AN 461: Design Guidelines for Implementing QDRII+ and QDRII SRAM Interfaces in Stratix III and Stratix IV Devices

February 2010, v1.2

QDRII+ and the QDRII SRAM devices are ideally suited for bandwidth– intensive and low-latency applications such as controller buffer memory, look-up tables (LUTs), and linked lists. QDRII+ and QDRII SRAM memory architecture features separate read and write ports operating twice per clock cycle to deliver a total of four data transfers per cycle.

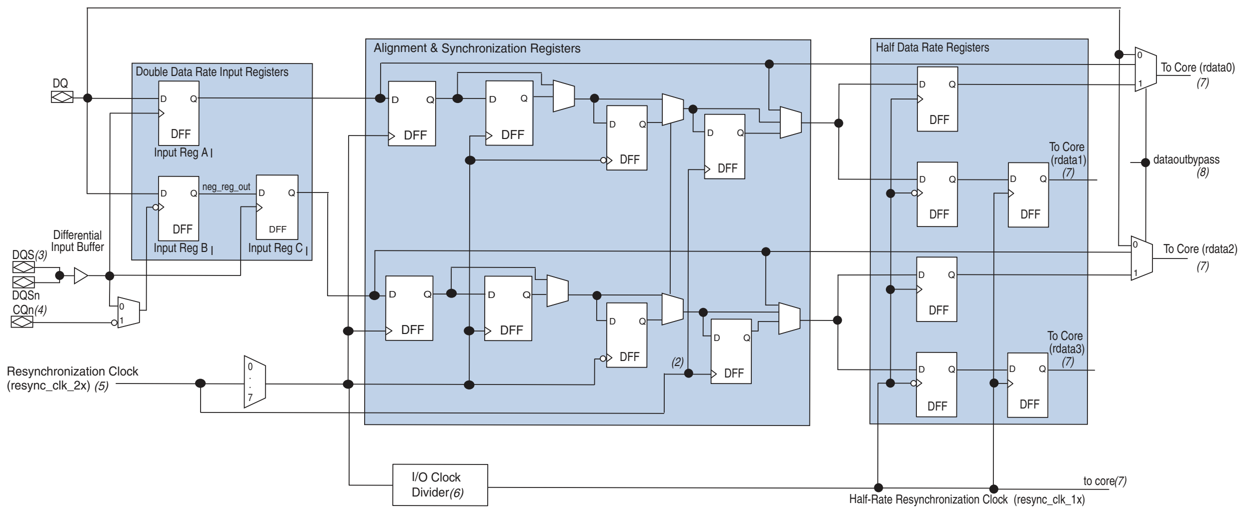
Stratix® III and Stratix IV I/Os are specifically designed to support double-data rate (DDR) external memory standards such as the QDRII+ and QDRII SRAM. Combined with the new self-calibrating physical interface, the ALTMEMPHY megafunction, Stratix III and Stratix IV devices deliver performance of up to 400 MHz or 1.6 Gbps on top and bottom I/O banks.

For information about performance specifications, refer to the *System Performance Specification* section of the *External Memory Interface Handbook*.

CAUTION The IP described in this document is scheduled for product obsolence and discontinued support. Therefore, Altera® does not recommend use of this IP in new designs. For more information about Altera's current IP offering, refer to Altera's Intellectual Property website.

Figure 1 and Figure 2 show the Stratix III and the Stratix IV input, output, and output-enable register paths for the QDRII+ and QDRII SRAM interface.

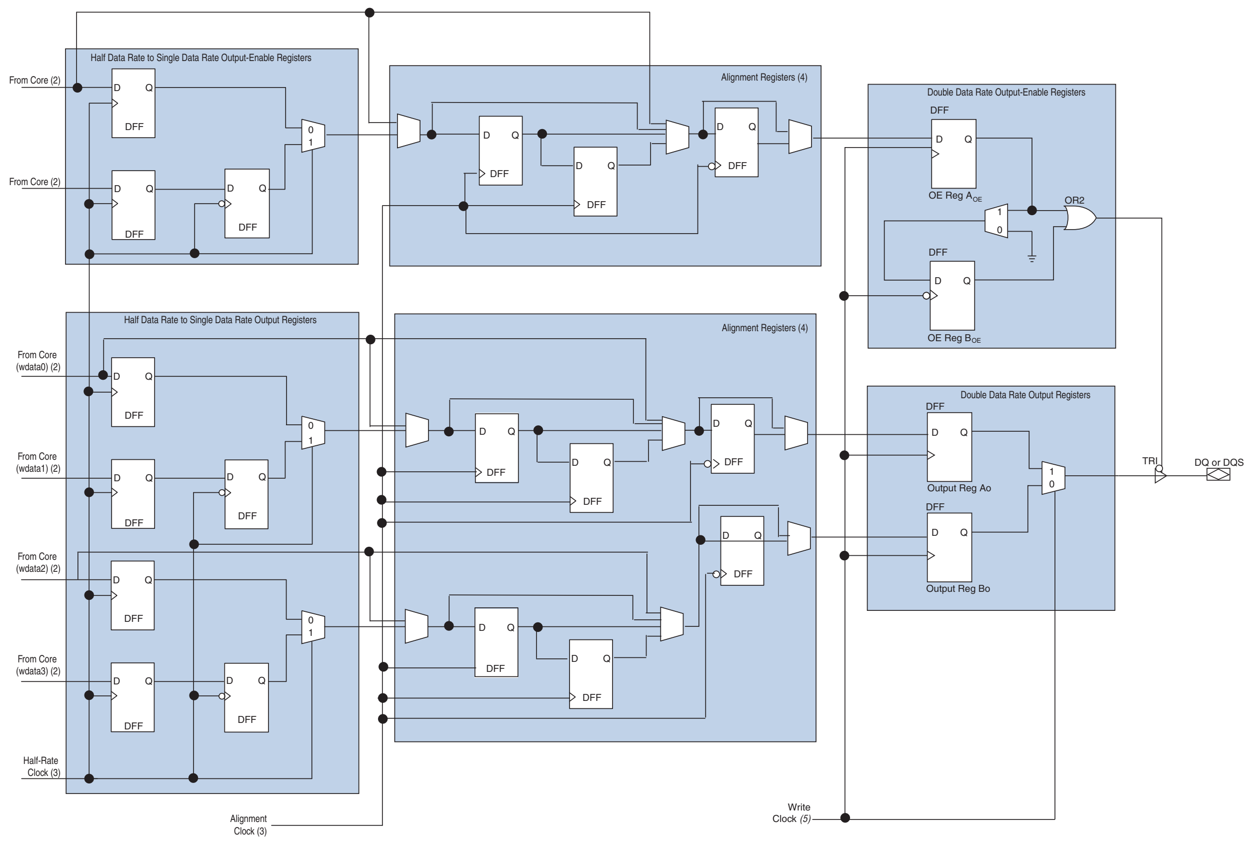
Figure 1. Stratix III and Stratix IV IOE Input Registers for QDRII+/QDRII SRAM Interface *(Note 1)*



Notes to Figure 1:

(1) You can bypass each register block in this path.
(2) This is the zero phase resynchronization clock (from read-leveling delay chain).
(3) The input clock comes either from the DQS logic block (whether the postamble circuitry is bypassed or not) or from a global clock line.
(4) This input clock comes from the CQn logic block.
(5) This resynchronization clock comes either from the PLL or from the read-leveling delay chain.
(6) The I/O clock divider resides adjacent to the DQS logic block. In addition to the PLL and read-leveled resynchronization clock, the I/O clock divider can also be fed by the DQS bus or CQn bus.
(7) The half-rate data and clock signals feed into a dual-port RAM in the FPGA core.
(8) You can dynamically change the `dataoutbypass` signal after configuration.

Figure 2. Stratix III and Stratix IV IOE Output and Output-Enable Path Registers for QDRII+/QDRII SRAM Interface *(Note 1)*



Notes to Figure 2:

(1) You can bypass each register block of the output and out-enable paths.

(2) Data coming from the FPGA core are at half the frequency of the memory interface.

(3) Half-rate and alignment clocks come from the PLL.

(4) These registers are only used in DDR3 SDRAM interfaces.

(5) The write clock comes either from the PLL or from the write-leveling delay chain. The DQ write clock and DQS write clock have a 90° offset between them.

This application note describes Altera's recommended design flow to implement a QDRII+ or a QDRII SRAM memory interface on a Stratix III device. You can implement the same design flow on a Stratix IV device as well. An example of a design walk-through that details the critical aspects of this flow is provided, including:

- Instantiating the physical interface (PHY) to a QDRII or QDRII+ or a QDRII SRAM device.
- Setting the appropriate constraints on the PHY.
- Verifying the design functionality using simulation.
- Timing closure using the TimeQuest timing analyzer in the Quartus® II software.

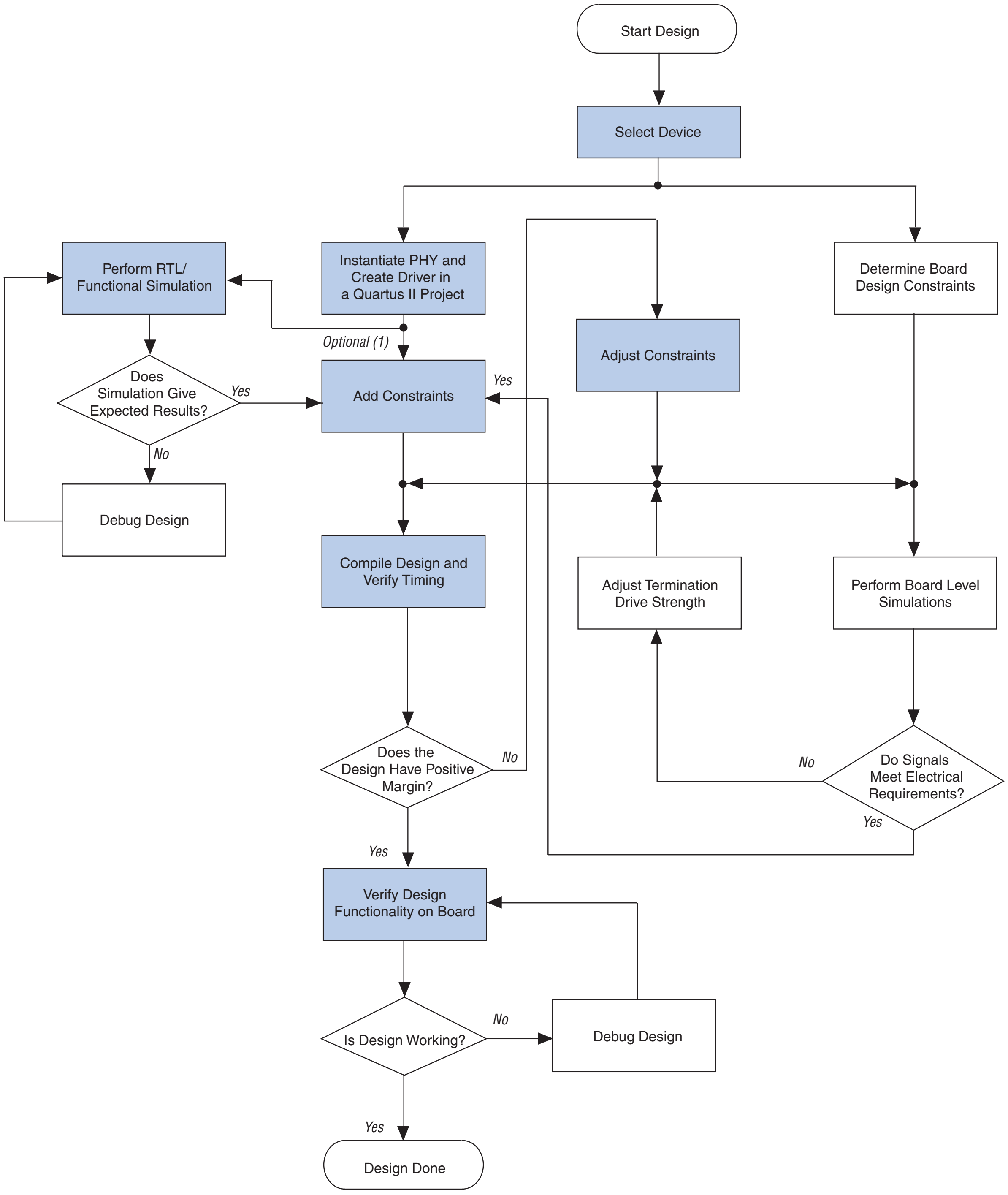
Use this application note along with the following literature:

- *External Memory Interfaces in the Stratix III Devices* chapter of the *Stratix III Device Handbook*.
- *External Memory Interfaces in the Stratix IV Devices* chapter of the *Stratix IV Device Handbook*.
- *Timing Analysis* section of the *External Memory Interface Handbook*.

Memory Interface Design Flow

Altera recommends the design flow described in this section as the best practices for successful memory interface implementation in Stratix III and the Stratix IV devices. A detailed discussion of each step shown in Figure 3 appears in this section. The next section includes a design example that discusses the shaded design flow steps (Figure 3) in detail.

Figure 3. Memory Interface Design Flow Chart for Implementing Stratix III and Stratix IV Devices


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

Note to Figure 3:

(1) This step is optional, but recommended.

Step 1: Select a Device

The first step in memory interface design is to determine the bandwidth required by your system. Bandwidth can be expressed as:

Bandwidth = Data width (bits) × data transfer rate (1 per second) × efficiency

In quadruple data rate memories like the QDRII+ or the QDRII SRAM, the data transfer rate is four times the clock rate. Hence, the data transfer rate for a 400 MHz interface is 1600 Mbps. The efficiency is the percentage of time the data bus is transferring data. It is dependent on the type of memory. For example, in a QDRII+ or a QDRII SRAM memory interface with separate write and read ports, the efficiency would be 100% when there is an equal number of read and write operations on these memory interfaces.

After calculating the bandwidth of the memory interface, determine which memory and FPGA device to use. For information about your memory device, refer to the *Memory Standard Overviews* section of the *External Memory Interface Handbook*.

Altera's FPGA devices support the various data widths for different memory interfaces. Stratix III and Stratix IV devices offer dedicated DQS circuitry for DQ group widths ranging from ×4 to ×36. In the ×36 mode, there is one DQS/CQn pin pair that can clock 36 DQ pins using a dedicated peripheral clock network. When interfacing with QDRII+ or the QDRII SRAM devices with a 36-bit read and write busses, connect the memory's CQ/CQn pins to the Stratix III or the Stratix IV device's DQS/CQn pins and connect the Q pins to the DQ pins. The number and width of DQ groups supported on each density and package combination differs, so you must determine the FPGA device density and package combination best suits your needs.

Support for 36-bit wide DQ groups is only available in Stratix III and Stratix IV devices in 1517-pin and 1760-pin packages. Stratix III and Stratix IV devices in 780-pin and 1152-pin packages also support ×36 mode emulation by using ×16/×18 DQS/DQ groups.

The maximum number of interfaces that you can implement on any given device is limited by the resource availability (number of DQ groups of desired width, number of phase-locked loops (PLLs), number of delay-locked loops (DLLs), and clock resources). For example, the EP3SL150F1152 device supports four ×18 groups on each side of the device. This allows you to interface with a total of 16 QDRII+ or QDRII SRAM devices. Also, each instance of the ALTMEMPHY megafunction for the QDRII+ and the QDRII SRAM requires one PLL, one DLL, and four clock resources. The EP3SL150F1152 supports a total of eight PLLs, four DLLs, and 80 clock networks (16 global and 64 regional clocks). The four DLLs enable support for four unique memory interface clock frequencies and each DLL is able to drive two adjacent sides of the device. The final consideration is to check the availability of the core resources (the adaptive logic modules (ALMs) and TriMatrix memory blocks), and user I/O pins for other memory interface signals including the write data, the address, the command, and memory clock signals.

For more information on the ×36 QDRII + QDRII/SRAM interface support in the F780 and F1152-pin packages, refer to the *Device and Pin Planning* section of the *External Memory Interface Handbook*. For information on the Stratix III and the Stratix IV FPGA density and package support for the different memory types, refer to the *External Memory Interfaces in Stratix III Devices* chapter of the *Stratix III Device Handbook* and the *External Memory Interfaces in Stratix IV Devices* chapter of the *Stratix IV Device Handbook*.

Step 2: Create a Project in the Quartus II Software

After selecting the appropriate Stratix III or Stratix IV FPGA device, create a project in the Quartus II software that targets the chosen device.

For step-by-step instructions on creating a Quartus II software project, refer to the Tutorial in the Quartus II Online Help.

Step 3: Instantiate PHY

For all the Stratix III and the Stratix IV memory interface designs, use the ALTMEMPHY megafunction when instantiating the physical interface to the memory device. This megafunction features a license-free physical interface that should be used with your memory interface driver logic. Since SRAM devices do not need a memory controller, Altera does not offer such a product. You should connect the local signals of the memory interface to logic in your top-level design that will access the memory.

For detailed information about the architecture, ports, parameters, and the resource usage information of the PHY, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

When implementing wider interfaces using multiple memory chips, you can configure a single ALTMEMPHY megafunction instance to support independent read and write signals for each memory device, the shared address, and the command signals to all memory devices. When implementing multiple independent interfaces (interfaces operating at different clock rates or interfaces with independent address and command buses), you should create unique ALTMEMPHY megafunction instances for each of the interface.

For details on sharing device resources (PLLs, DLLs, and clock networks) across multiple instances of the PHY, refer to the *Implementing Multiple ALTMEMPHY-Based Controllers* chapter in the *DDR, DDR2, and DDR3 SDRAM Design Tutorials* section of the *External Memory Interface Handbook*.

For the QDRII+ SRAM interface, the ALTMEMPHY megafunction supports devices with 2.5 clock cycle latency directly. QDRII+ SRAM devices with 2.0 clock cycle latency are not supported by the ALTMEMPHY megafunction.

Step 4: Perform RTL/Functional Simulation (Optional)

Simulating the design to verify functionality is optional but is recommended to ensure the interface performs as intended in your system. When you instantiate the ALTMEMPHY megafunction using the MegaWizard™ Plug-In Manager, the software provides you with an option to generate a simulation model of the PHY in either the Verilog HDL or the VHDL. This PHY functional simulation model is a cycle-accurate HDL model file produced by the Quartus II software. These models work with the Altera-supported VHDL and Verilog HDL simulators. You also need to get the memory's simulation model file from the memory vendor in order to perform the simulation.

Step 5: Add Constraints

The next step in the design process is to add the appropriate device constraints related to the external memory interface. These constraints include the timing, the I/O standards, the output-enable group and DQ/DQS group assignments. The Quartus II software MegaWizard Plug-In Manger generates the **.sdc/.tcl** files containing these constraints for each ALTMEMPHY megafunction instance.

The ALTMEMPHY megafunction requires the use of the TimeQuest timing analyzer and is fully timing-constrained using synopsys design constraints (SDC) assignments. The SDC timing constraints are derived from the parameters you entered for the ALTMEMPHY megafunction, based on the QDRII+ or QDRII SRAM data sheet and tolerances from the board layout. The ALTMEMPHY megafunction uses the TimeQuest timing constraints along with the Quartus II timing–driven fitter to achieve timing closure.

After instantiating the ALTMEMPHY megafunction, the ALTMEMPHY MegaWizard interface generates the following files in Table 1 that you need to apply in order to properly constrain your design.

Table 1. Constraint Files Created by the ALTMEMPHY MegaWizard Interface

File Name	Description
<variation_name>**_ddr_pins.sdc**	Contains procedures used in the *<variation_name>***_ddr_timing.sdc** and *<variation_name>***_report_timing.tcl** files.
<variation_name>**_report_timing.tcl**	Script that reports timing for your ALTMEMPHY variation during compilation.

For detailed information on creating, generating, and setting the constraints for the PHY, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

You must add the appropriate pin location and pin loading assignments to the memory interface signals in your design, in addition to timing and device constraints for the rest of your design.

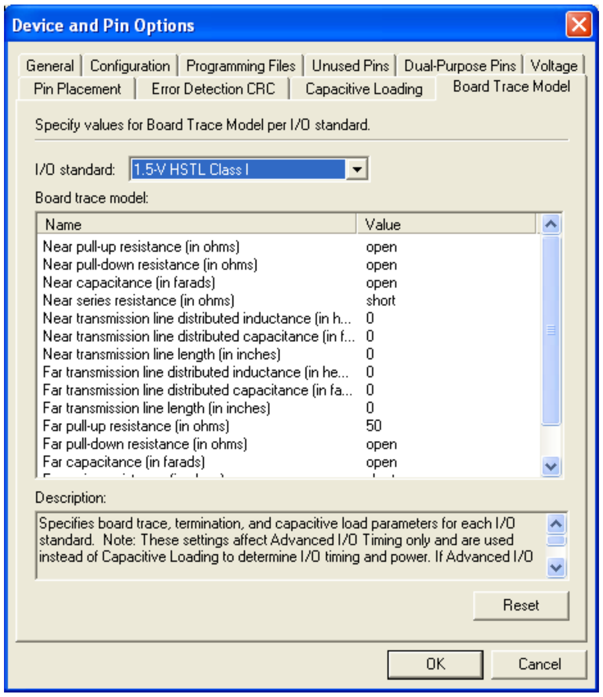
Stratix III devices provide an independent DQS logic block for each DQS and CQn pin for complementary read data-strobe/clock operations. In the Stratix III pin tables, the differential DQS pin-pairs are denoted as DQS pins and DQSn pins, while the complementary DQS signals are denoted as DQS pins and CQn pins. DQSn pins and CQn pins are marked in the pin table separately. Each CQn pin connects to a DQS logic block and the shifted CQn signals go to the negative-edged input registers in the IOE registers.

The Quartus II software has the Advanced I/O Timing feature that allows the timing analysis to take the board setup into consideration, specifically for bidirectional or output pins, for more accurate timing analysis results. This feature is turned on for Stratix III and Stratix IV devices by default. When using the Advanced I/O Timing feature, you should specify the board setup information such as the capacitance and inductance of the board traces, the pull-up or pull-down resistor values, at both far-end, and near-end of the transmission line, as well as the termination voltage.

You can either specify the information in the Board Trace Model setup through the **Device and Pin Options** dialog box in the Quartus II software for all the output or bidirectional pins (Figure 4), or separately for the each pin or net group, such as the address bus, through the Quartus II Pin Planner (Figure 5). The board traces information from the Quartus II Pin Planner takes precedence over the information from the **Device and Pin Options** settings.

Include only the board information. You do not need to include the FPGA device's pin or package capacitance, on-chip termination or drive strength setting. Typically, the far-end capacitance equals the input capacitance of the far-end device being interfaced to.

Figure 4. Board Trace Model Device and Pin Options



Specify values for Board Trace Model per I/O standard.

I/O standard: 1.5-V HSTL Class I

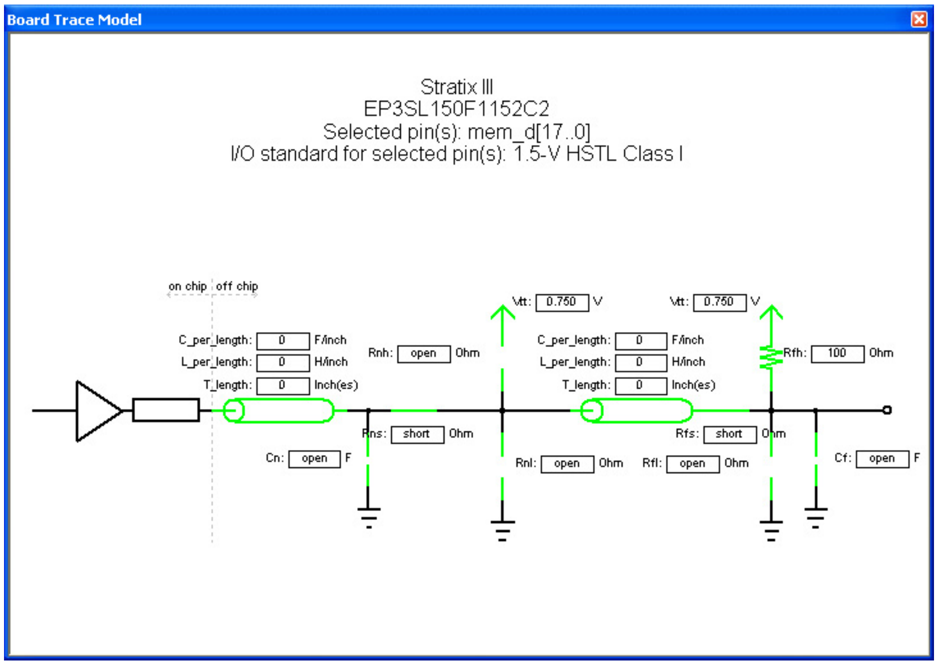
Board trace model:

Name	Value
Near pull-up resistance (in ohms)	open
Near pull-down resistance (in ohms)	open
Near capacitance (in farads)	open
Near series resistance (in ohms)	short
Near transmission line distributed inductance (in h...	0
Near transmission line distributed capacitance (in f...	0
Near transmission line length (in inches)	0
Far transmission line distributed inductance (in he...	0
Far transmission line distributed capacitance (in fa...	0
Far transmission line length (in inches)	0
Far pull-up resistance (in ohms)	50
Far pull-down resistance (in ohms)	open
Far capacitance (in farads)	open

Description:

Specifies board trace, termination, and capacitive load parameters for each I/O standard. Note: These settings affect Advanced I/O Timing only and are used instead of Capacitive Loading to determine I/O timing and power. If Advanced I/O

Figure 5. Board Trace Model Setup for Individual Pin or Net Group


Board Trace Model

Stratix III
EP3SL150F1152C2
Selected pin(s): mem_d[17..0]
I/O standard for selected pin(s): 1.5-V HSTL Class I

C_per_length: 0 F/inch
L_per_length: 0 H/inch
T_length: 0 Inch(es)
Rnh: open Ohm
Rns: short Ohm
Cn: open F
Vtt: 0.750 V
Rnl: open Ohm
C_per_length: 0 F/inch
L_per_length: 0 H/inch
T_length: 0 Inch(es)
Rfs: short Ohm
Rfl: open Ohm
Vtt: 0.750 V
Rfh: 100 Ohm
Cf: open F

Step 6: Compile the Design and Verify the Timing Closure

After constraining the design, compile the design in the Quartus II software. During the generation of the ALTMEMPHY megafunction, the MegaWizard Plug-In Manager generates a script called *<variation_name>***_report_timing.tcl** that generates a custom timing margin report for that PHY instance. After compiling the design in the Quartus II software, run the timing script to produce the timing report for different timing paths in the design, such as write data, read data, address, and command timing paths.

For information about timing constraints and timing analysis methodology used by the ALTMEMPHY megafunction, refer to *Timing Analysis* section of the *External Memory Interface Handbook*.

Step 7: Adjust the Constraints

In the timing report for your ALTMEMPHY instance, you can see the worst case setup and hold margins for the different timing paths in the design. If the setup and hold margins are unbalanced, achieve a balanced setup and hold margins by adjusting the phase setting of the clocks that are used in these paths.

For example, in the case of the address and the command timing margin, the address and command outputs are clocked by an address and command clock that is 90^{o} offset with respect to the system clock. The system clock is used to clock the K/K# clock outputs going to the memory. If the report timing script indicates that using the default phase-shift-setting for the address and command clock results in more setup margin than hold margin, adjust the address and command clock to have higher phase shift than the default setting so that there will be more hold margin. Similarly, adjust the address and command clock to have fewer phase offset with respect to the system clock if there is more hold margin than setup margin with the default settings. These clock-phase-shift adjustments can be made by editing the ALTMEMPHY megafunction instance in the MegaWizard Plug–In Manager.

For detailed information about the clocks used in the PHY megafunction, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

Step 8: Determine Board Design Constraints

After closing the timing for the design, evaluate the trade-offs posed by various board design choices. Different factors contribute to the signal integrity that can affect the overall timing margin for the memory and the FPGA. Some factors to consider that can affect the signal integrity include the termination scheme used, the slew rate, drive strength settings on the FPGA, and the loading seen by the driver. You should evaluate the trade-offs between the different types of termination schemes, effects of output drive strengths, and the loading, so that you can navigate through the multiple combinations and choose the best possible settings for your design.

Altera offers the following application notes regarding high-speed board design information:

- *AN 315: Guidelines for Designing High-Speed FPGA PCBs*.
- *AN 224: High-Speed Board Layout Guidelines*.
- *AN 75: High-Speed Board Designs*.

Stratix III and Stratix IV devices support both series and parallel on-chip termination (OCT) resistors to improve signal integrity. Another benefit of the Stratix III and the Stratix IV OCT feature is eliminating the need for external termination resistors on the FPGA side. This feature simplifies the board design and reduces the overall board cost. The OCT features offer user-mode calibration to compensate for any variation in voltage and temperature during normal operation to ensure that the OCT values remain constant. The parallel and series OCT features on Stratix III and Stratix IV devices are available in either 25-Ω or 50-Ω setting.

For the uni-directional signals found in the QDRII+ and the QDRII SRAM interfaces, Altera's recommended default setting is calibrated parallel OCT 50-Ω for input signals and calibrated series OCT 50-Ω for output signals. For the write clock signals Altera recommends series OCT 50-Ω without calibration. You should perform board simulations to validate if these recommendations are optimal for your system.

Step 9: Perform Board Level Simulations

To determine the correct board constraints, run board–level simulations to see if the settings provide the optimal signal quality. With many variables that can affect the signal integrity of the memory interface, simulating the memory interface provides an initial indicator of the memory interface's performance. There are various electronic design automation (EDA) simulation tools available to perform board level simulations. The simulations should be performed on the data, clock, control, command and address signals. If the memory interface does not have good signal integrity, adjust the settings, such as the drive strength setting, termination scheme or termination values, to improve the signal integrity.

Note that changing these settings affects the timing. It may be necessary to go back to the timing closure step if these change.

Step 10: Verify FPGA Functionality

Perform system-level verification to correlate the system against your design targets. Use Altera's SignalTap® II Logic Analyzer to help in this effort.

For detailed information about using the SignalTap II, refer to the *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Design Example for QDRII+ SRAM Interface at 400 MHz

This section describes the memory interface design flow outlined above and implements a QDRII+ SRAM memory interface with a Stratix III FPGA using Altera's Stratix III development board. This design example implements an 18–bit wide, 400 MHz or 1600 Mbps memory interface on the Stratix III EP3SL150F1152C2 device.

Although the design example is specifically for the QDRII+ SRAM memory interface with a Stratix III device, the design flow when using a QDRII SRAM or with a Stratix IV FPGA for the memory interface is the same.

This design example can only be instantiated using the Quartus II software version 9.0 and earlier. QDRII or QDRII+ SRAM designs using ALTMEMPHY are no longer available in the Quartus II software version 9.1 and later. You are recommended to migrate or instantiate your QDRII or QDRII+ SRAM designs with UniPHY.

Step 1: Select a Device

For this design, choose the Stratix III EP3SL150F1152C2 device. This device supports four ×18 QDRII interfaces on the I/O banks at each side of the device, at speeds up to 400 MHz. The memory device on the Stratix III development board is an 18-bit wide, 36Mbit, 4–word burst, Cypress CY7C1263V18 QDRII+SRAM device.

Step 2: Create a Project in the Quartus II Software

Create a project in the Quartus II design software to implement your QDRII+ SRAM memory interface. If you have already created a project that you would like to implement the memory interface, open that project. Set the target device as the Stratix III EP3SL150F1152C2 device.

For step-by-step instructions for creating a Quartus II software project, refer to the Tutorial in the Quartus II Online Help.

Step 3: Instantiate PHY

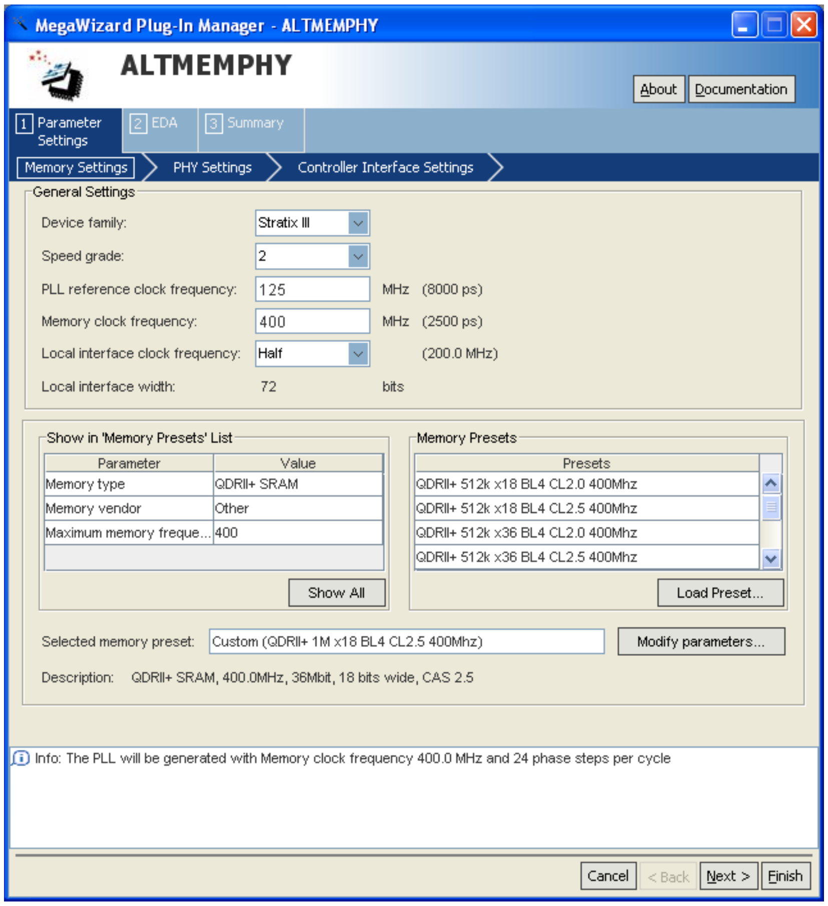
Instantiate the ALTMEMPHY megafunction variation using the following steps:

1. Invoke the MegaWizard Plug-In Manager under the Tools menu in the Quartus II software, and select **Create a new custom megafunction variation**.
2. Select the **ALTMEMPHY** megafunction from the list of I/O megafunctions. Choose **Stratix III** as the target device family, **Verilog HDL** as the output file type, enter an **instance name** of your choice (for example, fred) and click **Next**.
3. In the ALTMEMPHY parameterization page that appears, select **Stratix III** as the target device family, select the **2** speed grade, and enter the **clock rates** for the PLL reference input and the memory interface. For this example, set the PLL reference input frequency to **125 MHz** and memory clock rate to **400 MHz** (Figure 6).

Select **Half** under the selection for **Local interface clock frequency**. Full-rate PHY for the QDRII+/QDRII interface is currently not supported.

From the Memory Preset List, set the **Memory type** to **QDRII+ SRAM** and select **memory preset** titled **QDRII+ 1M ×18 BL4 CL2.5 400 MHz**.

Figure 6. Parameterizing the ALTMEMPHY for Your Target Memory Type and Frequency



General Settings

Device family:	Stratix III		
Speed grade:	2		
PLL reference clock frequency:	125	MHz	(8000 ps)
Memory clock frequency:	400	MHz	(2500 ps)
Local interface clock frequency:	Half		(200.0 MHz)
Local interface width:	72	bits	

Show in 'Memory Presets' List

Parameter	Value
Memory type	QDRII+ SRAM
Memory vendor	Other
Maximum memory freque...	400

Memory Presets

Presets
QDRII+ 512k x18 BL4 CL2.0 400Mhz
QDRII+ 512k x18 BL4 CL2.5 400Mhz
QDRII+ 512k x36 BL4 CL2.0 400Mhz
QDRII+ 512k x36 BL4 CL2.5 400Mhz

Selected memory preset: Custom (QDRII+ 1M x18 BL4 CL2.5 400Mhz)

Description: QDRII+ SRAM, 400.0MHz, 36Mbit, 18 bits wide, CAS 2.5

Info: The PLL will be generated with Memory clock frequency 400.0 MHz and 24 phase steps per cycle

4. The QDRII+SRAM device is a 2M × 18 device, so you need to select the option to **Modify Parameters**. This launches a memory preset editor window that you can modify all the memory parameters. Set the **Address width** to **19**. Verify that the other memory timing parameters match the QDRII+ SRAM datasheet as well. After making all necessary updates, select **OK**. This will create a custom memory preset with your selections.
5. Enter the dedicated clock phase for the address/command clock and the board skew specification for your design. These parameters account for skew across all the memory interface pins including K/K#/CQ/CQ# clock pins, D/Q data pins, and address and command pins. You can also fix the read latency. Use the default setting that is 0 cycle, where the read latency is referred as non-deterministic latency. This minimum read latency equals the worst-case latency measured out of all of the DQS groups. Otherwise, you can specify a latency between 15 and 22 cycles, where the latency is deterministic.

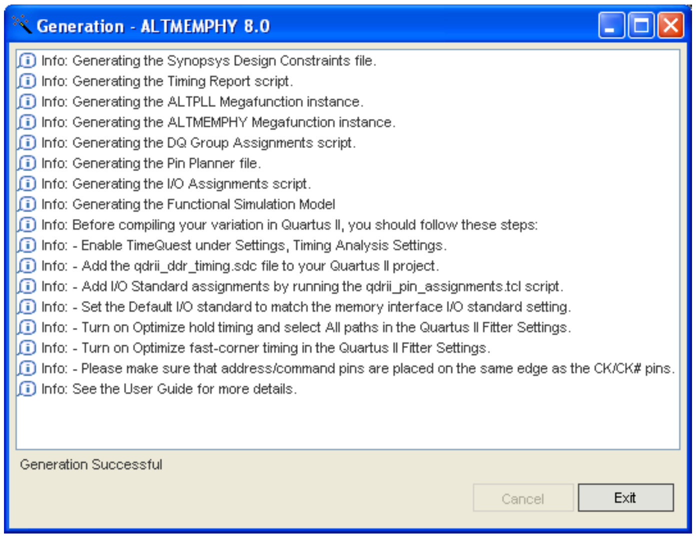
   ☞ For more information on the read latency setting, refer to the *External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

6. In the **Controller Interface Settings** page, click **Next** to proceed.

7. For performing functional simulation, select **Generate simulation model**.
8. Select **Finish**. The MegaWizard Plug-In Manager generates all the necessary files for your ALTMEMPHY instance.

Upon the completion of the ALTMEMPHY megafunction instance generation, the MegaWizard Plug-In Manager provides detailed instructions to apply the generated constraints as shown in Figure 7. This will be discussed in Step 5. With the ALTMEMPHY instance generated, you must now connect the PHY instance to the driver logic in your design.

Figure 7. Instructions for Applying Constraints for the PHY



```
Info: Generating the Synopsys Design Constraints file.
Info: Generating the Timing Report script.
Info: Generating the ALTPLL Megafunction instance.
Info: Generating the ALTMEMPHY Megafunction instance.
Info: Generating the DQ Group Assignments script.
Info: Generating the Pin Planner file.
Info: Generating the I/O Assignments script.
Info: Generating the Functional Simulation Model
Info: Before compiling your variation in Quartus II, you should follow these steps:
Info: - Enable TimeQuest under Settings, Timing Analysis Settings.
Info: - Add the qdrii_ddr_timing.sdc file to your Quartus II project.
Info: - Add I/O Standard assignments by running the qdrii_pin_assignments.tcl script.
Info: - Set the Default I/O standard to match the memory interface I/O standard setting.
Info: - Turn on Optimize hold timing and select All paths in the Quartus II Fitter Settings.
Info: - Turn on Optimize fast-corner timing in the Quartus II Fitter Settings.
Info: - Please make sure that address/command pins are placed on the same edge as the CK/CK# pins.
Info: See the User Guide for more details.
```

Generation Successful

As SRAM devices do not need a memory controller, Altera does not offer such a product. You should connect the local signals of the memory interface from the PHY to the logic that will access the memory in your top-level design.
If you want to use the ×36 emulation mode, refer to the *Creating An Emulated ×36 QDRII+/QDRII SRAM ALTMEMPHY Variation* section of the *External Memory PHY Interface Megafunction User Guide*.

Step 4: Perform RTL/Functional Simulation (Optional)

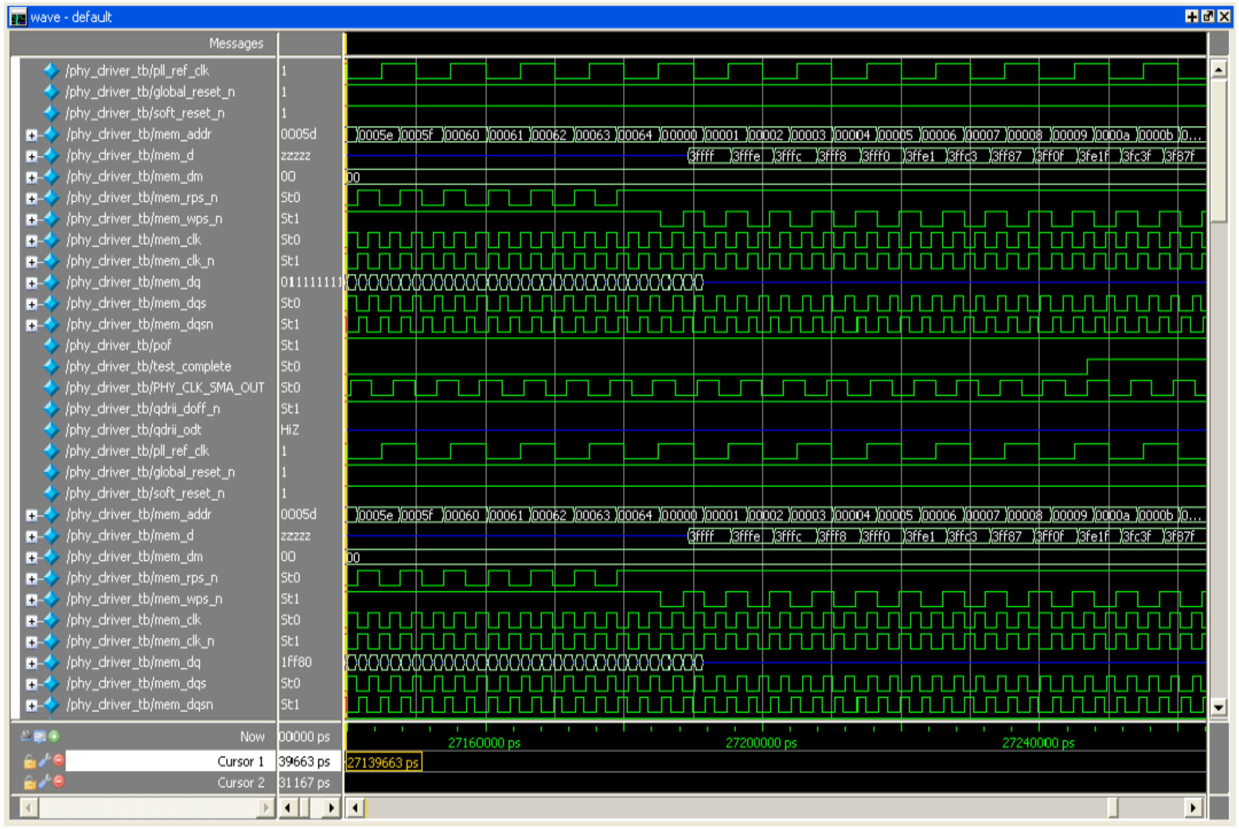
For performing the functional simulation of your memory interface, use the functional models of the ALTMEMPHY megafunction generated by the MegaWizard Plug–In Manager in "Step 3: Instantiate PHY". You should use this model along with your own driver or test bench that issues read and write operations, and a memory model.

The Verilog HDL or VHDL simulation model of the PHY can be found in your project directory and is named <*variation_name*>**_alt_mem_phy_qdrii_sequencer_wrapper.vo/.vho**.

The design example includes the driver, test bench, as well as the memory model that allows you to perform functional simulation on the design. Use the ModelSim-Altera simulator to perform the functional simulation. The design example also comes with a script file that will direct the simulator to perform the necessary compilation and simulation.

Open the ModelSim-Altera simulator and change the path to your project folder. From the File menu, select **Change Directory** and specify your project directory. To run the simulation, from the Tools menu, go to **TCL** and select **Execute Macro**. From your project directory, go to the modelsim directory and run the **run_all.do** file. This file directs the ModelSim simulator to compile the necessary files and execute the **wave.do** file for the simulator to display the waveforms. Figure 8 shows the ModelSim simulation waveform of the design example.

Figure 8. Simulation Waveform


wave - default
Messages
/phy_driver_tb/pll_ref_clk 1
/phy_driver_tb/global_reset_n 1
/phy_driver_tb/soft_reset_n 1
/phy_driver_tb/mem_addr 0005d
/phy_driver_tb/mem_d zzzzz
/phy_driver_tb/mem_dm 00
/phy_driver_tb/mem_rps_n St0
/phy_driver_tb/mem_wps_n St1
/phy_driver_tb/mem_clk St0
/phy_driver_tb/mem_clk_n St1
/phy_driver_tb/mem_dq 011111111
/phy_driver_tb/mem_dqs St0
/phy_driver_tb/mem_dqsn St1
/phy_driver_tb/pof St1
/phy_driver_tb/test_complete St0
/phy_driver_tb/PHY_CLK_SMA_OUT St0
/phy_driver_tb/qdrii_doff_n St1
/phy_driver_tb/qdrii_odt HiZ
/phy_driver_tb/pll_ref_clk 1
/phy_driver_tb/global_reset_n 1
/phy_driver_tb/soft_reset_n 1
/phy_driver_tb/mem_addr 0005d
/phy_driver_tb/mem_d zzzzz
/phy_driver_tb/mem_dm 00
/phy_driver_tb/mem_rps_n St0
/phy_driver_tb/mem_wps_n St1
/phy_driver_tb/mem_clk St0
/phy_driver_tb/mem_clk_n St1
/phy_driver_tb/mem_dq 1ff80
/phy_driver_tb/mem_dqs St0
/phy_driver_tb/mem_dqsn St1
0005e 0005f 00060 00061 00062 00063 00064 00000 00001 00002 00003 00004 00005 00006 00007 00008 00009 0000a 0000b 0...
3ffff 3fffe 3fffc 3fff8 3fff0 3ffe1 3ffc3 3ff87 3ff0f 3fe1f 3fc3f 3f87f
00
Now 00000 ps
Cursor 1 39663 ps
Cursor 2 31167 ps
27160000 ps
27200000 ps
27240000 ps
27139663 ps

The testbench included with the design files is set to 125 MHz. If your design requires a different setting, the PLL reference clock (`pll_ref_clk`) in the testbench must be modified to match the PLL reference clock settings in the MegaWizard Plug-In Manager.

Step 5: Add Constraints

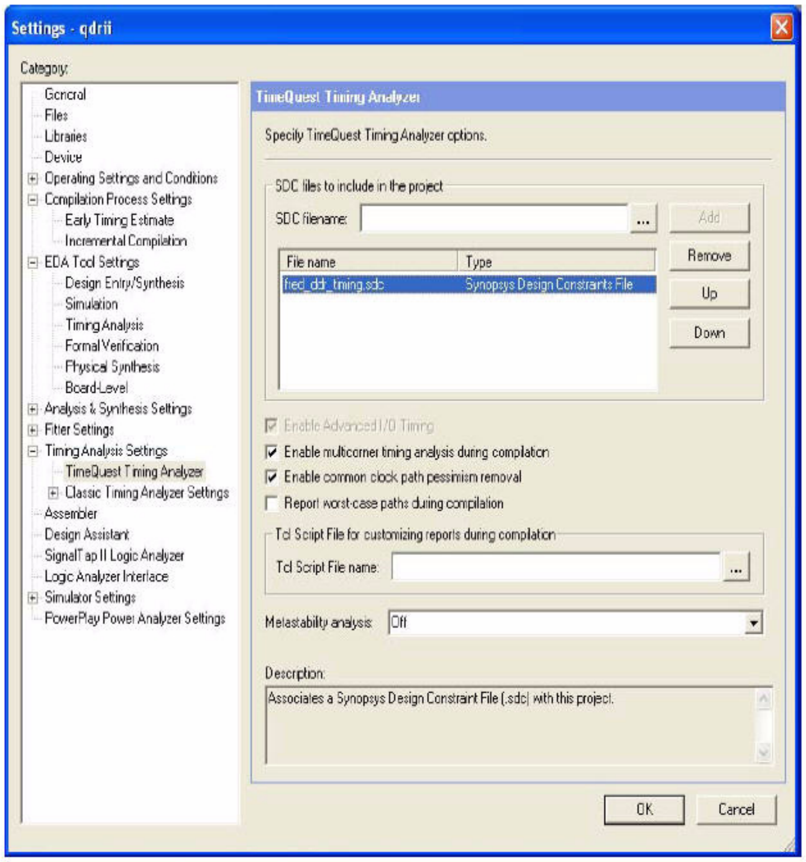
As mentioned in Step 3, Figure 7 lists the steps you should follow to add the relevant timing and device constraints for your memory interface.

To add constraints, follow these steps:

1. Enable the TimeQuest timing analyzer tool under **Settings**, **Timing Analyzer Settings**. All designs using the ALTMEMPHY megafunction are required to use the TimeQuest timing analyzer for proper and accurate timing analysis.

2. Add the <*variation_name*>**_ddr_timing.sdc** file to your project. This file contains all the necessary timing constraints for the memory interface portion of your design (Figure 9).

Figure 9. Adding Timing Constraints for the PHY



Specify TimeQuest Timing Analyzer options.

SDC files to include in the project

SDC filename:

File name	Type
fied_ddr_timing.sdc	Synopsys Design Constraints File

[✓] Enable Advanced I/O Timing
[✓] Enable multicorner timing analysis during compilation
[✓] Enable common clock path pessimism removal
[ ] Report worst-case paths during compilation

Tcl Script File for customizing reports during compilation

Tcl Script File name:

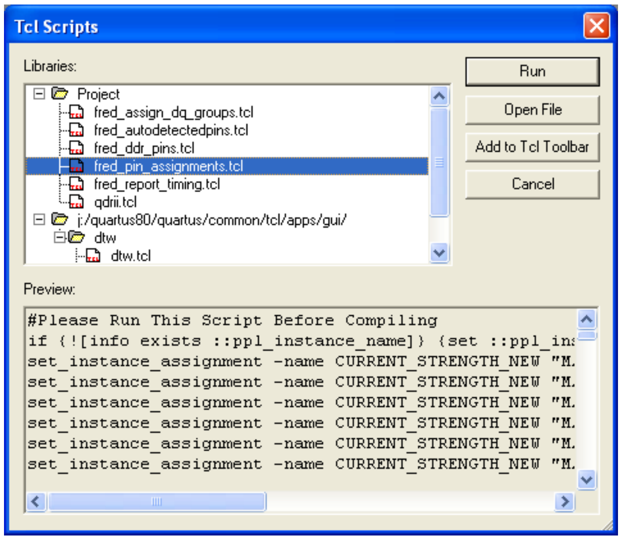
Metastability analysis: Off

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

3. Add the I/O standard and output enable group assignments for your memory interface pins by executing the <*variation_name*>**_pin_assignments.tcl** file (Figure 10). Output enable group assignments are required so as not to violate the requirements for the maximum number of pins driving out of a VREF group when a voltage-referenced input pin or bi-directional pin is present.

Note that the pin names listed in this script are generic names. You should modify them to match the top-level pin names in your design.

Figure 10. Run Pin Assignment



```
#Please Run This Script Before Compiling
if {![info exists ::ppl_instance_name]} {set ::ppl_in
set_instance_assignment -name CURRENT_STRENGTH_NEW "M
set_instance_assignment -name CURRENT_STRENGTH_NEW "M
set_instance_assignment -name CURRENT_STRENGTH_NEW "M
set_instance_assignment -name CURRENT_STRENGTH_NEW "M
set_instance_assignment -name CURRENT_STRENGTH_NEW "M
set_instance_assignment -name CURRENT_STRENGTH_NEW "M
```

4. Enable the **Optimize Hold Timing** option for **All Paths** for the Quartus II fitter, under **Settings**, **Fitter Settings**. Specifying the All Paths setting directs the Fitter to optimize the hold time of all paths by adding delay to the appropriate paths.

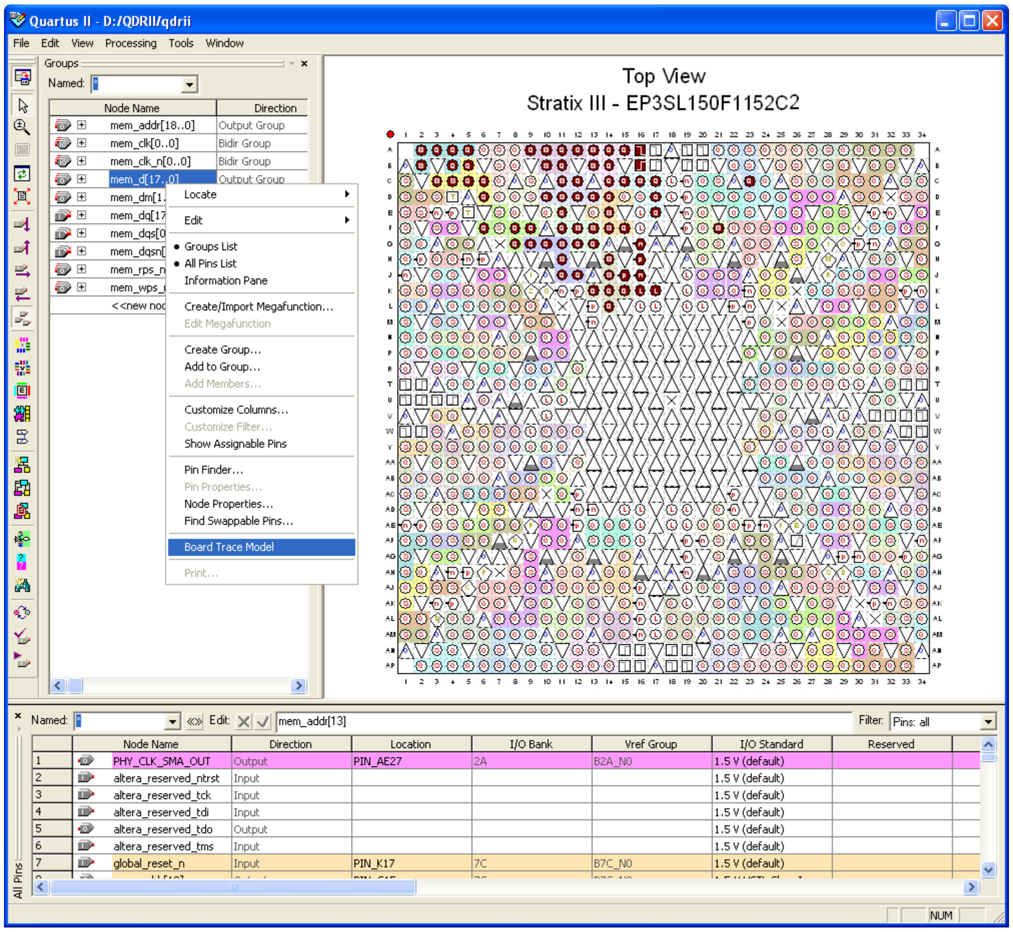
Note that the pin names listed in this script are generic names and you should modify them to match the top-level pins names in your design.
Executing the <*variation_name*>**_pin_assignments.tcl** file does not make the pin location assignments to the pins. You still need to make the pin location assignments based on your board layout. For the Stratix III Host Board, you can use the **SIII_host_qdrii_pin_location.tcl** file from the design example to create the pin location assignment.

In addition to adding constraints for all of your memory interface blocks, you should ensure that the rest of your design is also appropriately constrained. For example, you may need to add timing constraints into other clocks in your design, including the reference clock input to the ALTMEMPHY megafunction's PLL instance.

When using the Quartus II Advanced I/O Timing option, provide the board trace information for the output and bidirectional pins of your design through the Board Trace Model in the Quartus II software. Through the Quartus II Pin Planner, you can enter the board trace information for every output or bidirectional pin, or for a net group separately.

From the Quartus II Pin Planner, right-click on the pin or net group, and select Board Trace Model, as in Figure 11.

Figure 11. Defining Board Trace Model in the Quartus II Pin Planner



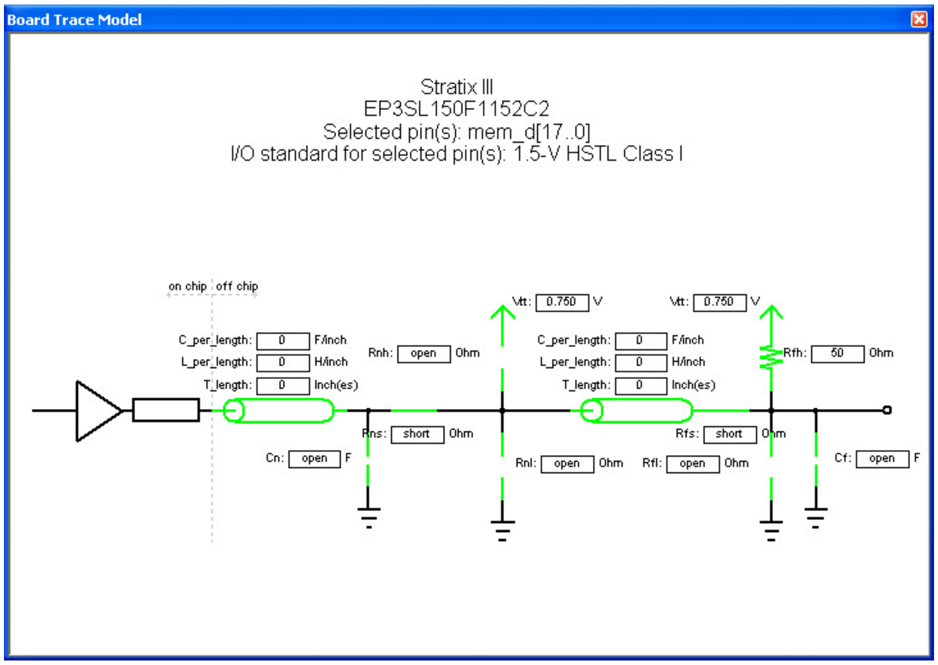
Node Name	Direction
mem_addr[18..0]	Output Group
mem_clk[0..0]	Bidir Group
mem_clk_n[0..0]	Bidir Group
mem_d[17..0]	Output Group
mem_dm[1	
mem_dq[17	
mem_dqs[0	
mem_dqsn[	
mem_rps_n	
mem_wps_	
<<new noc	

mem_addr[13]

	Node Name	Direction	Location	I/O Bank	Vref Group	I/O Standard	Reserved
1	PHY_CLK_SMA_OUT	Output	PIN_AE27	2A	B2A_N0	1.5 V (default)	
2	altera_reserved_ntrst	Input				1.5 V (default)	
3	altera_reserved_tck	Input				1.5 V (default)	
4	altera_reserved_tdi	Input				1.5 V (default)	
5	altera_reserved_tdo	Output				1.5 V (default)	
6	altera_reserved_tms	Input				1.5 V (default)	
7	global_reset_n	Input	PIN_K17	7C	B7C_N0	1.5 V (default)	

Figure 12 shows the board trace model window.

Figure 12. Board Trace Model



Stratix III
EP3SL150F1152C2
Selected pin(s): mem_d[17..0]
I/O standard for selected pin(s): 1.5-V HSTL Class I

C_per_length: 0 F/inch
L_per_length: 0 H/inch
T_length: 0 Inch(es)
Cn: open F
Rns: short Ohm
Rnh: open Ohm
Vtt: 0.750 V
Rnl: open Ohm
C_per_length: 0 F/inch
L_per_length: 0 H/inch
T_length: 0 Inch(es)
Vtt: 0.750 V
Rfh: 50 Ohm
Rfs: short Ohm
Rfl: open Ohm
Cf: open F

Depending on your board, enter the values such as the trace length, capacitance/length, inductance/length, pull-up, pull-down and others.

Step 6: Compile the Design and Verify the Timing Closure

After your design is constrained, you are ready to compile your design.

After successfully compiling your design in the Quartus II software, run the timing reporting script generated by the ALTMEMPHY megafunction during the megafunction instantiation. This script is called <variation_name>_report_timing.tcl and it produces the timing report for the memory interface portion of your design. You can either run the **.tcl** file from the TimeQuest timing analyzer, or from the Quartus II Tcl Console.

To run the report timing script in the TimeQuest timing analyzer window, open the tool in the Quartus II software. Then complete the following tasks:

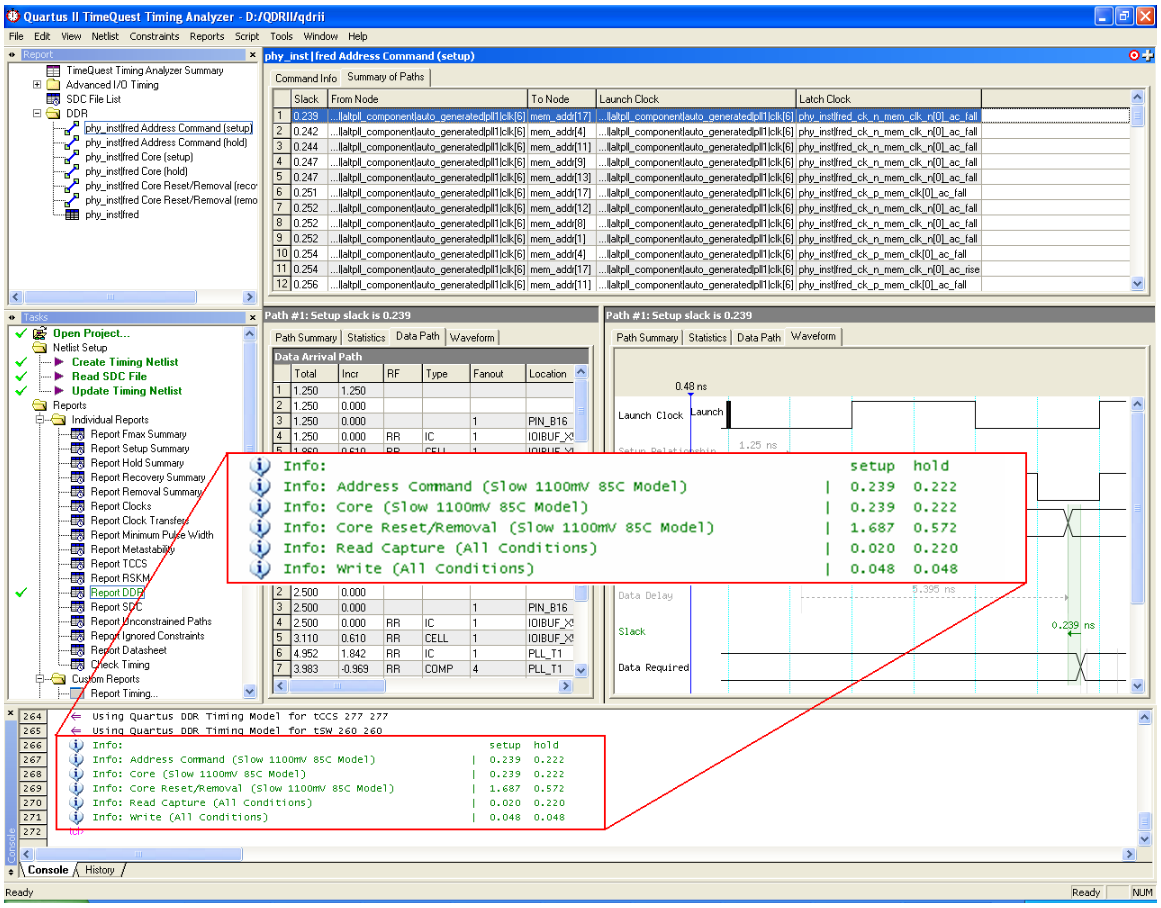
- Update timing netlist
- Create timing netlist
- Read SDC file

Perform these tasks by double-clicking **Update Timing Netlist** in the left **Tasks** pane that will automatically perform the three tasks. After a task is executed, the task in the **Tasks** pane turns green.

Run the report timing script next by going to the Script menu and clicking **Run Tcl Script**. Select the *<variation_name>***_report_timing_tcl**. You can also double click **Report DDR** on the **Tasks** pane to get the timing numbers.

Figure 13 shows the timing margin reported in the TimeQuest timing analyzer window after running the report timing script.

Figure 13. Running the Report Timing Script



Quartus II TimeQuest Timing Analyzer - D:/QDRII/qdrii

phy_inst|fred Address Command (setup) — Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock
1	0.239	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[17]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_fall
2	0.242	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[4]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_fall
3	0.244	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[11]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_fall
4	0.247	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[9]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_fall
5	0.247	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[13]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_fall
6	0.251	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[17]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_p_mem_clk[0]_ac_fall
7	0.252	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[12]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_fall
8	0.252	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[8]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_fall
9	0.252	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[1]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_fall
10	0.254	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[4]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_p_mem_clk[0]_ac_fall
11	0.254	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[17]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_n_mem_clk_n[0]_ac_rise
12	0.256	...ll altpll_component|auto_generated|pll1|clk[6]	mem_addr[11]	...ll altpll_component|auto_generated|pll1|clk[6]	phy_inst|fred_ck_p_mem_clk[0]_ac_fall

Path #1: Setup slack is 0.239

Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location
1	1.250	1.250				
2	1.250	0.000				
3	1.250	0.000			1	PIN_B16
4	1.250	0.000	RR	IC	1	IOIBUF_X!

	Total	Incr	RF	Type	Fanout	Location
2	2.500	0.000				
3	2.500	0.000			1	PIN_B16
4	2.500	0.000	RR	IC	1	IOIBUF_X!
5	3.110	0.610	RR	CELL	1	IOIBUF_X!
6	4.952	1.842	RR	IC	1	PLL_T1
7	3.983	-0.969	RR	COMP	4	PLL_T1

```
264  ⇐ Using Quartus DDR Timing Model for tCCS 277 277
265  ⇐ Using Quartus DDR Timing Model for tSW 260 260
266    Info:                                                  setup  hold
267    Info: Address Command (Slow 1100mV 85C Model)        |  0.239  0.222
268    Info: Core (Slow 1100mV 85C Model)                   |  0.239  0.222
269    Info: Core Reset/Removal (Slow 1100mV 85C Model)     |  1.687  0.572
270    Info: Read Capture (All Conditions)                  |  0.020  0.220
271    Info: Write (All Conditions)                         |  0.048  0.048
272    tcl>
```

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

- Address and command setup and hold margin.
- Core setup and hold margin.
- Core reset and removal setup and hold margin.
- Read capture setup and hold margin.
- Write setup and hold margin.

The timing should be verified on every corner of the timing model. You must run the timing report script with all available timing models: slow 0^{o} C, slow 85^{o} C, and fast 0^{o} C model to ensure positive margins across the process, the voltage and the temperature. To analyze timing on a different corner, first double-click **Set Operating Conditions** in the left pane. Select a new timing corner, then go to the Script menu and rerun the *<variation>*_**report_timing.tcl** script.

For more information about the TimeQuest timing analyzer window, refer to the *TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Step 7: Adjust the Constraints

If the timing report indicates that the setup or hold margins for one of the timing paths is negative, you must adjust the constraints on the design to improve margins for the failing path.

For instance, suppose that the address command timing path fails the hold time, you need to adjust the PLL clock-phase-shift setting for the clock generating the address and command signals to improve the hold margins (by increasing the phase-shift on the clock). This clock setting could be adjusted by editing the ALTMEMPHY megafunction instance in the MegaWizard Plug-In Manager.

Step 8: Verify FPGA Functionality

In order to use SignalTap II Embedded Logic Analyzer to help you verify your design on the board, create an **.stp** file and add in the necessary signals. You need to specify a clock source as well as the triggering conditions.

☞ The SignalTap II Embedded Logic Analyzer does not allow you to monitor the D,DQ,DQS, and DQSn signals at the pins.

The design example includes the **.stp** file for you to verify the design on the Stratix III Host Board. Open the **system_test.stp** file to perform the verification using the SignalTap II Embedded Logic Analyzer. This file has the relevant internal signals added so that you can monitor the signal activities when the Stratix III device performs the read and write operations on the QDRII+SRAM device. Upon successfully configuring the device on the board, you can start monitoring the signal activities. The high `pnf` signal indicates the design works correctly.

Design Checklist

Table 2 contains a design checklist that you can use when implementing QDRII+ and QDRII SRAM memory interfaces in Stratix III and Stratix IV devices.

Table 2. Checklist for Implementing QDRII+ and QDRII SRAM Memory Interfaces in Stratix III Devices (Part 1 of 2)

Item	Description	Yes/No
Select Device		
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, as well as the FPGA must have sufficient I/O pins for the DQ/DQS read and write groups. For detailed device resource information, refer to the device handbook chapter on external memory interface support.	
Instantiate PHY and Controller		
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 the *Implementing Multiple ALTMEMPHY-Based Controllers* chapter in the *DDR, DDR2, and DDR3 SDRAM Design Tutorials* section of the *External Memory Interface Handbook*.	
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		
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		
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 scripts 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). Provide the board trace information for the output and bidirectional pins through the Board Trace Model in the Quartus II software.	

Table 2. Checklist for Implementing QDRII+ and QDRII SRAM Memory Interfaces in Stratix III Devices (Part 2 of 2)

Item	Description	Yes/No
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. Run this process across all device timing models (slow 0^o C, slow 85^o C, fast 0^o 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.	
Board Level Considerations		
8	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. Stratix III and Stratix IV devices support on-chip termination. Altera recommends calibrated parallel OCT 50 Ω setting for uni-directional input signals from the memory, calibrated series OCT 50 Ω setting for uni-directional output signals to the memory, and series OCT 50 Ω without calibration for the write clock signals. When using the OCT feature on the Stratix III and Stratix IV 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).	
9	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 Quartus II timing margin reports.	
System Verification		
10	Have you verified functionality of your memory interface in system?	

Conclusion

Stratix III and Stratix IV FPGAs complement high-performance QDRII+ and QDRII SRAM devices by providing high memory bandwidth, improved timing margin, and great flexibility in system design. QDRII+ and QDRII SRAM with Stratix III and Stratix IV devices support the high-throughput requirements of today's communication, networking, and digital signal processing systems.

Document Revision History

Table 3 shows the revision history for this document.

Table 3. Document Revision History

Date & Document Version	Changes Made	Summary of Changes
February 2010 v1.2	■ Moved specification information to the *External Memory Interface Handbook*. ■ Updated figures for Stratix III and Stratix IV IOE Input, and Output and Output-Enabled Registers: Figure 1 and Figure 2. ■ Updated references.	—
July 2008 v1.1	■ Changed application notes title to "AN 461: Design Guidelines for Implementing QDRII+ and QDRII SRAM Interfaces in Stratix III and Stratix IV Devices". ■ Updated specification information for Stratix III devices. ■ Added specification information for Stratix IV devices. ■ Added figures for Stratix III and Stratix IV IOE Input, and Output and Output-Enabled Registers: Figure 1 and Figure 2. ■ Updated GUI-related figures. ■ Added new paragraph under "Step 3: Instantiate PHY", "Step 4: Perform RTL/Functional Simulation (Optional)", "Step 5: Add Constraints", and "Step 7: Adjust the Constraints" sections.	—
June 2007 v1.0	Initial release	—