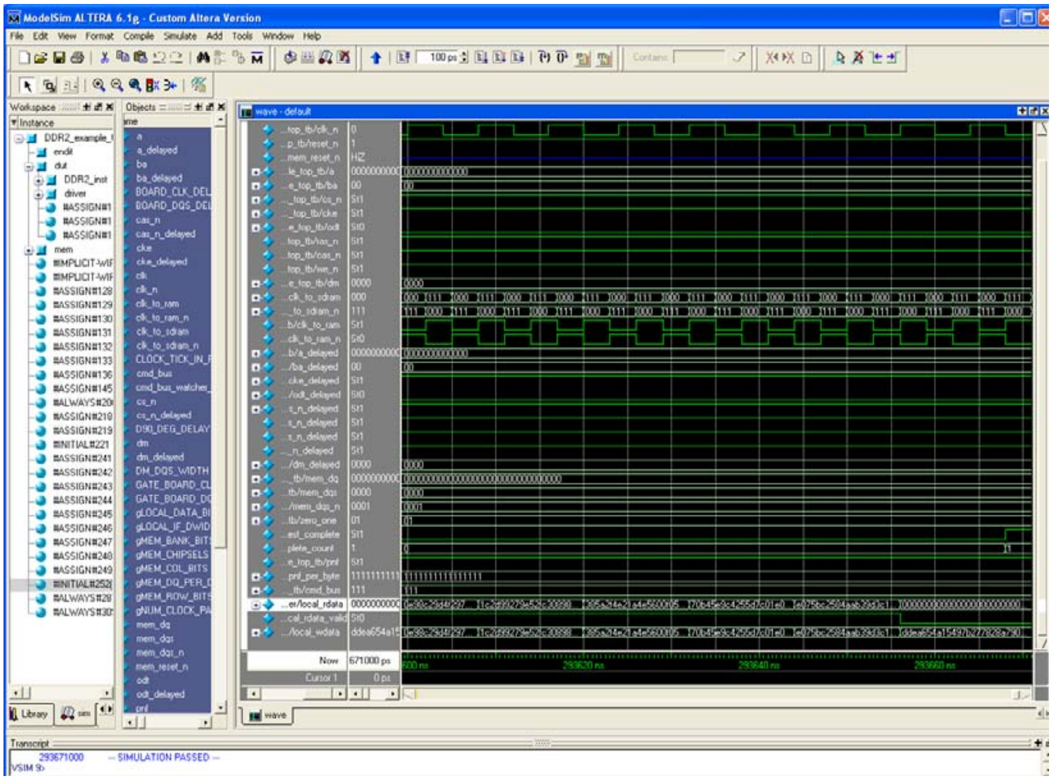
You can simulate the memory interface with the MegaWizard Plug-In Manager-generated IP functional simulation model. You should use this model in conjunction with your own driver or the testbench generated by the MegaWizard that issues read and write operations. The memory model file is also automatically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

Figure 12. Altera ModelSim RTL/Functional Simulation


ModelSim ALTERA 6.1g - Custom Altera Version
File Edit View Format Compile Simulate Add Tools Window Help
Workspace
Objects
wave - default
Library
sim
wave
Transcript

```
293671000        -- SIMULATION PASSED --
VSIM 9>
```

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**.

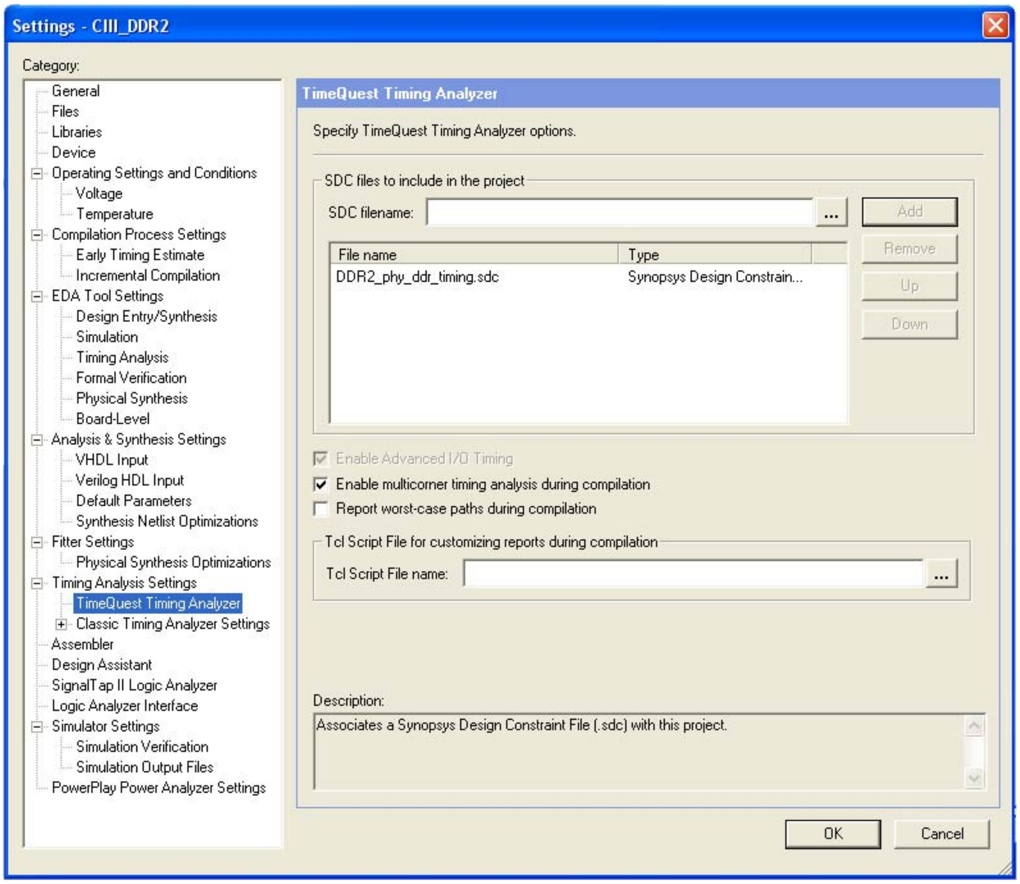
Next, to add the timing constraints, perform the following steps:

1. On the **Settings** dialog box, click **Timing Analysis Settings** and select **TimeQuest Timing Analyzer**. The **TimeQuest Timing Analyzer** page appears.

2. Specify the SDC file and click **Add** (Figure 13).

3. Click **OK**.

Figure 13. Specifying the Timing Constraint SDC File to the Example Design



Specify TimeQuest Timing Analyzer options.

SDC files to include in the project

SDC filename:

File name	Type
DDR2_phy_ddr_timing.sdc	Synopsys Design Constrain...

☑ Enable Advanced I/O Timing

☑ Enable multicorner timing analysis during compilation

☐ Report worst-case paths during compilation

Tcl Script File for customizing reports during compilation

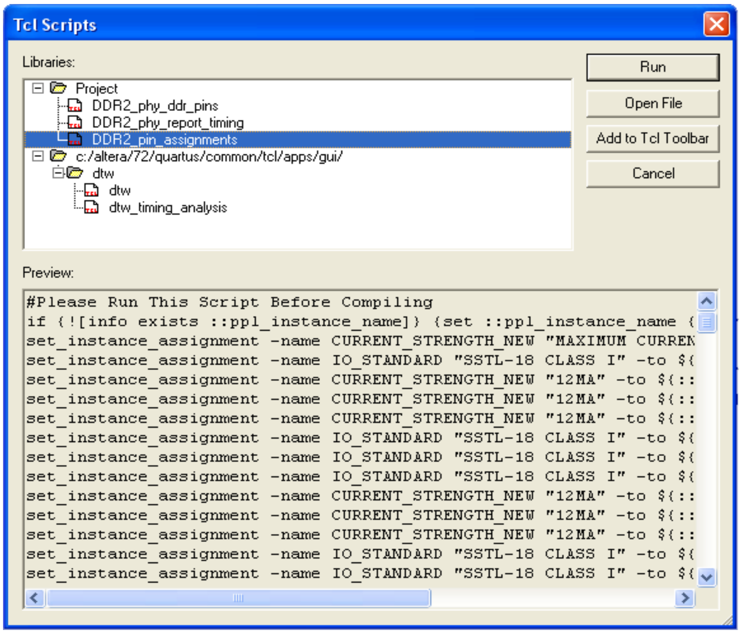
Tcl Script File name:

Description:

Associates a Synopsys Design Constraint File (.sdc) with this project.

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.

Figure 14. Running Pin Assignment Contraint Script in the Tcl Script Panel



```
#Please Run This Script Before Compiling
if {![info exists ::ppl_instance_name]} {set ::ppl_instance_name {
set_instance_assignment -name CURRENT_STRENGTH_NEW "MAXIMUM CURREN
set_instance_assignment -name IO_STANDARD "SSTL-18 CLASS I" -to ${
set_instance_assignment -name CURRENT_STRENGTH_NEW "12MA" -to ${::
set_instance_assignment -name CURRENT_STRENGTH_NEW "12MA" -to ${::
set_instance_assignment -name CURRENT_STRENGTH_NEW "12MA" -to ${::
set_instance_assignment -name IO_STANDARD "SSTL-18 CLASS I" -to ${
set_instance_assignment -name IO_STANDARD "SSTL-18 CLASS I" -to ${
set_instance_assignment -name IO_STANDARD "SSTL-18 CLASS I" -to ${
set_instance_assignment -name CURRENT_STRENGTH_NEW "12MA" -to ${::
set_instance_assignment -name CURRENT_STRENGTH_NEW "12MA" -to ${::
set_instance_assignment -name CURRENT_STRENGTH_NEW "12MA" -to ${::
set_instance_assignment -name IO_STANDARD "SSTL-18 CLASS I" -to ${
set_instance_assignment -name IO_STANDARD "SSTL-18 CLASS I" -to ${
```

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**.

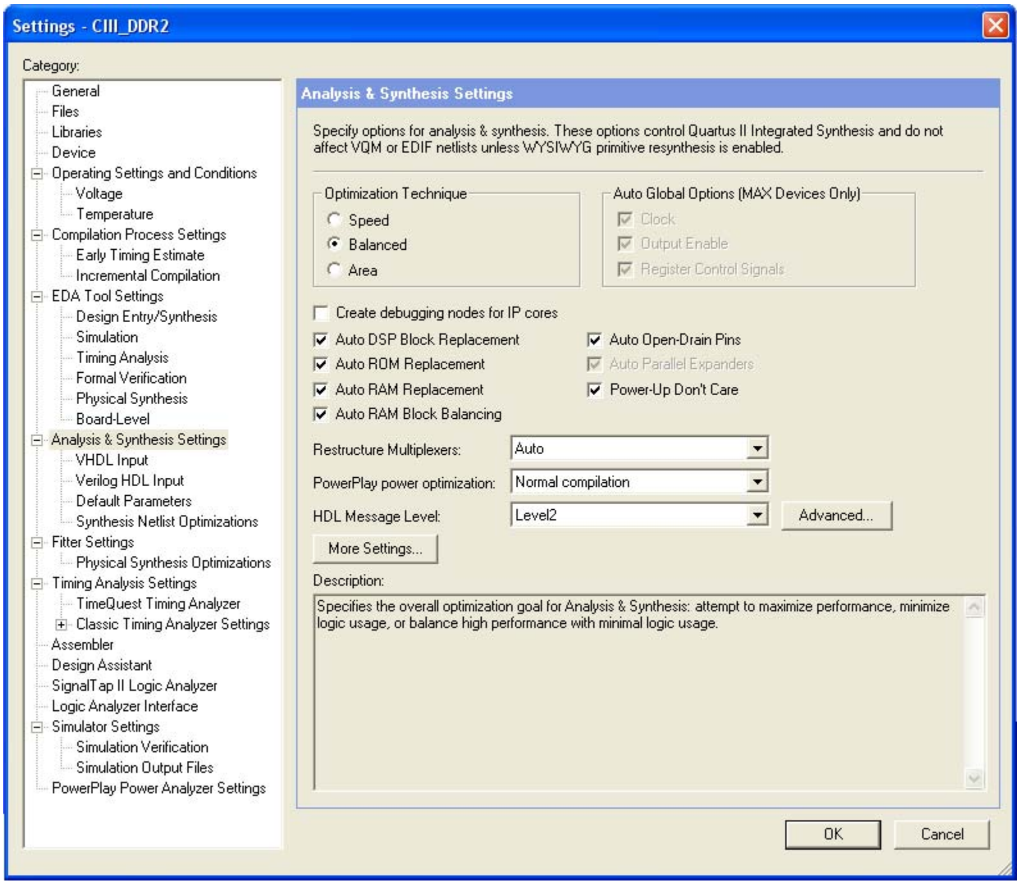
After setting the design example as the top level entity, set the Quartus II software to ensure the remaining unconstrained paths are routed with the highest speed and efficiency. To do this, perform the following steps:

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



Specify options for analysis & synthesis. These options control Quartus II Integrated Synthesis and do not affect VQM or EDIF netlists unless WYSIWYG primitive resynthesis is enabled.

Optimization Technique: Speed [✓ Balanced] Area

Auto Global Options (MAX Devices Only): ✓ Clock, ✓ Output Enable, ✓ Register Control Signals

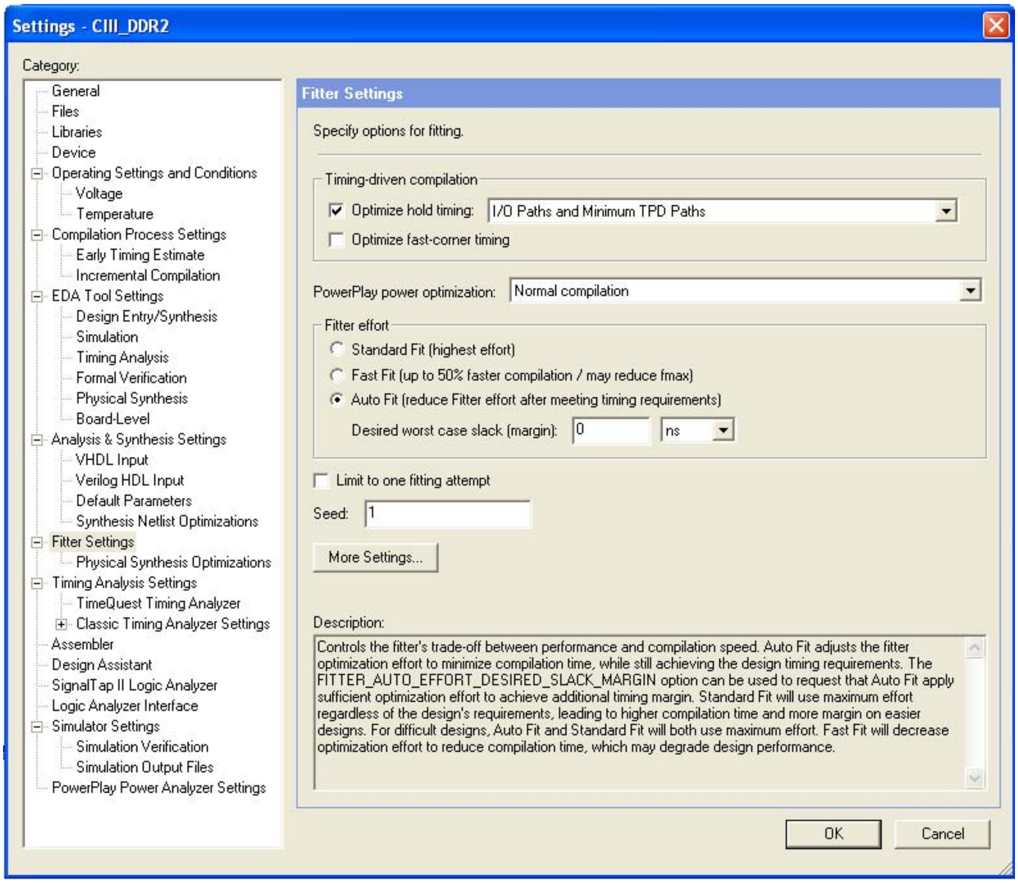
☐ Create debugging nodes for IP cores
✓ Auto DSP Block Replacement
✓ Auto ROM Replacement
✓ Auto RAM Replacement
✓ Auto RAM Block Balancing
✓ Auto Open-Drain Pins
✓ Auto Parallel Expanders
✓ Power-Up Don't Care

Restructure Multiplexers: Auto
PowerPlay power optimization: Normal compilation
HDL Message Level: Level2

Description:
Specifies the overall optimization goal for Analysis & Synthesis: attempt to maximize performance, minimize logic usage, or balance high performance with minimal logic usage.

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.

Figure 16. Default Setting for Fitter Effort



Specify options for fitting.

Timing-driven compilation

[✓] Optimize hold timing: I/O Paths and Minimum TPD Paths

[ ] Optimize fast-corner timing

PowerPlay power optimization: Normal compilation

Fitter effort

○ Standard Fit (highest effort)

○ Fast Fit (up to 50% faster compilation / may reduce fmax)

● Auto Fit (reduce Fitter effort after meeting timing requirements)

Desired worst case slack (margin): 0 ns

[ ] Limit to one fitting attempt

Seed: 1

Description:

Controls the fitter's trade-off between performance and compilation speed. Auto Fit adjusts the fitter optimization effort to minimize compilation time, while still achieving the design timing requirements. The FITTER_AUTO_EFFORT_DESIRED_SLACK_MARGIN option can be used to request that Auto Fit apply sufficient optimization effort to achieve additional timing margin. Standard Fit will use maximum effort regardless of the design's requirements, leading to higher compilation time and more margin on easier designs. For difficult designs, Auto Fit and Standard Fit will both use maximum effort. Fast Fit will decrease optimization effort to reduce compilation time, which may degrade design performance.

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.

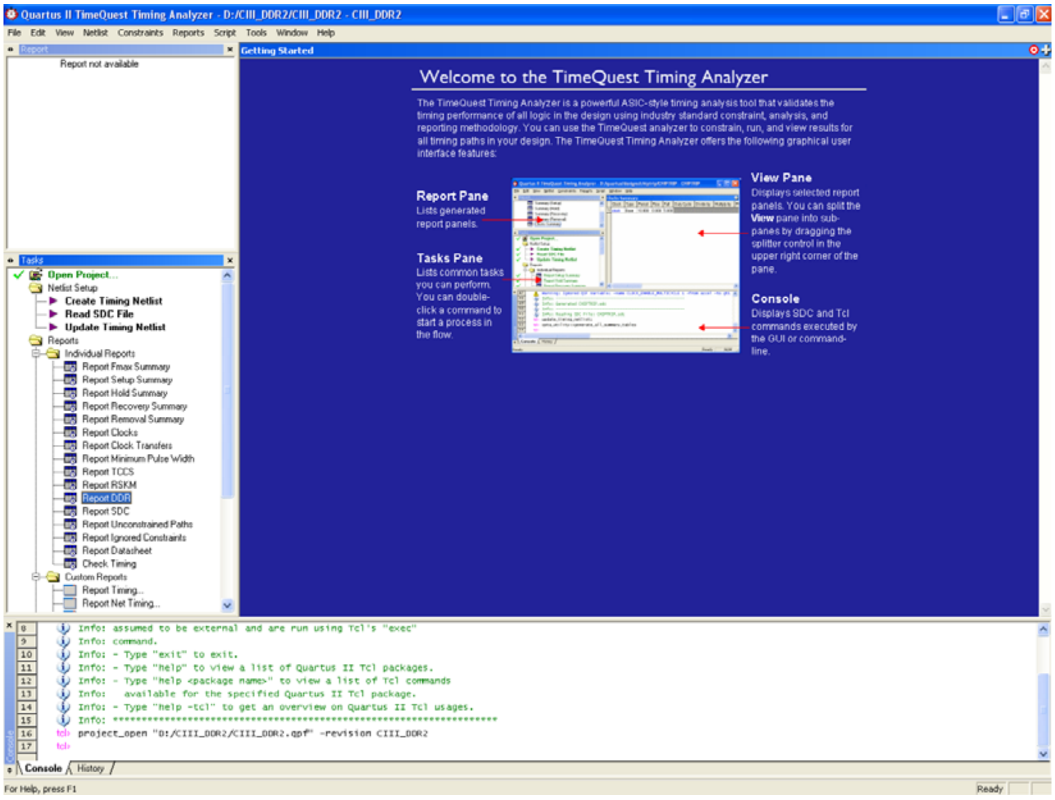
2. After compilation is successful, on the Tools menu, select **TimeQuest Timing Analyzer**.

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*.

Figure 17. TimeQuest Timing Analyzer Window



Welcome to the TimeQuest Timing Analyzer

The TimeQuest Timing Analyzer is a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in the design using industry standard constraint, analysis, and reporting methodology. You can use the TimeQuest analyzer to constrain, run, and view results for all timing paths in your design. The TimeQuest Timing Analyzer offers the following graphical user interface features:

Report Pane
Lists generated report panels.

Tasks Pane
Lists common tasks you can perform. You can double-click a command to start a process in the flow.

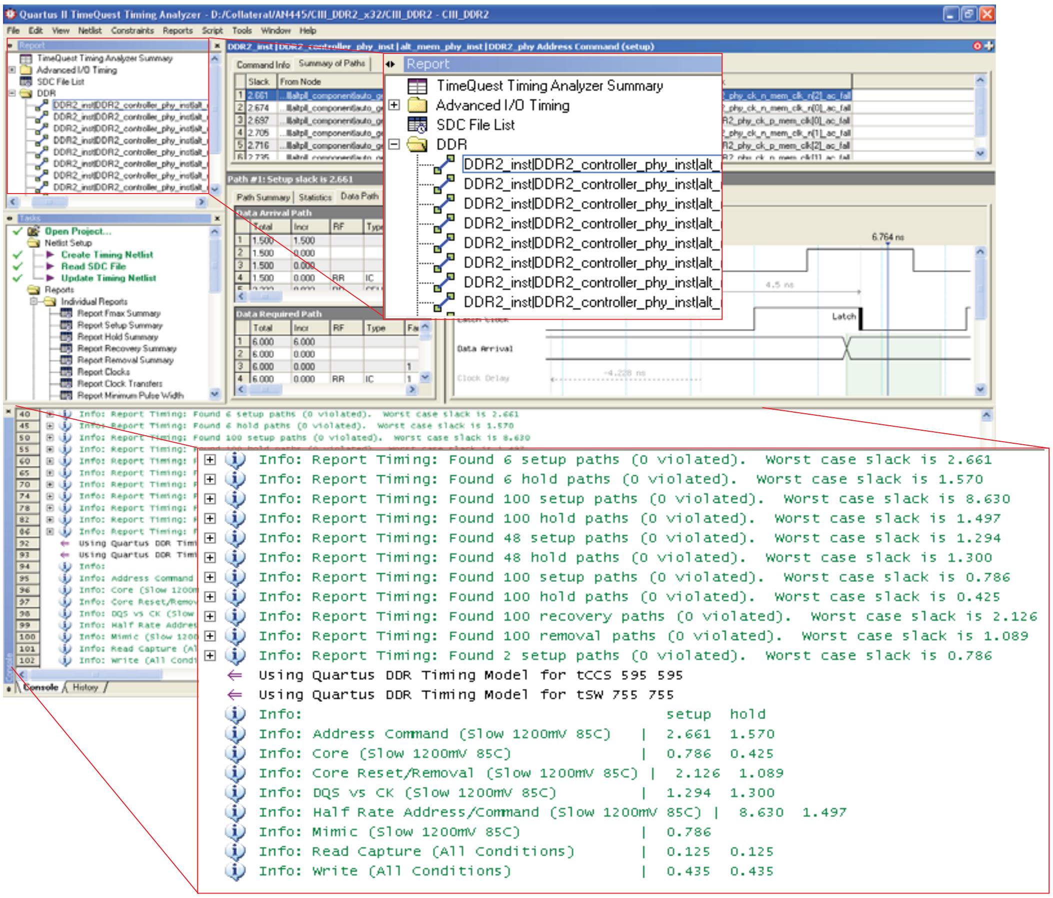
View Pane
Displays selected report panels. You can split the **View** pane into sub-panes by dragging the splitter control in the upper right corner of the pane.

Console
Displays SDC and Tcl commands executed by the GUI or command-line.

```
Info: assumed to be external and are run using Tcl's "exec"
Info: command.
Info: - Type "exit" to exit.
Info: - Type "help" to view a list of Quartus II Tcl packages.
Info: - Type "help <package name>" to view a list of Tcl commands
Info:   available for the specified Quartus II Tcl package.
Info: - Type "help -tcl" to get an overview on Quartus II Tcl usages.
Info: ****************************************************************
tcl> project_open "D:/CIII_DDR2/CIII_DDR2.qpf" -revision CIII_DDR2
tcl>
```

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. DDR2 Report Panel and Timing Margin in TimeQuest Timing Analyzer


Quartus II TimeQuest Timing Analyzer - D:/Collateral/AN445/CIII_DDR2_x32/CIII_DDR2 - CIII_DDR2
File Edit View Netlist Constraints Reports Script Tools Window Help
Report
TimeQuest Timing Analyzer Summary
Advanced I/O Timing
SDC File List
DDR
DDR2_inst|DDR2_controller_phy_inst|alt_
DDR2_inst|DDR2_controller_phy_inst|alt_
DDR2_inst|DDR2_controller_phy_inst|alt_
DDR2_inst|DDR2_controller_phy_inst|alt_
DDR2_inst|DDR2_controller_phy_inst|alt_
DDR2_inst|DDR2_controller_phy_inst|alt_
DDR2_inst|DDR2_controller_phy_inst|alt_
DDR2_inst|DDR2_controller_phy_inst|alt_
Tasks
Open Project...
Netlist Setup
Create Timing Netlist
Read SDC File
Update Timing Netlist
Reports
Individual Reports
Report Fmax Summary
Report Setup Summary
Report Hold Summary
Report Recovery Summary
Report Removal Summary
Report Clocks
Report Clock Transfers
Report Minimum Pulse Width
Path #1: Setup slack is 2.661
Data Arrival Path
Data Required Path
6.764 ns
4.5 ns
Latch
Data Arrival
Clock Delay
Console
History

```
Info: Report Timing: Found 6 setup paths (0 violated).  Worst case slack is 2.661
Info: Report Timing: Found 6 hold paths (0 violated).  Worst case slack is 1.570
Info: Report Timing: Found 100 setup paths (0 violated).  Worst case slack is 8.630
Info: Report Timing: Found 100 hold paths (0 violated).  Worst case slack is 1.497
Info: Report Timing: Found 48 setup paths (0 violated).  Worst case slack is 1.294
Info: Report Timing: Found 48 hold paths (0 violated).  Worst case slack is 1.300
Info: Report Timing: Found 100 setup paths (0 violated).  Worst case slack is 0.786
Info: Report Timing: Found 100 hold paths (0 violated).  Worst case slack is 0.425
Info: Report Timing: Found 100 recovery paths (0 violated).  Worst case slack is 2.126
Info: Report Timing: Found 100 removal paths (0 violated).  Worst case slack is 1.089
Info: Report Timing: Found 2 setup paths (0 violated).  Worst case slack is 0.786
Using Quartus DDR Timing Model for tCCS 595 595
Using Quartus DDR Timing Model for tSW 755 755
Info:                                           setup  hold
Info: Address Command (Slow 1200mV 85C)       | 2.661  1.570
Info: Core (Slow 1200mV 85C)                  | 0.786  0.425
Info: Core Reset/Removal (Slow 1200mV 85C) |  2.126  1.089
Info: DQS vs CK (Slow 1200mV 85C)             | 1.294  1.300
Info: Half Rate Address/Command (Slow 1200mV 85C) |  8.630  1.497
Info: Mimic (Slow 1200mV 85C)                 | 0.786
Info: Read Capture (All Conditions)           | 0.125  0.125
Info: Write (All Conditions)                  | 0.435  0.435
```

The report timing script provides information about the following margins and paths:

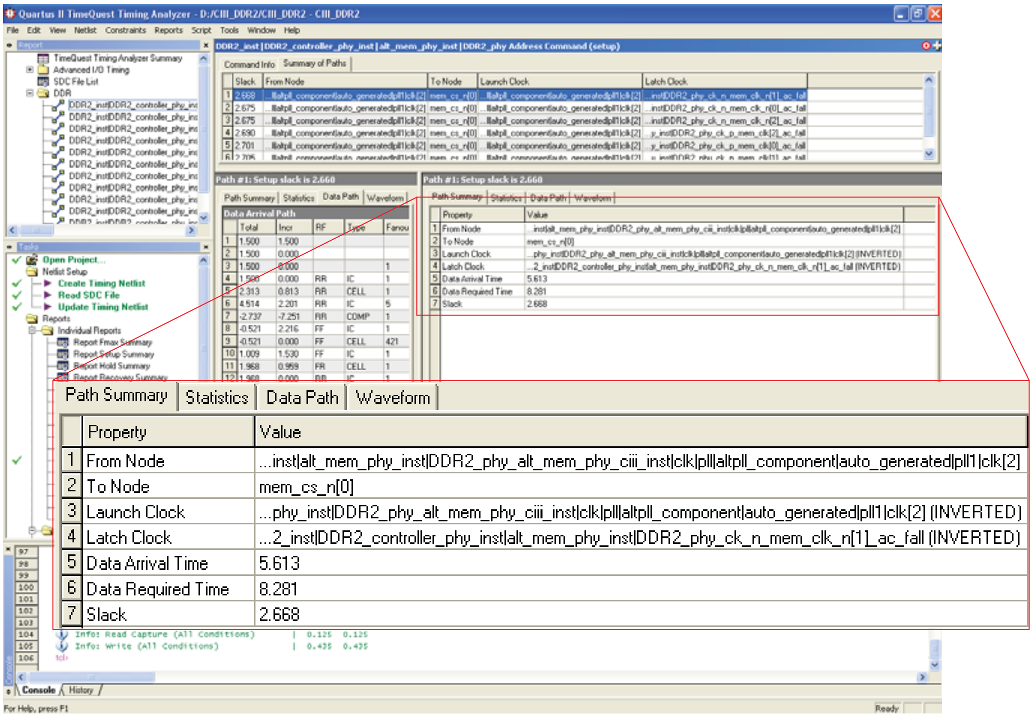
- Address/command setup and hold margin
- Half-rate address/command setup and hold margin
- Core setup and hold margin
- Core reset/removal setup and hold margin
- DQS vs CK setup and hold margin
- Mimic path
- Write setup and hold margin
- Read capture setup and hold margin

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



DDR2_inst | DDR2_controller_phy_inst | alt_mem_phy_inst | DDR2_phy Address Command (setup)

	Slack	From Node	To Node	Launch Clock	Latch Clock
1	2.668	...llaltpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...llaltpll_component|auto_generated|pll1|clk[2]	...inst|DDR2_phy_ck_n_mem_clk_n[1]_ac_fall
2	2.675	...llaltpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...llaltpll_component|auto_generated|pll1|clk[2]	...inst|DDR2_phy_ck_n_mem_clk_n[0]_ac_fall
3	2.675	...llaltpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...llaltpll_component|auto_generated|pll1|clk[2]	...inst|DDR2_phy_ck_n_mem_clk_n[2]_ac_fall
4	2.690	...llaltpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...llaltpll_component|auto_generated|pll1|clk[2]	...y_inst|DDR2_phy_ck_p_mem_clk[2]_ac_fall
5	2.701	...llaltpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...llaltpll_component|auto_generated|pll1|clk[2]	...y_inst|DDR2_phy_ck_p_mem_clk[0]_ac_fall

Path #1: Setup slack is 2.660

Data Arrival Path

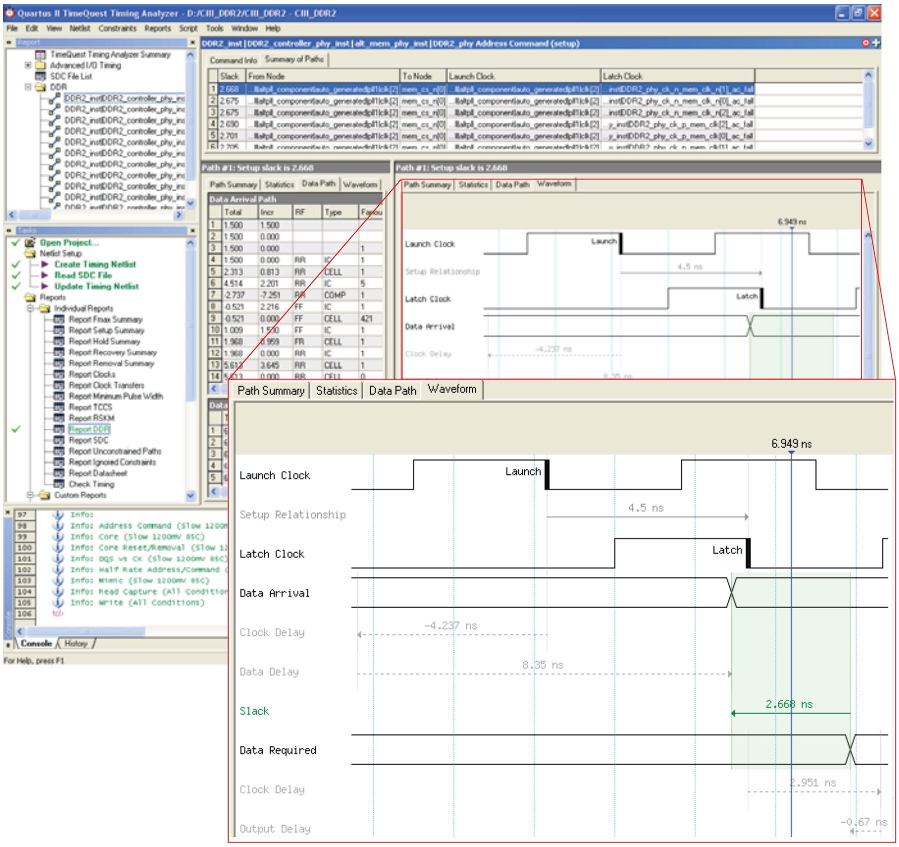
	Total	Incr	RF	Type	Fanou
1	1.500	1.500			
2	1.500	0.000			
3	1.500	0.000			1
4	1.500	0.000	RR	IC	1
5	2.313	0.813	RR	CELL	1
6	4.514	2.201	RR	IC	5
7	-2.737	-7.251	RR	COMP	1
8	-0.521	2.216	FF	IC	1
9	-0.521	0.000	FF	CELL	421
10	1.009	1.530	FF	IC	1
11	1.968	0.959	FR	CELL	1

Path #1: Setup slack is 2.660

	Property	Value
1	From Node	...inst|alt_mem_phy_inst|DDR2_phy_alt_mem_phy_ciii_inst|clk|pll|altpll_component|auto_generated|pll1|clk[2]
2	To Node	mem_cs_n[0]
3	Launch Clock	...phy_inst|DDR2_phy_alt_mem_phy_ciii_inst|clk|pll|altpll_component|auto_generated|pll1|clk[2] (INVERTED)
4	Latch Clock	...2_inst|DDR2_controller_phy_inst|alt_mem_phy_inst|DDR2_phy_ck_n_mem_clk_n[1]_ac_fall (INVERTED)
5	Data Arrival Time	5.613
6	Data Required Time	8.281
7	Slack	2.668

```
Info: Read Capture (All Conditions)    |  0.125  0.125
Info: Write (All Conditions)           |  0.435  0.435
```

Figure 20. Waveform View for the Address and Command Datapath



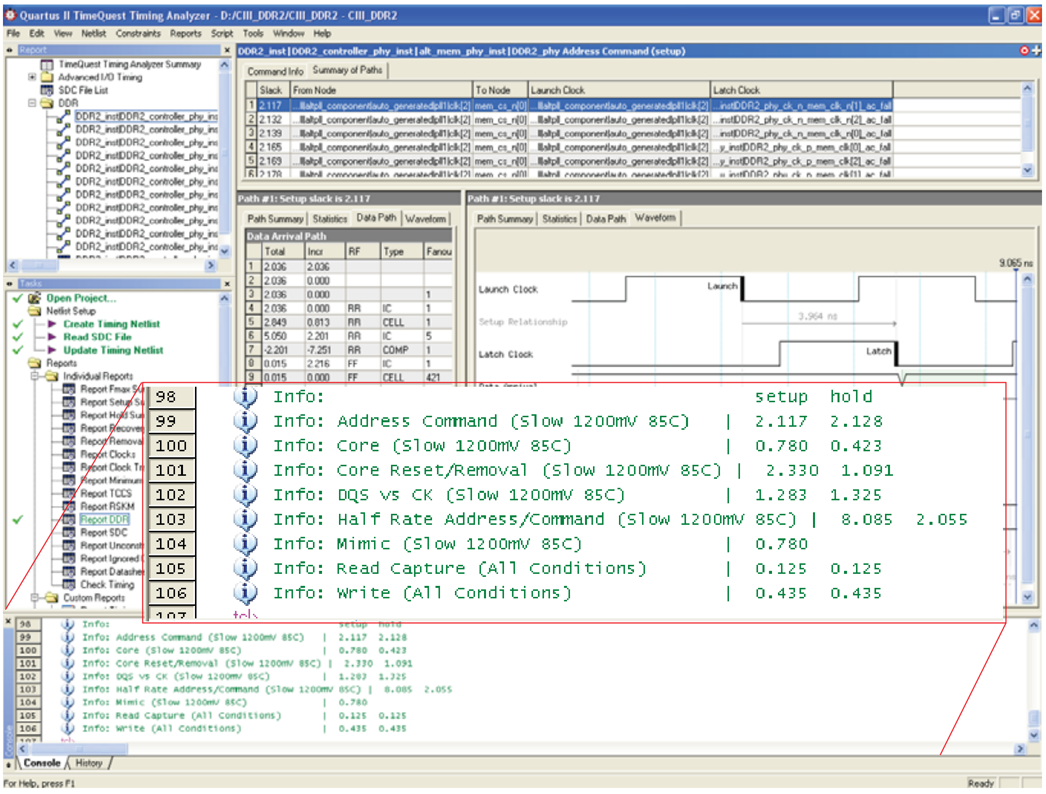
	Slack	From Node	To Node	Launch Clock	Latch Clock
1	2.668	...|altpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...|altpll_component|auto_generated|pll1|clk[2]	...inst|DDR2_phy_ck_n_mem_clk_n[1]_ac_fall
2	2.675	...|altpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...|altpll_component|auto_generated|pll1|clk[2]	...inst|DDR2_phy_ck_n_mem_clk_n[0]_ac_fall
3	2.675	...|altpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...|altpll_component|auto_generated|pll1|clk[2]	...inst|DDR2_phy_ck_n_mem_clk_n[2]_ac_fall
4	2.690	...|altpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...|altpll_component|auto_generated|pll1|clk[2]	...p_inst|DDR2_phy_ck_p_mem_clk[2]_ac_fall
5	2.701	...|altpll_component|auto_generated|pll1|clk[2]	mem_cs_n[0]	...|altpll_component|auto_generated|pll1|clk[2]	...p_inst|DDR2_phy_ck_p_mem_clk[0]_ac_fall
6	2.705	[illegible]	[illegible]	[illegible]	[illegible]

	Total	Incr	RF	Type	Fanout
1	1.500	1.500			
2	1.500	0.000			
3	1.500	0.000			1
4	1.500	0.000	RR	IC	1
5	2.313	0.813	RR	CELL	1
6	4.514	2.201	RR	IC	5
7	-2.737	-7.251	RR	COMP	1
8	-0.521	2.216	FF	IC	1
9	-0.521	0.000	FF	CELL	421
10	1.009	1.530	FF	IC	1
11	1.968	0.959	FR	CELL	1
12	1.968	0.000	RR	IC	1
13	5.613	3.645	RR	CELL	1
14	5.613	0.000	RR	CELL	0

The report indicates that `clk2` of the PLL is clocking the address and command registers. Go to the PLL megafunction and change the phase setting of `clk2`. For this design, the initial phase setting of `clk2` is set to –90° with reference to the system clock. By adjusting `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 `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 `clk2`. The address and command datapath now has a more balanced 2.117 ns setup and 2.128 ns hold margin.

Figure 21. Timing Margin Reported After Adjusting clk2



```
Info:                                              setup  hold
Info: Address Command (Slow 1200mV 85C)       |  2.117  2.128
Info: Core (Slow 1200mV 85C)                  |  0.780  0.423
Info: Core Reset/Removal (Slow 1200mV 85C) |   2.330  1.091
Info: DQS vs CK (Slow 1200mV 85C)             |  1.283  1.325
Info: Half Rate Address/Command (Slow 1200mV 85C) |  8.085  2.055
Info: Mimic (Slow 1200mV 85C)                 |  0.780
Info: Read Capture (All Conditions)           |  0.125  0.125
Info: Write (All Conditions)                  |  0.435  0.435
```

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 Ω or 50 Ω, 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 Ω, 75 Ω, and 50 Ω. The 50-Ω 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

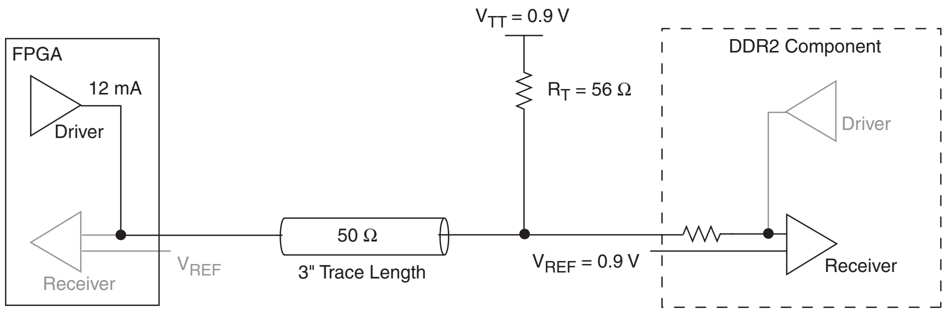
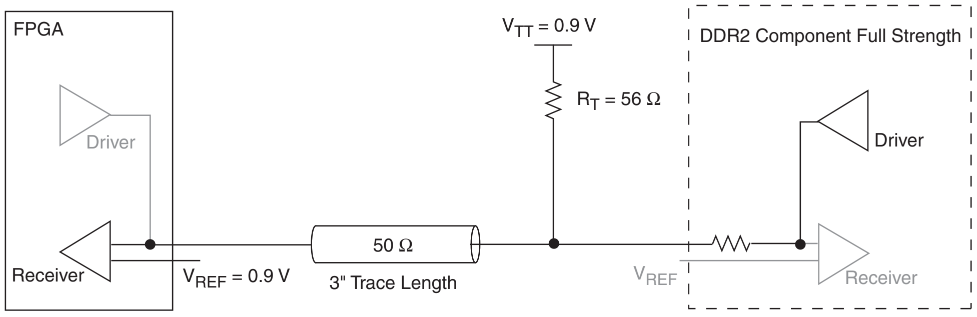


Figure 23 shows the setup for read operation from the DDR2 SDRAM memory.

Figure 23. Read Operation from DDR2 SDRAM Memory with Class I Termination



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

- *JEDEC Standard Publication JESD79C, DDR AND DDR2 SDRAM Specification*, JEDEC Solid State Technology Association.
- *MT47H32M16 DDR2 SDRAM Data Sheet*, Micron Technology, Inc.

Referenced Documents

This application note references the following documents:

- *ALTMEMPHY Megafunction 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*.

Document Revision History

Table 5 shows the revision history for this document.

Table 5. Document Revision History

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

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.

Project Name:
Date:

Select Device

	Done	N/A	
1	☐	☐	Have you selected the memory interface frequency of operation and bus width? And, have you selected the FPGA device density and package combination that you will be targeting? Ensure that the target FPGA device supports the desired clock rate and memory bus width. For detailed device resource information, refer to the *External Memory Interfaces in Cyclone III Devices* chapter in volume 1 of the *Cyclone III Device Handbook*.

Instantiate PHY and Controller

	Done	N/A	
2	☐	☐	Have you parameterized and instantiated the ALTMEMPHY megafunction for your target memory interface? When instantiating multiple instances of the ALTMEMPHY megafunction, ensure effective sharing of device resources and appropriate constraints by referring to *AN 462: Implementing Multiple Memory Interfaces Using the ALTMEMPHY Megafunction*.
3	☐	☐	Have you connected the PHY's local signals to your driver logic and the PHY's memory interface signals to top-level pins? Ensure that the local interface signals of the PHY are appropriately connected to your own logic. If the ALTMEMPHY megafunction is compiled without these local interface connections, you may encounter compilation problems when the number of signals exceeds the pins available on your target device.

Functional Simulation

	Done	N/A	
4	☐	☐	Have you simulated your design using the RTL functional model? Use the ALTMEMPHY megafunction 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

	Done	N/A	
5	☐	☐	Have you added constraints to the PHY and the rest of your design? The ALTMEMPHY megafunction 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

	Done	N/A	
8	☐	☐	Have you performed a timing simulation on your design?

Board-Level Considerations

	Done	N/A	
9	☐	☐	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).

10	☐	☐	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

	Done	N/A	
11	☐	☐	Have you verified functionality of the memory interface in the system?

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support/
Literature Services:
literature@altera.com

I.S. EN ISO 9001