Interfacing QDRII+ & QDRII with Stratix II, Stratix II GX, Stratix, & Stratix GX Devices

May 2008, ver. 5.1 Application Note 326

Introduction

Synchronous static RAM (SRAM) architectures support the high throughput requirements of communications, networking, and digital signal processing (DSP) systems. The successor to quad data rate (QDR) SRAM, QDRII+ and QDRII SRAM support higher memory bandwidth and improved timing margins and offer more flexibility in system designs.

Tables 1 and 2 show the maximum clock frequency support for each device family.

Table 1. QDRII+ and QDRII SRAM Maximum Clock Frequency Support in Stratix II and Stratix II GX Devices Notes (1), (2), (3)

Speed Grade	Frequency (MHz)	
	DLL-Based Implementation	PLL-Based Implementation (4)
–3	300	200
–4	250	167
–5	250	167

Notes to Table 1:

(1) 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.

(2) These numbers are from the Quartus® II software, version 7.2 SP1. Altera recommends using the latest version of the Quartus II software for your design.

(3) These numbers apply to both commercial and industrial devices.

(4) The lowest frequency at which a QDRII+ SRAM device can operate is 238 MHz. Therefore, the PLL-based implementation does not support the QDRII+ SRAM memory interface.

Table 2. QDRII SRAM Maximum Clock Frequency Supported in Stratix and Stratix GX Devices Notes (1), (2), (3), (4), (5)

Speed Grade	Frequency (MHz) EP1S10 to EP1S40 and all Stratix GX Devices	Frequency (MHz) EP1S60 to EP1S80 Devices
–5	200	167
–6	167	167
–7	133	133

Notes to Table 2:

(1) 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.

(2) These numbers are from the Quartus II software, version 7.2 SP1. Altera recommends using the latest version of the Quartus II software for your design.

(3) These numbers apply to both commercial and industrial devices.

(4) For more information refer to *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

(5) Stratix® and Stratix GX devices support only PLL-based implementation.

This application note describes essential information you will need to interface a QDRII+/QDRII SRAM memory device with a Stratix II, Stratix II GX, Stratix, and Stratix GX device. Additionally, this application note contains an example design to help walk you through how to set up a QDRII SRAM interface in a Stratix II device.

To design QDRII+/QDRII with Stratix II or Stratix II GX, use this application note with the *External Memory Interfaces* chapters of the *Stratix*, *Stratix II*, or *Stratix GX Device Handbooks*, *Application Note 449: External Memory Interface Design Guidelines for Stratix II, Stratix II GX, and Arria GX Devices*.

To design QDRII with Stratix or Stratix GX, use this application note with the *External Memory Interfaces* chapters of the *Stratix*, *Stratix II*, or *Stratix GX Device Handbooks*, and *AN 349: QDR SRAM Controller Reference Design for Stratix & Stratix GX Devices*.

QDRII+ and QDRII SRAM Overview

QDRII+ and QDRII SRAM can perform two data writes and two data reads per clock cycle. They use one port for writing data (D) and another port for reading data (Q). These unidirectional data ports support simultaneous reads and writes and allows back-to-back transactions without the contention issues that can occur when using a single bidirectional data bus. Write and read operations share address ports.

The QDRII SRAM devices are available in ×8, ×9, ×18, and ×36 data bus width configurations. The QDRII+ SRAM devices are available in ×9, ×18, and ×36 data bus width configurations. Write and read operations are burst-oriented. All the data bus width configurations of QDRII SRAM support burst lengths of two and four. When all the data bus width configurations of QDRII+ SRAM supports only a burst length of four. Burst-of-two and burst-of-four for QDRII and burst-of-four for QDRII+ SRAM devices provide the same overall bandwidth at a given clock speed.

Read latency is the time between the read command being clocked into memory and the time data is presented at the memory pins. For QDRII SRAM devices, the read latency is 1.5 clock cycles, while for QDRII+ SRAM devices it is 2 or 2.5 clock cycles, depending on the memory device. Write latency is the time between the write command being clocked into memory and the time data is presented at the memory pins. For QDRII+ and burst-of-four QDRII SRAM devices, the write commands and addresses are clocked on the rising edge of clock and write latency is one clock cycle. For burst-of-two QDRII SRAM devices, the write command is clocked on the rising edge of clock and the write address is clocked on the falling edge of clock. Therefore, the write latency is zero, because the write data is presented at the same time as the write command.

Stratix and Stratix GX devices use 1.5-V HSTL Class I or II to interface QDRII SRAM devices. Stratix II and Stratix II GX devices use either 1.5-V HSTL or 1.8-V HSTL Class I or II I/O standard. For maximum performance, Altera recommends 1.8-V HSTL I/O standard.

QDRII+ and QDRII SRAM devices use a delay-locked loop (DLL) inside the device to edge-align the data with respect to the K and Kn or C and Cn pins (refer to Table 3). You can optionally turn off the DLL, but the performance of the QDRII+ and QDRII SRAM devices is degraded. All timing specifications listed in this document assume that the DLL is on.

QDRII+ and QDRII SRAM devices also offer programmable impedance output buffers. You can set the buffers by terminating the ZQ pin to V_{SS} through a resistor, RQ. The value of RQ should be five times the desired output impedance. The range for RQ should be between 175 Ωand 350 Ω with a tolerance of 10%.

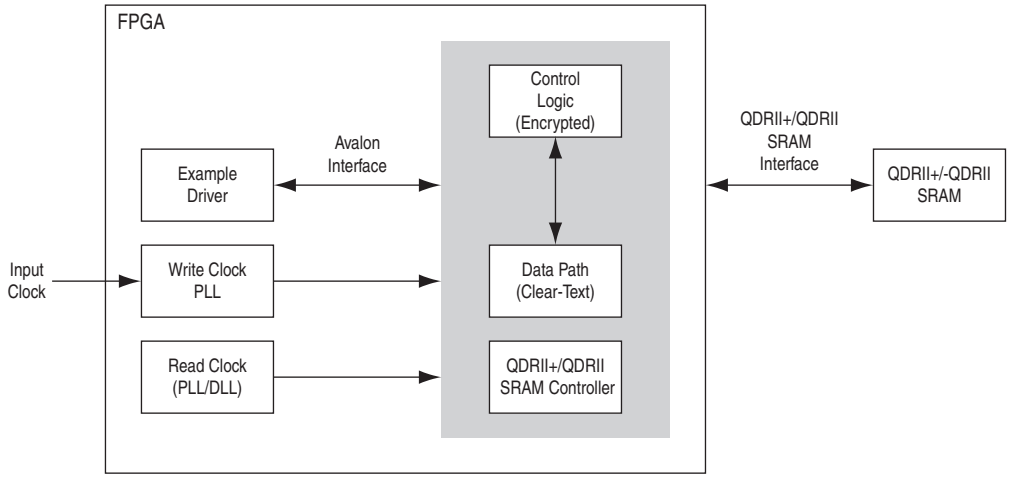
Interface Description

This section provides a description of the interface signals between the FPGA and the QDRII+ and QDRII SRAM devices.

The QDRII SRAM Controller MegaCore® function provides an easy-to-use interface to QDRII+ and QDRII SRAM devices. The QDRII SRAM Controller ensures that the placement and timing are in line with QDRII+ and QDRII SRAM specifications.

The Altera® QDRII+ and QDRII SRAM MegaCore function allows you to use the Altera-recommended data path regardless of whether you use the Altera memory controller MegaCore function. When you use this provided data path, the QDRII+ and QDRII SRAM controller MegaCore function constrains your interface pins and data path logic optimal operation. Figure 1 shows the FPGA to QDRII+ or QDRII SRAM interface.

Figure 1. QDRII+ and QDRII SRAM Controller System Level Block Diagram



The ALTMEMPHY megafunction—an interface between the memory controller and the memory devices to perform read and write operations to the memory—is only available for QDRII+/QDRII SRAM in Stratix III devices.

For more information about the ALTMEMPHY megafunction, refer to the *External DDR Memory PHY Interface Megafunction User Guide (ALTMEMPHY)*.

Interface Signals

This section provides a description of the clock, control, address, and data signals on QDRII+ and QDRII SRAM devices.

Table 3 shows the QDRII+ and QDRII SRAM interface pins and how to connect them to Stratix II, Stratix II GX, Stratix, and Stratix GX devices.

Table 3. QDRII+ and QDRII SRAM Interface Pins

Pins	Description	Stratix II Device Pin Utilization (DLL-Based)	Stratix II Device Pin Utilization (PLL-Based)	Stratix and Stratix GX Device Pin Utilization (PLL-Based) *(1)*
D	Write data	User I/O pin	User I/O pin	User I/O pin
Q	Read data	DQ	User I/O pin	User I/O pin
K	Write clock	User I/O pin	User I/O pin	User I/O pin
K#	Inverted write clock	User I/O pin	User I/O pin	User I/O pin
C *(2)*	Read clock	N/A	N/A	N/A
C# *(2)*	Inverted read clock	N/A	N/A	N/A
CQ	Echo clock	DQS	PLL dedicated clock input	PLL dedicated clock input
CQ#	Inverted echo clock	DQSn	Not used	Not used
WPSn	Write port select	User I/O pin	User I/O pin	User I/O pin
RPSn	Read port select	User I/O pin	User I/O pin	User I/O pin
A	Address	User I/O pin	User I/O pin	User I/O pin
BWSn	Byte write select	User I/O pin	User I/O pin	User I/O pin
QVLD *(3)*	Optional valid output indicator	DQVLD *(4)*	N/A	User I/O pin
All other	Address and command	User I/O pin	User I/O pin	User I/O pin

Notes to Table 3:

(1) Stratix and Stratix GX devices do not offer DLL-based QDRII SRAM implementation.

(2) When interfacing with one QDRII+ or QDRII SRAM device, Stratix II, Stratix, and Stratix GX devices use single-clock mode where the QDRII+ or QDRII SRAM device's C or C# port is tied to supply voltage (V_{DD}).

(3) The QVLD signal is only available in QDRII+ SRAM devices.

(4) The Quartus II software labels this pin as DQ pins in the top and bottom I/O banks.

Clock Signals

QDRII+ and QDRII SRAM devices have three pairs of clocks:

- Input clocks K and K#
- Input clocks C and C#
- Echo clocks CQ and CQ#

The positive input clock, K, is the logical complement of the negative input clock, K#. Similarly, C and CQ are complements of C# and CQ#, respectively.

The QDRII+ and QDRII SRAM devices use the K and K# clocks for write access and the C and C# clocks for read accesses only when interfacing more than one QDRII+ or QDRII SRAM device. Because the number of loads that the K and K# clocks drive affects the switching times of these outputs when a controller drives a single QDRII+ or QDRII SRAM device, C and C# are unnecessary. This is because the propagation delays from the controller to the QDRII+ or QDRII SRAM device and back are the same. Therefore, to reduce the number of loads on the clock traces, QDRII+ and QDRII SRAM devices have a single clock mode, and the K and K# clocks are used for both reads and writes. In this mode, the C and C# clocks are tied to the supply voltage (V_{DD}).

CQ and CQ# are the source-synchronous output clocks from the QDRII or QDRII+ SRAM device that accompanies the read data.

The Stratix II, Stratix II GX, Stratix, and Stratix GX device outputs the K and K# clocks, data, address, and command lines to the QDRII+ or QDRII SRAM device. For the controller to operate properly, the write data (D), address (A), and control signal trace lengths (and therefore the propagation times) should be equal to the K and K# clock trace lengths.

You can generate C, C#, K, and K# clocks using any of the I/O registers via the DDR registers. Because of strict skew requirements between K and K# signals, use adjacent pins to generate the clock pair. The propagation delays for K and K# from the FPGA to the QDRII+ or QDRII SRAM device are equal to the delays on the data and address (D, A) signals. Therefore, the signal skew effect on the write and read request operations is minimized by using identical DDR output circuits to generate clock and data inputs to the memory.

Control Signals

QDRII+ and QDRII SRAM devices use the write port select (`WPSn`) signal to control write operations and the read port select (`RPSn`) signal to control read operations. The byte write select signal (`BWSn`) is a third control signal that indicates to the QDRII+ or QDRII SRAM device which byte to write into the QDRII+ or QDRII SRAM device. You can use any of the FPGA's user I/O pins to generate control signals, preferably on the same side and the same bank.

Address Signals

QDRII+ and QDRII SRAM devices use one address bus (A) for both read and write addresses. You can use any of the FPGA's user I/O pins to generate address signals, preferably on the same side and the same banks.

Data and QVLD Signals

QDRII+ and QDRII SRAM devices use two unidirectional data buses: one for writes (D) and one for reads (Q). Connect Q to the DQ pins on the Stratix II and Stratix II GX FPGAs. On Stratix and Stratix GX FPGAs you can connect the Q pins to any FPGA user I/O pins in I/O banks 3, 4, 7, and 8. You can connect the QDRII+ and QDRII SRAM D ports to any of the user I/O pins in the same I/O bank as the Q pins.

QDRII+ SRAM devices also have a QVLD pin that indicates valid read data. The QVLD signal is edge-aligned with the echo clock and is asserted high for approximately half a clock cycle before data is output from memory. If you are designing your own controller, you can connect the QVLD pin from the QDRII+ SRAM device to the DQVLD pin in the Stratix II or Stratix II GX device.

Stratix and Stratix GX devices do not support QDRII+ SRAM.

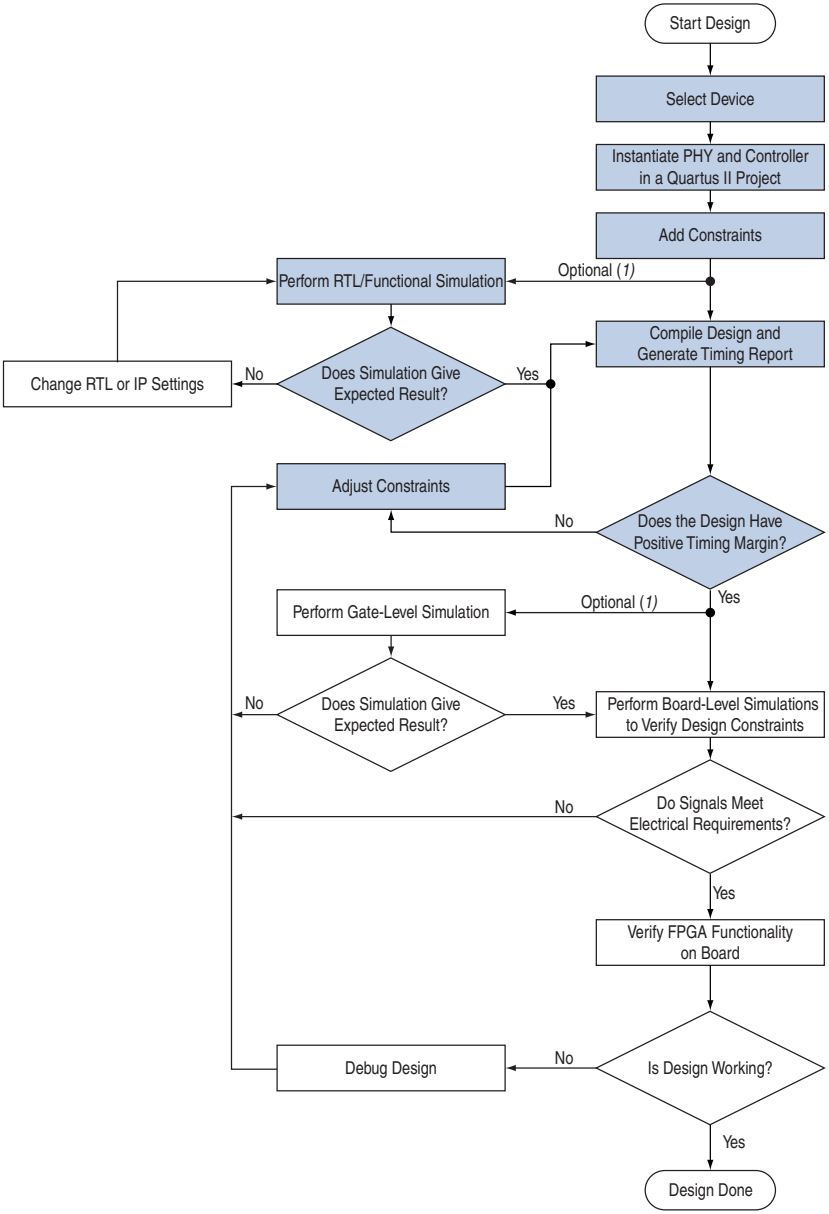
Altera's QDRII SRAM Controller MegaCore function does not use the QVLD signal from the QDRII+ SRAM, because Altera's legacy PHY is expected to work with both QDRII+ and QDRII SRAM. Because QDRII does not have the QVLD signal, Altera's PHY is designed to detect which words are valid during the calibration phase. Altera therefore does not use the QVLD signal.

Memory Interface Design Flow

This section describes the Altera-recommended memory interface design flow, shown in Figure 2, for successful memory interface implementation in Stratix II or Stratix II GX devices. These guidelines provide the best experience with external memory interfaces in these device families. The shaded boxes in the design flow are the steps detailed in the design example of this Application Note.

For more information about this process flow, a detailed description of each step appears in *AN 449: External Memory Interface Design Guidelines for Stratix II, Stratix II GX, and Arria GX Devices*.

Figure 2. Design Flow for Implementing External Memory Interfaces in Stratix II and Stratix II GX Devices



Note to Figure 2:

(1) Though this is an optional step, Altera recommends you perform this step to ensure design functionality.

Table 4 features a design checklist that you can use when implementing QDRII+ and QDRII SRAM memory interfaces in Stratix II and Stratix II GX devices.

Table 4. Checklist for Implementing QDRII+/QDRII SRAM Memory Interfaces in Stratix II and Stratix II GX Devices (Part 1 of 3)

Item	Description
Select Device	
1	• Have you selected the memory interface frequency of operation and bus width? • 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 relevant device handbook chapter on external memory interface support.
2	• Have you selected your QDRII+/QDRII Controller implementation? Altera provides two methods of read-side instantiation: PLL-based and DLL-based SRAM. For more information, refer to "Appendix A: Interface Architecture" on page 57 and to *External Memory Interfaces in Stratix II and Stratix II GX Devices* in the *Stratix II Device Handbook*.
Instantiate Controller	
3	• Have you parameterized and instantiated the controller for your target memory interface? For more information about instantiating your controller, refer to *QDRII SRAM Controller Megacore Function User Guide*. For more information about using legacy PHY, refer to *AN 392: Implementing Multiple Legacy DDR/DDR2 SDRAM Controller Interfaces.*
4	• If you are using your own controller, 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 PHY 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.
5	• Have you simulated your design using the RTL functional model? Use the QDRII+/QDR II SRAM Controller megafunction functional simulation model in conjunction with your own driver logic/testbench and a memory model to ensure proper read and write transactions to the memory. For more information about the QDRII+/QDRII SRAM Controller megafunction, refer to the *QDRII SRAM Controller Megacore Function User Guide*.

Table 4. Checklist for Implementing QDRII+/QDRII SRAM Memory Interfaces in Stratix II and Stratix II GX Devices (Part 2 of 3)

Item	Description
Timing Closure	
6	• Have you added constraints to the QDRII+/QDRII SRAM controller and the rest of your design? The QDRII+/QDRII SRAM Controller megafunction is constrained when you use the generated **.sdc** file and **.tcl** files. However, you may need to adjust these settings to fit your memory interface configuration. The QDRII+/QDRII SRAM controller is constrained with the DDR Timing Wizard (DTW) and the generated **.tcl** file. For more information, refer to the *DDR Timing Wizard User Guide*. • Altera recommends using DTW to constrain the QDRII+/QDRII controller. • 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.
7	• Have you compiled your design and verified timing closure using all available models? • Run the **dtw_timing_analysis.tcl** script for the QDRII+/QDRII controller. For more information, refer to the *DDR Timing Wizard User Guide*.
8	• If there are timing violations, have you adjusted your constraints to optimize timing? If you are using TimeQuest Timing Analysis, adjust the PLL clock phase shift settings or appropriate timing and location assignment margins for the various timing paths within the controller, according to the **dtw_timing_analysis.tcl** script timing report recommendations. For more information, refer to the *DDR Timing Wizard User Guide*.

Table 4. Checklist for Implementing QDRII+/QDRII SRAM Memory Interfaces in Stratix II and Stratix II GX Devices (Part 3 of 3)

Item	Description
Board-Level Considerations	
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. Stratix II and Stratix II GX devices support on-chip termination (OCT). On-chip termination includes series OCT with and without calibration and parallel OCT with calibration. On-chip series termination with and without calibration is supported on output pins or on the output function of bidirectional pins. On-chip parallel termination with calibration is only supported for input pins. Pins configured as bidirectional do not support on-chip parallel termination. Therefore, bidirectional signals such as CQ and CQn need to be terminated on the board by using parallel resistors. If there are multiple loads on certain FPGA output pins (for example, if the address bus is shared across many memory devices), the maximum drive strength setting may be preferred over the series OCT setting.  For more information, refer to *Section IV. I/O Standards* in the *Stratix II GX Device Handbook*. Single-ended, non-voltage-referenced I/O standards do not require termination. Altera recommends using external termination to improve signal integrity where required. Voltage-referenced I/O standards require both an input reference voltage (V_{REF}), and a termination voltage, V_{TT}. Use off-chip termination on the board for series and parallel termination.  For more information, refer to the *I/O Standards* section of the *Stratix II GX Device Handbook*. On the memory side, Altera recommends using external parallel termination on input signals to the memory (write data, address, command, and clock signals). Use board-level simulations to pick the optimal setting for best signal integrity.
10	• Have you performed board-level simulations to ensure electrical and timing margins for your memory interface? Ensure that you have a sufficient eye opening using simulations. 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.

Design Example for a QDRII SRAM Interface in a Stratix II Device

This design example describes the steps necessary to create, constrain, and verify operation of a 200-MHz QDRII SRAM memory interface on a Stratix II device. This design example uses the QDRII SRAM Controller v7.2 SP1, targeting the Stratix II device.

This design example shows the memory interface design flow steps, shown in Figure 1 on page 4, and detailed in *AN 449: External Memory Interface Design Guidelines for Stratix II, Stratix II GX, and Arria GX Devices*.

The design example was created using Quartus II software v7.2 SP1 and MegaCore IP Library software version 7.2 SP1, and targets an Altera internal board (called S2MB2), which features the EP2S60F1020C5ES device. The bottom banks of the Stratix II device interface to QDRII SRAM with a bus width of 18 and a burst length of four operates at 200 MHz (800 Mbps).

The board contains two QDRII SRAM devices:

- Cypress Semiconductor CY7C1313AV18-250BZC (used in this design example)
- Samsung Semiconductor K7R321884M-FC25000

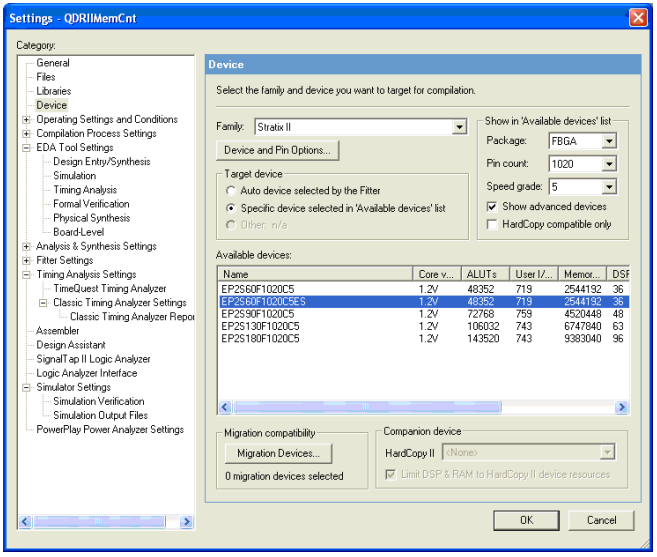
To begin, create a new Quartus II project or open an existing project in which you would like to implement the QDRII SRAM memory interface. When creating a new project, specify the target FPGA device (Stratix II).

Select the Device

1. On the **Assignments** menu, click **Device**.
2. To select a family, from the drop-down menu, click **Stratix II**.
3. Under **Available devices**, select **EP2S60F1020C5ES**. You can also filter the device list by setting the speed grade, pin count, and package of your device.
4. Click **OK**.

   Refer to Figure 3.

Figure 3. Select Device

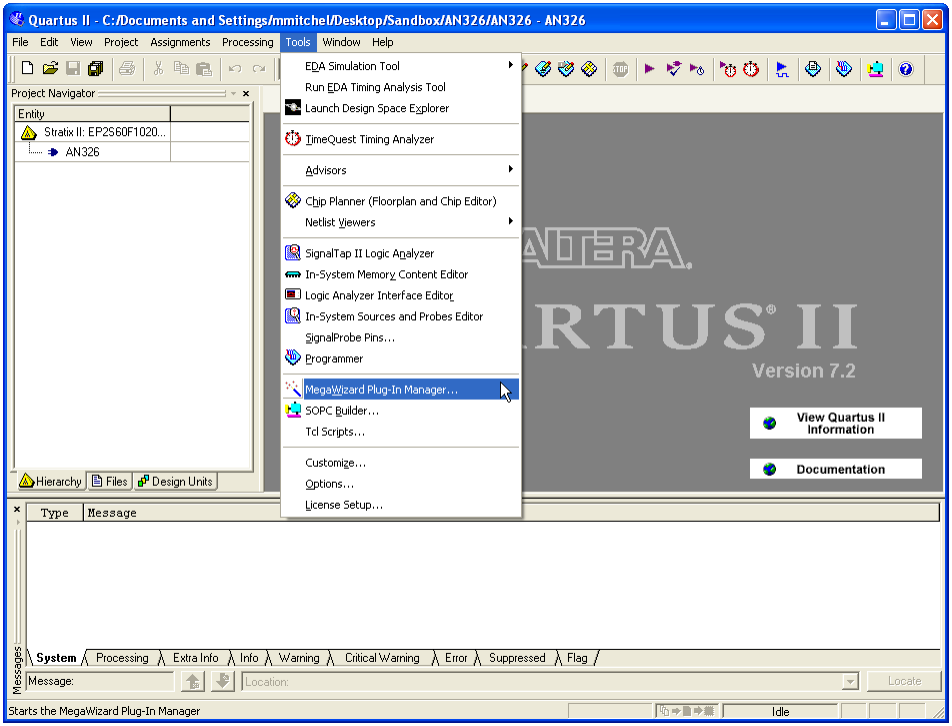


Instantiate the Memory Controller in a Quartus II Project

Begin your memory interface design by instantiating the memory controller modules.

1. On the **Tools** menu, click **MegaWizard Plug-In Manager** (Figure 4).

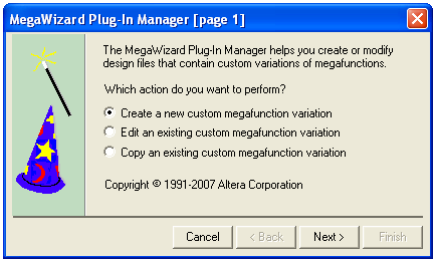
Figure 4. Tools Menu



2. In the **MegaWizard Plug-In Manager [page 1]** dialog box, select **Create a new custom megafunction variation** (Figure 5).

3. Click **Next**.

Figure 5. MegaWizard, Page 1

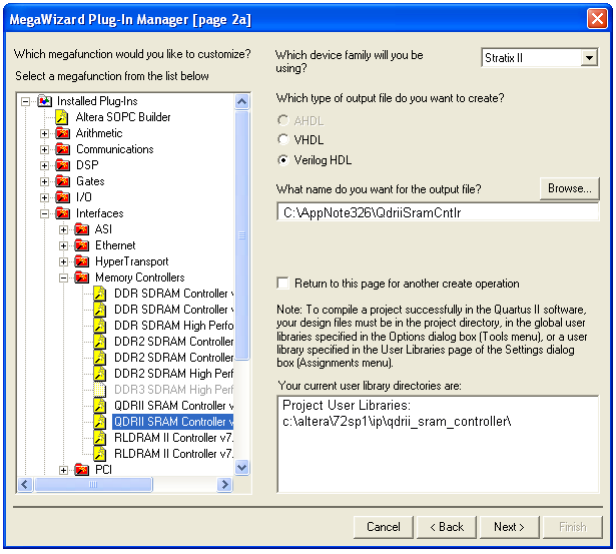


4. On page 2a, under **Select a megafunction from the list below**, expand the **Interfaces** folder and the **Memory Controllers** folder.

5. Click **QDRII SRAM Controller v7.2sp1** (Figure 6).

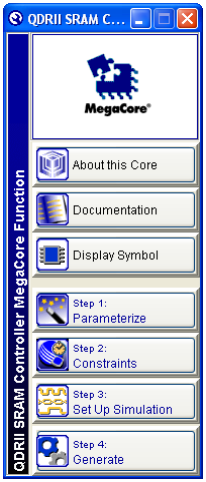
   In this example, it is called **QdriiSramCntlr**.

Figure 6. MegaWizard, Page 2a



6. Click **Next**. The Quartus II software then loads the QDRII SRAM Controller MegaWizard and the MegaCore window appears, as shown in Figure 7.

Figure 7. QDRII SRAM Controller MegaWizard

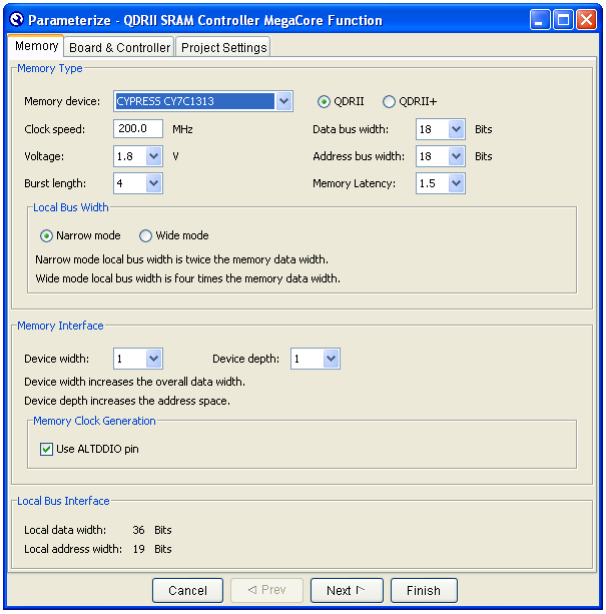


The first three boxes are informational boxes.

For more information, refer to the *QDRII SRAM Controller Megacore Function User Guide* as you are going through IP creation. To access information about and location of the latest version of the user guide, on the MegaCore QDRII SRAM Controller MegaWizard, click **Documentation**.

7. Click **Step 1: Parameterize**. The Memory tab of the **Parameterize** dialog box appears (Figure 8).

Figure 8. Parameterize: Memory Settings



The **Memory** tab of this dialog box allows you to set the memory parameters.

8. To select your desired memory device, from the drop-down menu, click your device. You can also select **Default**. In this example, the **CYPRESS CY7C1313** memory device is selected.

   Selecting **Default** automatically fills many fields to the correct value, per the memory device data sheet specification. If you change the value of any of the automatically-set parameters, the memory device automatically changes to **Custom**.

Be sure to check the values filled in each field of the MegaWizard pages to ensure that they are correct.

If your memory device is not in the list, select **Custom**, and select and adjust the rest of the fields on this tab of the dialog box according to your memory device specifications. Refer to Table 5 for an explanation of the available parameters.

Table 5. Memory Parameters

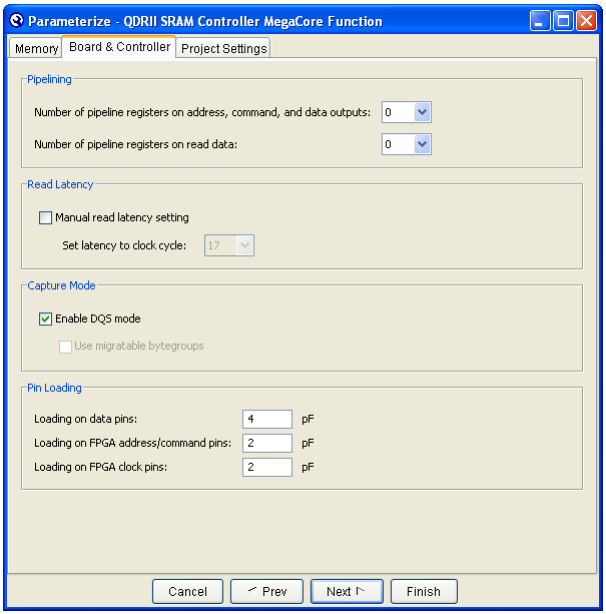
Field	Function
Data bus width	This indicates the width of the QDR SRAM device. The width can be anything up to what the device supports (for example, you can make a 72-bit interface out of four 18-bit interfaces).
Address bus width	This indicates the memory space, according to the memory device data sheet.
Local Bus Width	With a burst length of four, there is the option of narrow mode or wide mode.The narrow mode local bus width is twice the memory data width. The wide mode local bus width is four times the memory data width. • For write into and read from consecutive address, use wide mode. • For non-consecutive addresses, use narrow mode.
Memory Interface	The device width specifies the number of devices to increase the width of the data bus. If more than one memory device is selected, consider that the QDRII+/QDRII SRAM Controller always generates one address bus per memory device. If the address buses are shared, the loading on the address bus could be high and could result in timing violations. You are given an option to have one address bus per memory device. It is up to you whether to use them or not. If the you choose not to use the additional address bus, you can leave the second bus unconnected (the Quartus II software optimizes it away). You could instead remove the second bus from the clear text path. You can set the device depth `1` or `2`. Choose `2` (the maximum) if you want to double the memory space.
Memory Clock Generation	When **Use ALTDDIO pin** is toggled on, the ALTDDIO megafunction generates a repetitive 1-0-1-0 output to be used as your memory clock. This ensures the clock tracks with the data outputs (unless you target HardCopy® II devices). If you target HardCopy II devices, Altera recommends using dedicated PLL outputs for your memory clock to ensure best noise rejection, which requires that you turn off the **Use ALTDDIO pin** option.

You can add your own devices to the list by editing the **memory_types.dat** file in the **C:\altera\<*quartus II version folder*>\ip\qdrii_sram_controller\constraints** directory. In this example, the path is **c:\altera\v72sp1\ip\qdrii_sram_controller\constraints**.

The first time you use the IP QDRII SRAM Controller MegaWizard, the selected clock frequency for the IP QDRII SRAM Controller MegaWizard-generated PLL's input and output clocks is the default. You can set the clock speed up to 300 MHz. The QDRII SRAM Controller MegaWizard allows you to enter up to 600 MHz, but Altera only supports the QDRII SRAM controller up to 300 MHz in Stratix II and Stratix II GX devices.

9. Click **Next**. The **Board & Controller** tab of the **Parameterize** dialog box (Figure 9) appears.

Figure 9. Parameterize: Board and Controller Settings



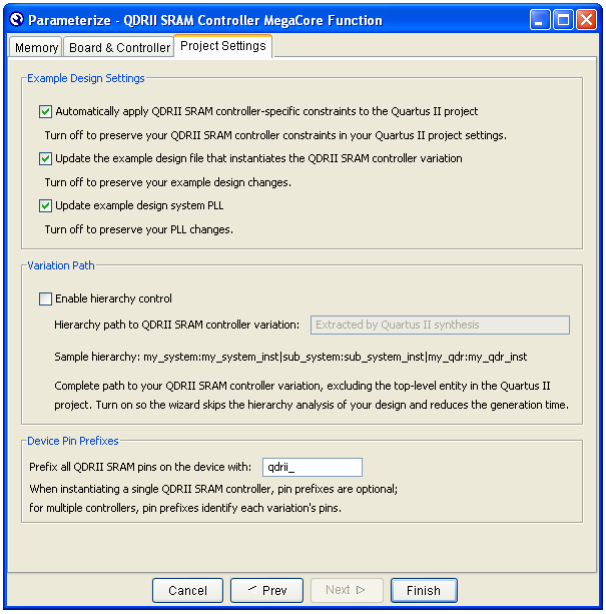
10. Adjust the fields on this tab of the dialog box according to your memory device specifications. Refer to Table 6 for an explanation of the available parameters on this tab.

Table 6. Board and Controller Parameters

Field	Description
Pipelining	You can choose 0, 1, 2, or 3 pipeline registers between the memory controller and the address, command, data outputs, and read data input. These registers help to achieve the required performance at higher frequencies. To reduce latency throughout the design, remove the pipelines.
Read Latency	The default value of 17 is the Altera-recommended read latency value. Although you have the choice to set read latency manually to any value from 15 to 21, Altera strongly recommends no values fewer than 17. This means that it takes 17 clock cycles to get the read data at the local interface. This latency includes the controller and command latency plus CAS latency, plus controller read data input latency.
Capture Mode	Enable DQS mode for Stratix II and Stratix II GX devices. (This example design is only for Stratix II.) When DQS mode is disabled, you can turn on **Use migratable bytegroups** to migrate the design to a migration device. This option is available for Stratix II devices only. When turned off, the MegaWizard allows much greater flexibility in the placement of byte groups.
Pin Loading	Enter the loading that matches your board and memory device. If you are double loading the clock pins you need to double up the capacitance. Note that the loading on memory interface pins is dependent on your board topology.

11. Click **Next**. The **Project Settings** tab of the **Parameterize** dialog box (Figure 10) appears.

Figure 10. Parameterize: Project Settings



12. Adjust the fields on this tab of the dialog box according to your specifications. Refer to Table 7 for an explanation of the available parameters on this tab

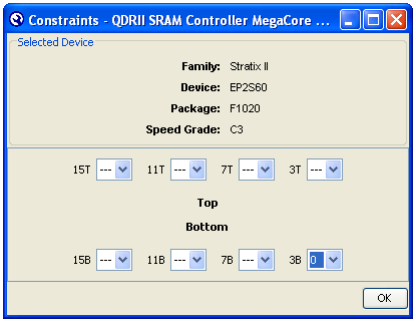
.

Table 7. Project Settings Parameters	
Field	**Description**
Example Design Settings	
Automatically apply QDRII SRAM controller-specific constraints to the Quartus II project	When this option is checked, the next time you compile after you generate the controller, the Quartus II software automatically runs the add constraints script.
Update the example design file that instantiates the QDRII SRAM controller variation	When this option is turned on, the IP QDRII SRAM Controller MegaWizard parses and updates the example design file. It only updates sections that are between the following markers: <<START MEGAWIZARD INSERT <*tagname*> <<END MEGAWIZARD INSERT <*tagname*> If you edit the example design file, ensure that your changes are outside of the markers or remove the markers. Once you remove the markers, you must keep the file updated, because the IP QDRII SRAM Controller MegaWizard can no longer update the file.
Update example design system PLL	When this option is checked, the IP QDRII SRAM Controller MegaWizard automatically overwrites the PLL. The first time you create a custom variation, this option must be enabled, then turn it off for subsequent generations if you don't want to overwrite the PLL.
Enable hierarchy control	The constraints script analyzes your design to automatically extract the hierarchy to your variation. To prevent the constraints script analyzing your design, turn on **Enable hierarchy control**, and enter the correct hierarchy path to your controller.
Hierarchy path to variation	The hierarchy path is the path to your QDRII SRAM controller, minus the top-level name. The hierarchy entered in the wizard must match your design, because the constraints scripts rely on this path for correct operation.
Device Pin Prefixes	Filling in this parameter adds a prefix to the pin names for the FPGA pins that are connected to the QDRII SRAM controller.

13. Click **Finish**.

14. On the QDRII SRAM Controller MegaWizard page, click **Step 2: Constraints**. The **Constraints** dialog box appears (Figure 11).

Figure 11. Constraints Dialog Box



15. Choose the positions on the device for each of the QDRII SRAM byte groups. A byte group consists of a CQ pin and a number of Q pins (the same number as the data width). With more than one device, one CQ/CQn pair and q bus are connected per device in the width direction.

    To place a byte group, in each position, select the byte group from the drop-down list at your chosen position as shown below. The floorplan in Figure 11 matches the orientation of the Quartus II floorplanner. The layout represents the Q.

    CAUTION

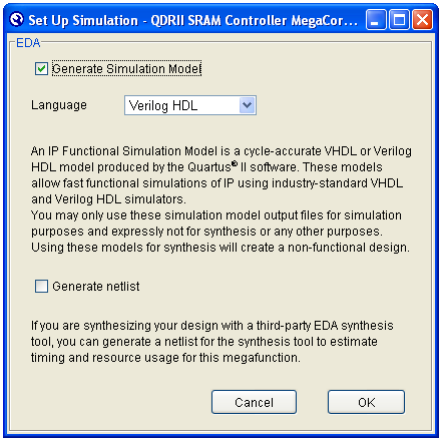
    Ensure that what you select matches the board connections between the FPGA and SRAM devices.

    In this example, **3B** is selected. By selecting **3B**, I/O bank 7 of the Stratix II device will be populated by CQ and Q pins. (Refer to the pin planner, Figure 34 on page 43. Pin #1 is at the top left, and bank 7 is the bottom left group of pins.) The reason **3B** is selected here is that the QDRII SRAM device is connected to the Stratix II device by the I/Os in bank 7.

16. Click **OK**.

17. Click **Step 3: Set Up Simulation**. The **Set Up Simulation** dialog box appears (Figure 12).

Figure 12. Set Up Simulation Dialog Box

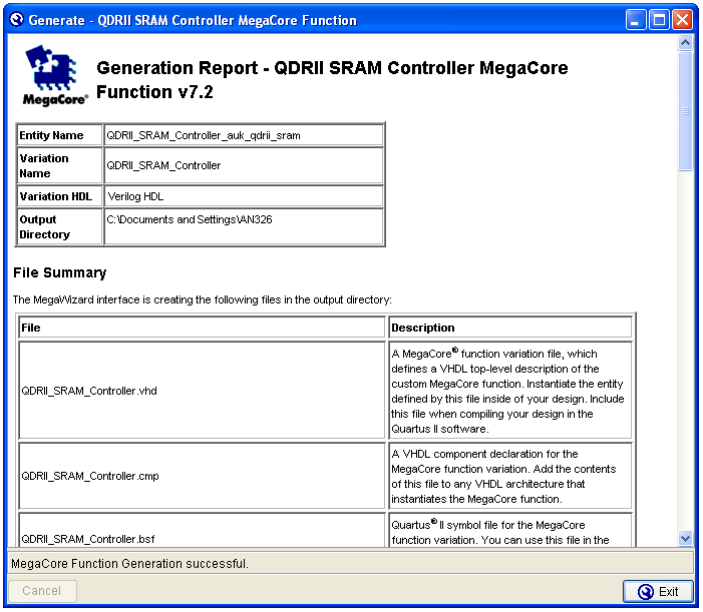


18. Turn on **Generate Simulation Model** and choose the language of preference. In this design example, **VHDL** is chosen.

19. Click **OK**.

20. Click **Step 4: Generate**. This generates all of the necessary files and lists them in the **Generate - QDRII SRAM Controller MegaCore Function** window (Figure 13).

Figure 13. Generation Report—QDRII SRAM Controller MegaCore Function



A description of all generated files appears in the *QDRII SRAM Controller MegaCore Function User Guide*.

21. When **MegaCore Function Generation successful** appears at the bottom of the window, click **Exit**.

RTL Functional Simulation

The next step is to simulate your design, using the ModelSim-Altera software.

For more information about how to use the ModelSim-Altera software, refer to the ModelSim-Altera Software Support page of the Altera website.

1. In your project directory, navigate to the **testbench** folder and click **qdr_model.v** to open the ModelSim-Altera software.

2. When ModelSim is opened, you should be in the **testbench** directory. To change the directory to **modelsim**, at the prompt in the Tcl Transcript window, type:
   `cd modelsim` ↵

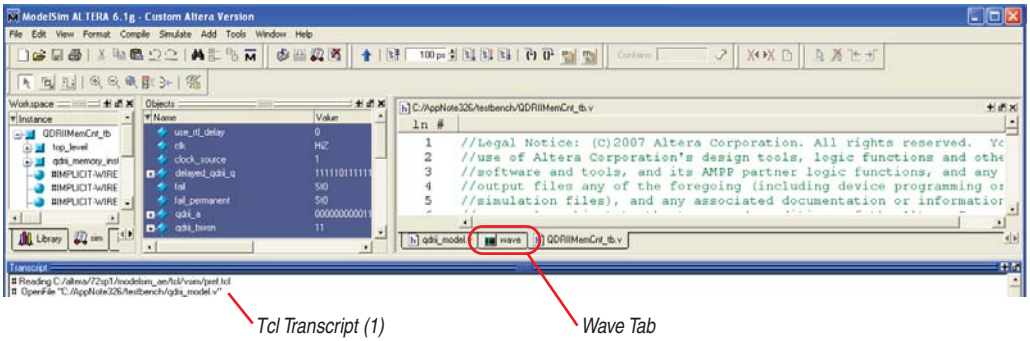
3. To run the simulation, in the Tcl window, at the prompt, type: `source <variation_name>_vsim.tcl` ↵

   This command runs a script that compiles all of the files including the testbench in a batch mode.

4. When the script is complete, the **Finish Vsim** window appears, asking if you are sure you want to finish. If you click **Yes**, ModelSim closes. If, however, you want to look at your waveforms, click **No**.

5. To see the signals, click the **Wave** tab (Figure 14).

Figure 14. Waveform Tab *Note (1)*



Note to Figure 14:
(1) The Tcl transcript appears in detail in Example 1.

Example 1. Tcl Transcript

```
# Reading C:/altera/72sp1/modelsim_ae/tcl/vsim/pref.tcl
#  OpenFile "C:/AppNote326/testbench/qdrii_model.v"
ModelSim> pwd
# C:/AppNote326/testbench
ModelSim> cd modelsim
# reading modelsim.ini
ModelSim> pwd
# C:/AppNote326/testbench/modelsim
ModelSim> dir
#  Volume in drive C has no label.
#  Volume Serial Number is 5057-7CD7
#
#  Directory of C:\AppNote326\testbench\modelsim
#
# 12/20/2007  11:32 AM    <DIR>          .
# 12/20/2007  11:32 AM    <DIR>          ..
# 12/20/2007  11:32 AM    <DIR>          auk_qdrii_lib
# 12/20/2007  11:32 AM             6,284 modelsim.ini
# 10/31/2007  01:46 PM             8,504 QdriiSramCntrl_vsim.tcl
# 12/20/2007  11:31 AM            11,280 tcl_stacktrace.txt
# 12/21/2007  12:49 PM           327,680 vsim.wlf
# 12/20/2007  11:15 AM             5,988 wave.do
#                5 File(s)        359,736 bytes
#                3 Dir(s)  79,868,579,840 bytes free
ModelSim> source QdriiSramCntrl_vsim.tcl
# eval vlog -work auk_qdrii_lib ../qdrii_model.v
# Model Technology ModelSim ALTERA vlog 6.1g Compiler 2006.08 Aug 12 2006
# -- Compiling module qdrii_model
```

6. On the **View** menu, point to **Wave**, then **Zoom**, then click **Zoom In**. Alternately, you can click the Zoom icon ( ).

7. Using the Zoom-In marker ( ), drag to the section of the waveform that you want to see in greater detail (Figure 15). Note the time division indicator at the top of the window, which indicates how zoomed-in you are viewing the waveform. To view the entire waveform, click the filled-in Zoom icon ( ).

Figure 15. Waveform

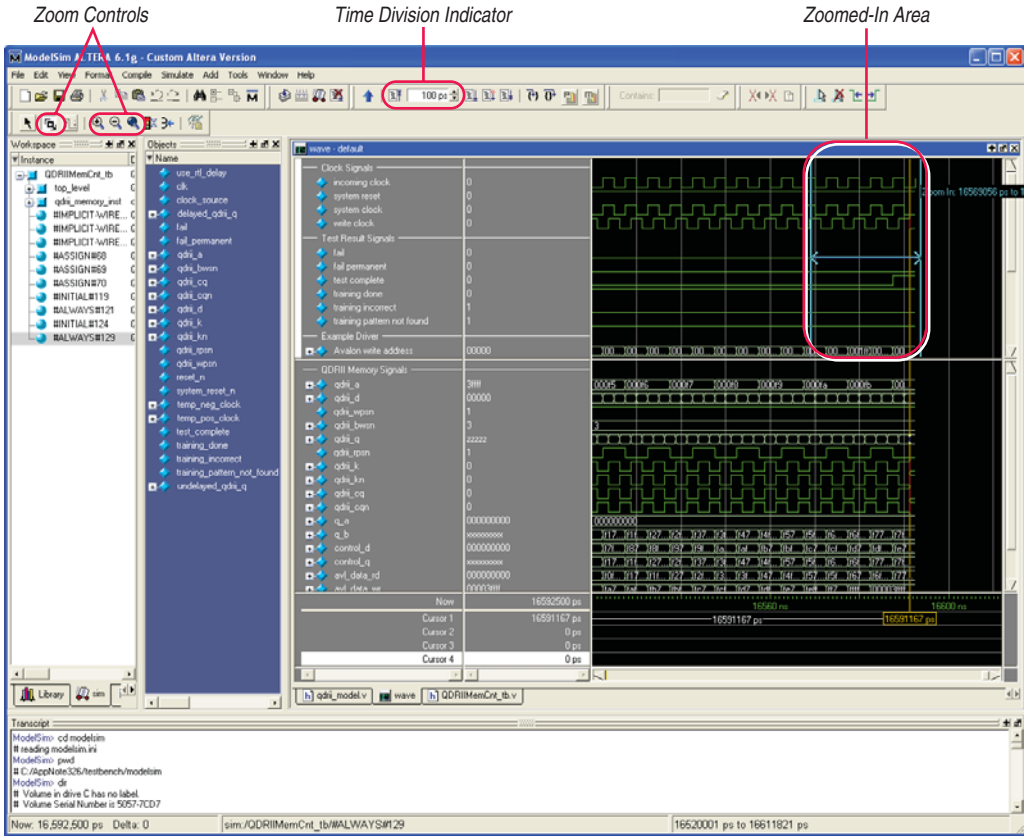
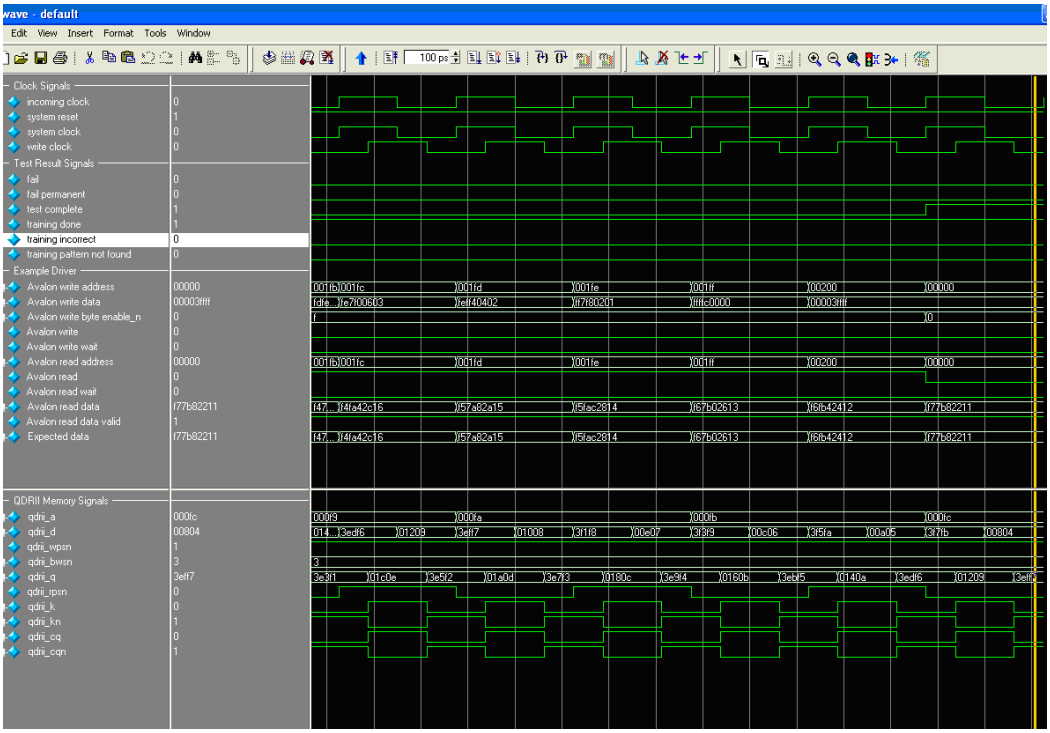


Figure 16 shows 1500 ps zoomed into the very end of the waveform. As you can see at the end, the test results signals should show completion of the test with no failure.

Figure 16. Waveform, Zoom



The system clock triggers the QDRII SRAM signals, `qdrii_a` (address), `qdrii_wpsn` (write port select), and `qdrii_rpsn` (read port select).

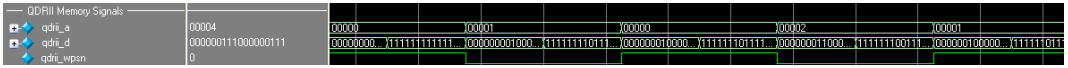
`Write` is performed as the write is requested by triggering **Avalon write** (`avl_write`) from low to high. To start a write, **Avalon write wait** (`avl_write_wait`) must be low.

When `qdrii_wpsn` goes low, the write into the FPGA is permitted and **Avalon write data** (`avl_write_data`) is written in the address `qdrii_a` on the rising and falling edges of the system clock. This is because the Avalon write data bus is two times wider than the qdrii data bus.

The write is shown in Figures 17 and 18. Figure 17 shows the Avalon side of the write; Figure 18 shows the QDRII SRAM side.

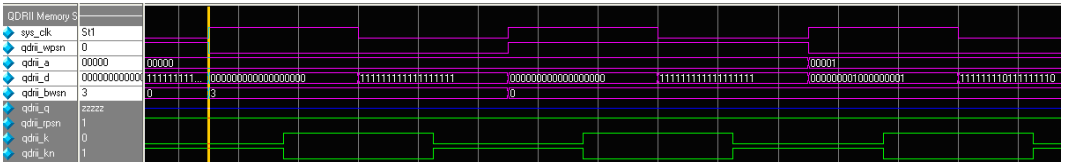
Figure 17. Avalon Waveform

Figure 18. QDRII Waveform



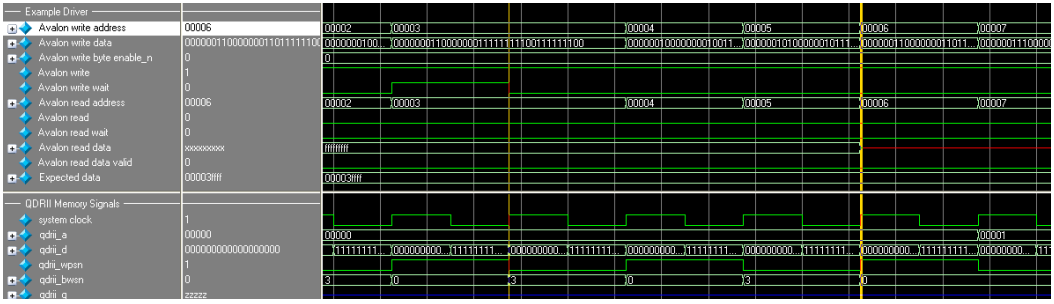
If write is performed in consecutive addresses, as long as `qdrii_bwsn` is low, the data is written on the positive and negative edges of the system clock when `qdrii_wspn` is low (Figure 19).

Figure 19. Write Into Consecutive Memory Addresses



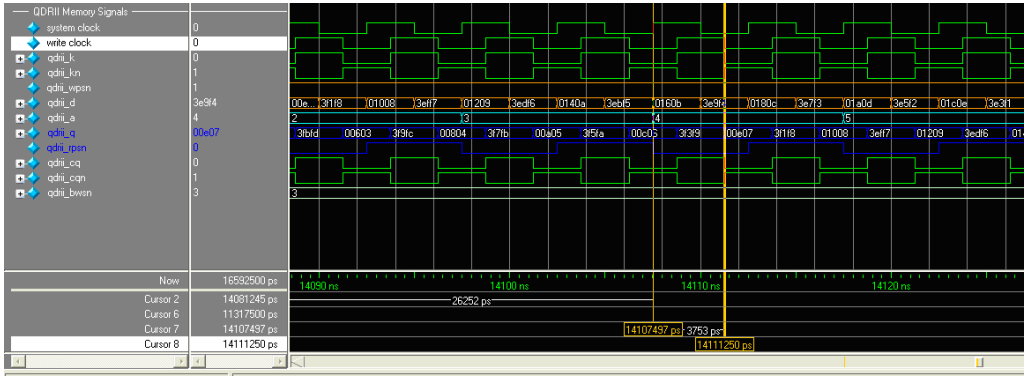
The write latency—the number of cycles it takes to present the data at the memory interface after a write request—is shown in Figure 20. It takes three clock cycles from the time that **Avalon write wait** (`avl_write_wait`) goes low until the data is available on the `qdrii_d` bus.

Figure 20. Write Latency Waveform



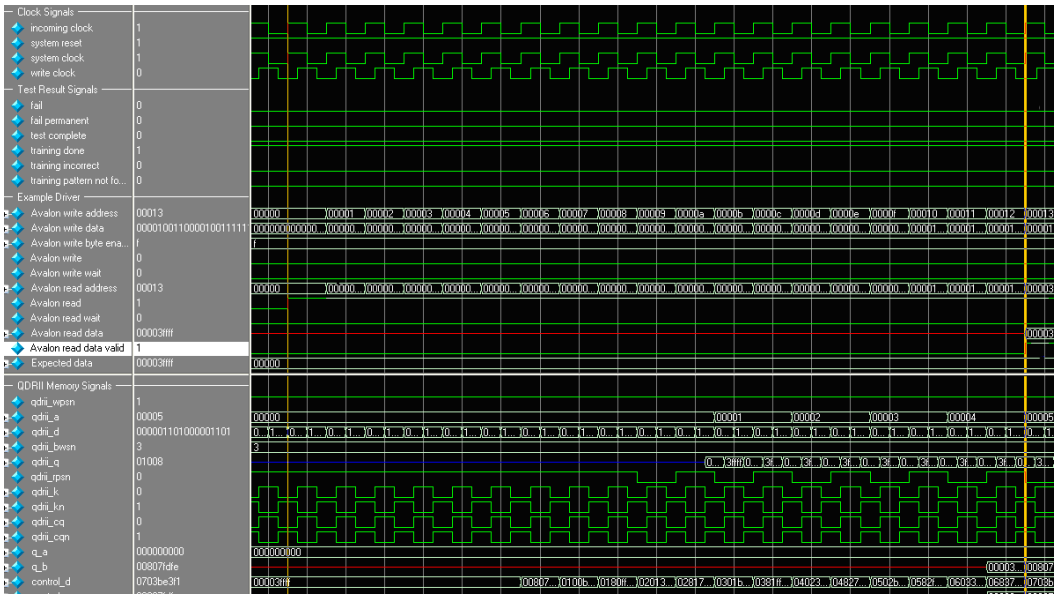
A read is performed when **Avalon read** (`avl_read`) goes high, and on the rising edge of the system clock, `qdrii_rpsn` goes low. Then, on the following falling edge of `qdrii_cq`, data is available on the `qdrii_q` bus (Figure 21).

Figure 21. Read Waveform



Read latency—the number of cycles it takes to get the data at the local interface after a read request— is shown in Figure 22. It takes 19 clock cycles from the time **Avalon read** is triggered high to the time that the data is available on Avalon read data bus.

Figure 22. Read Latency Waveform



Add Constraints

To adjust and add new constraints, follow these steps:

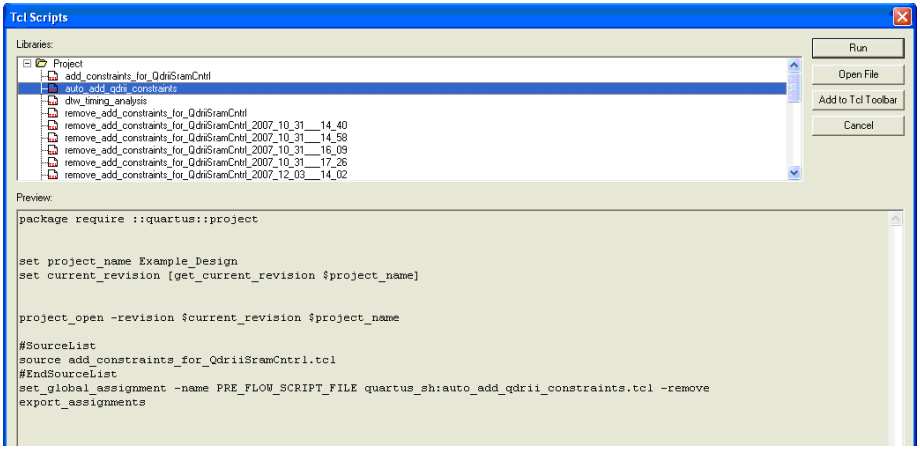
Run Constraints

Open the Tcl Script window. To do this, on the **Tools** menu, click **Tcl Scripts**. The various scripts that apply to your design are shown (Figure 23).

To preview any of the listed scripts, click the script and view the text in the **Preview** section of the window. Click **auto_add_qdrii_constraints.tcl** and click **Run**. This begins Analysis and Elaboration.

The **auto_add_qdrii_constraints.tcl** script calls the **add_constraints_for_<***variation name***>.tcl** script for each variation in your design. The **add_constraints_for_<***variation name***>.tcl** script checks for any previously added constraints, removes them, and then adds constraints for that variation. You therefore must run this script after making any changes to your design.

Figure 23. Tcl Scripts



When the constraints script runs, it creates another script called **remove_constraints_for_<*variation name*>.tcl**, which you can use to remove the constraints from your design.

The constraints script implements the following types of assignments:

- CQ, CQn, and Q capture pins placement
- Capacitance loading
- CQ pin set to global signal
- I/O type for all interface pins
- Cut timing assignments for false timing paths, such as `q_captured_rising*` and `q_captured_falling*`

By running **remove_constraints_for_**<*variation_name*>**.tcl**, you can disable the constraints above and run your design under different timing numbers.

Run DTW

This step is optional but recommended.

DTW is a Tcl-based GUI that calculates timing constraints based on the FPGA and memory device chosen. To use DTW, you must enter the memory device parameters and your board information correctly in the QDRII SRAM Controller MegaWizard. The Quartus II fitter uses timing-driven compilation to route the design to meet the timing constraints set by the DTW.

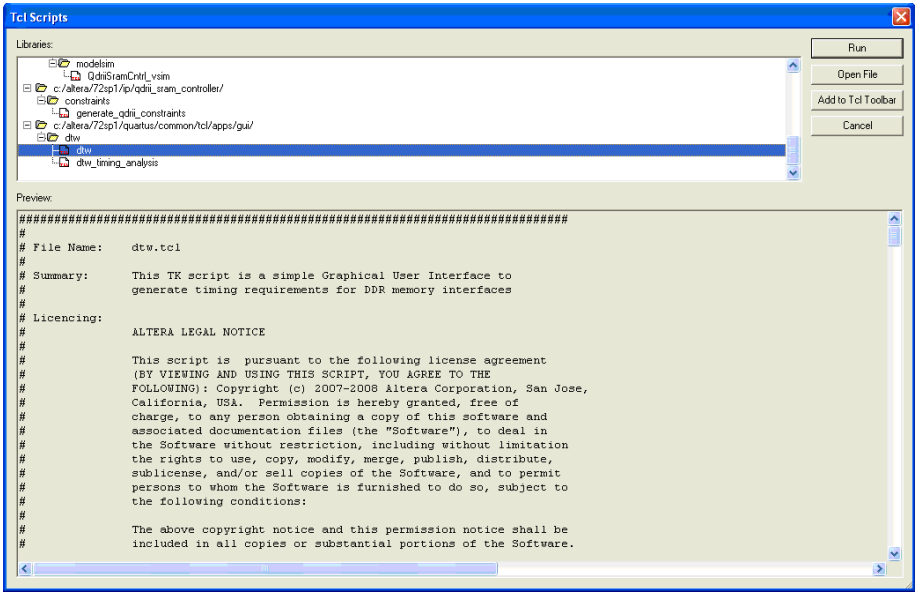
DTW simplifies the process of constraining your design by using timing assignments, which the Quartus II software uses to place and route the design in the target device. These timing constraints are applicable for FPGAs and their HardCopy-equivalent devices, eliminating the need to do conversion (as you would have previously done with the placement constraints from the QDRII SRAM Controller MegaWizard).

For more information about DTW, refer to the *DDR Timing Wizard User Guide*.

To run the DTW tcl script, perform the following steps:

1. On the **Tools** menu, click **Tcl Scripts**. The **Tcl Scripts** window appears.
2. Click **DTW** to run the DDR Timing Wizard, as shown in Figure 24.

Figure 24. DDR Timing Wizard



3. Click **Run**.

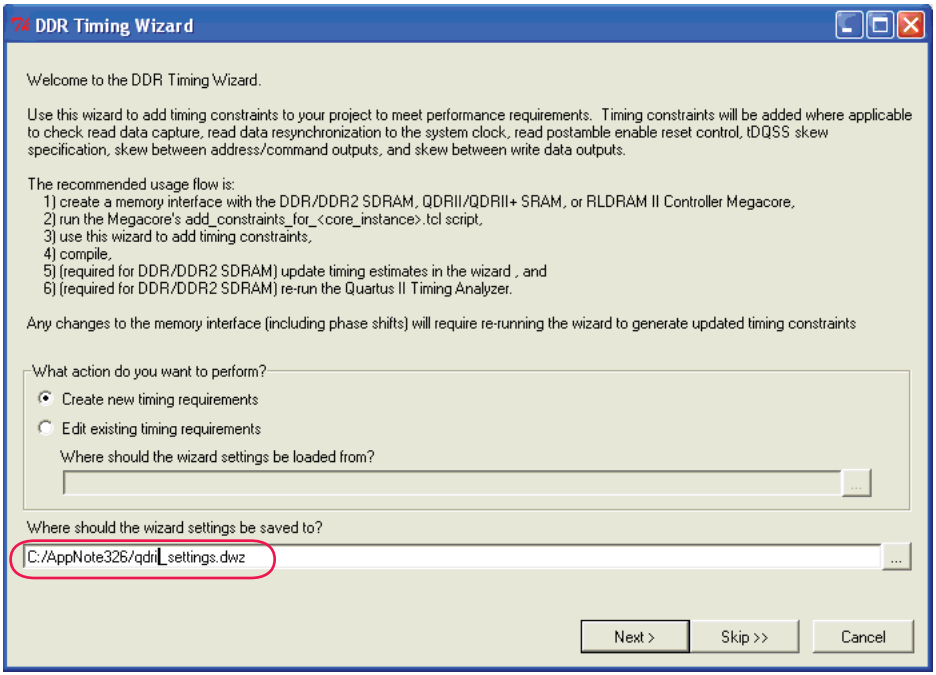
The first time you open this wizard for your project, you must create new timing requirements, as shown in Figure 25.

To do this, follow these steps:

a. Double-check that you are in the right directory when creating the **.dwz** file (refer to the circled path in Figure 25).

b. Click **Next**.

Figure 25. DDR TIming Wizard

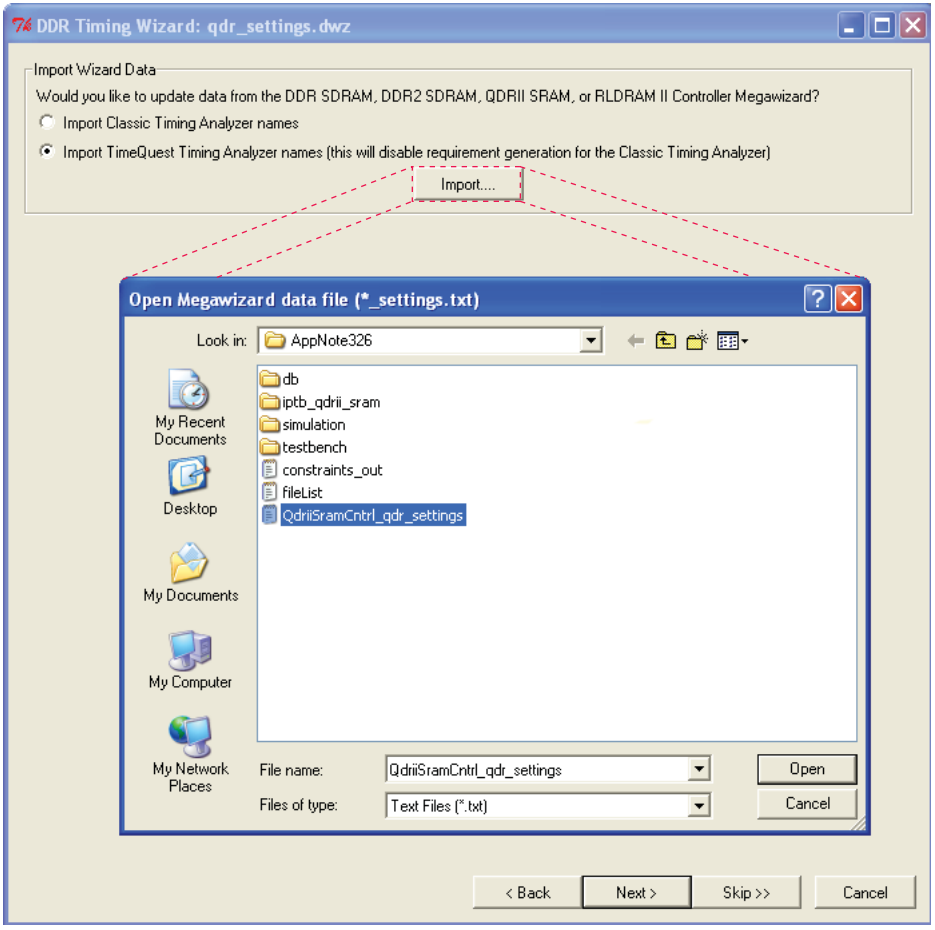


4. To import your settings, click **Import**. The **Open Megawizard data file** dialog box appears.

5. Choose **Import Classic Timing Analyzer names** or **Import TimeQuest Timing Analyzer names**. (Altera recommends using the **TimeQuest Timing Analyzer** option.)

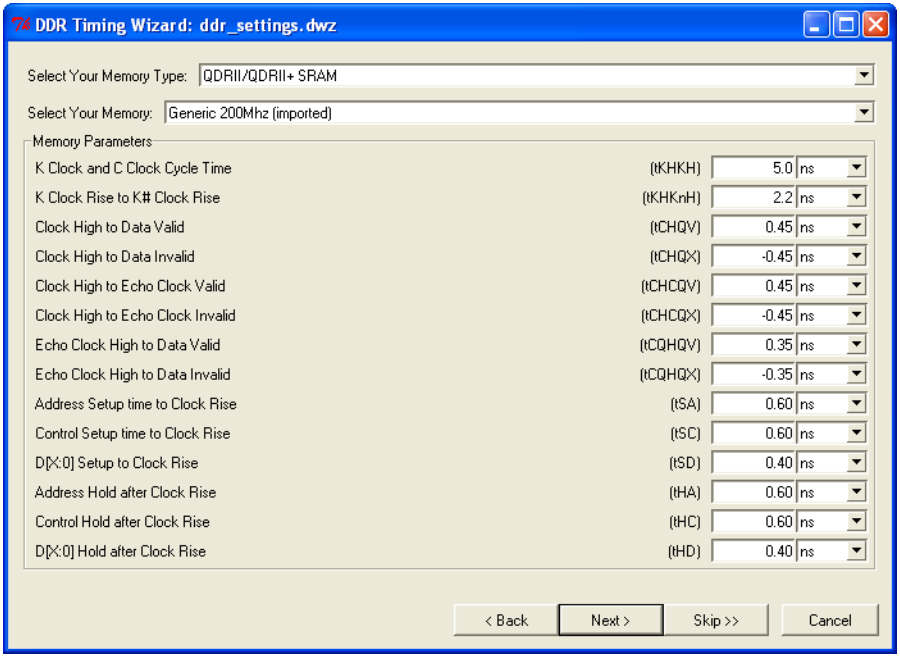
6. Double-click **<*variation_name*>_qdr_settings.txt** (Figure 26).

Figure 26. Open Megawizard Data File



7. When file import is complete, click **Next**. The **DDR Timing Wizard: qdr_settings.dwz** dialog box appears (Figure 27).

***Figure 27. QDRII Settings** Note (1)*



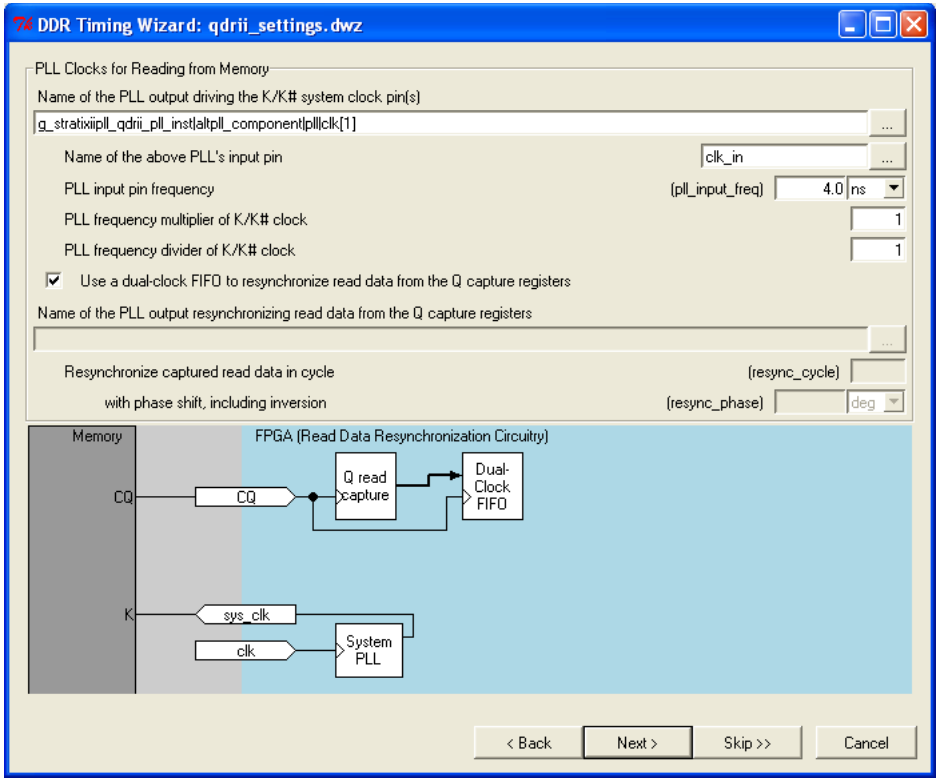
Note to Figure 27:

(1) By importing your QDRII settings from the MegaWizard file, all fields should be filled. Ensure that the parameters are correct before proceeding.

8. From the first drop-down menu in Figure 27, select your memory type.
9. From the second drop-down menu in Figure 27, select your memory.
    a. If your memory device is listed in the drop-down menu, select it. The Memory Parameters are imported automatically. Double-check that these parameters are correct.
    b. If your memory device is not listed in the drop-down menu, select **Custom** and add the Memory Parameters manually.
10. Click **Next**.

The next series of dialog boxes require you to make various settings. By importing the QDRII settings from the MegaWizard file, these fields should be filled automatically. Figures 28, 29, and 30 show some of the fields that must be assigned.

Figure 28. PLL Clocks for Reading from Memory *Note (1)*



Note to Figure 28:

(1) By importing your QDRII settings from the MegaWizard file, all fields should be filled. Ensure that the parameters are correct before proceeding.

Figure 29. Board Parameters

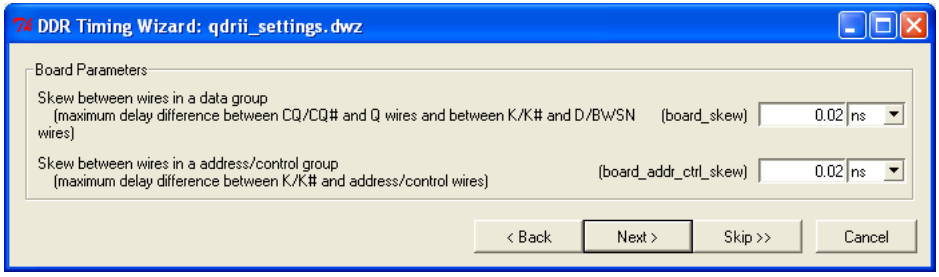
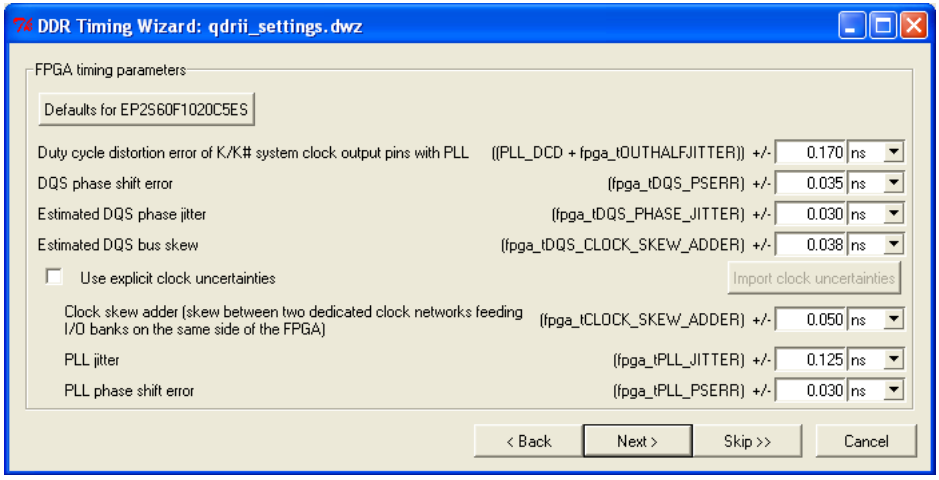


Figure 30. FPGA Timing Parameters

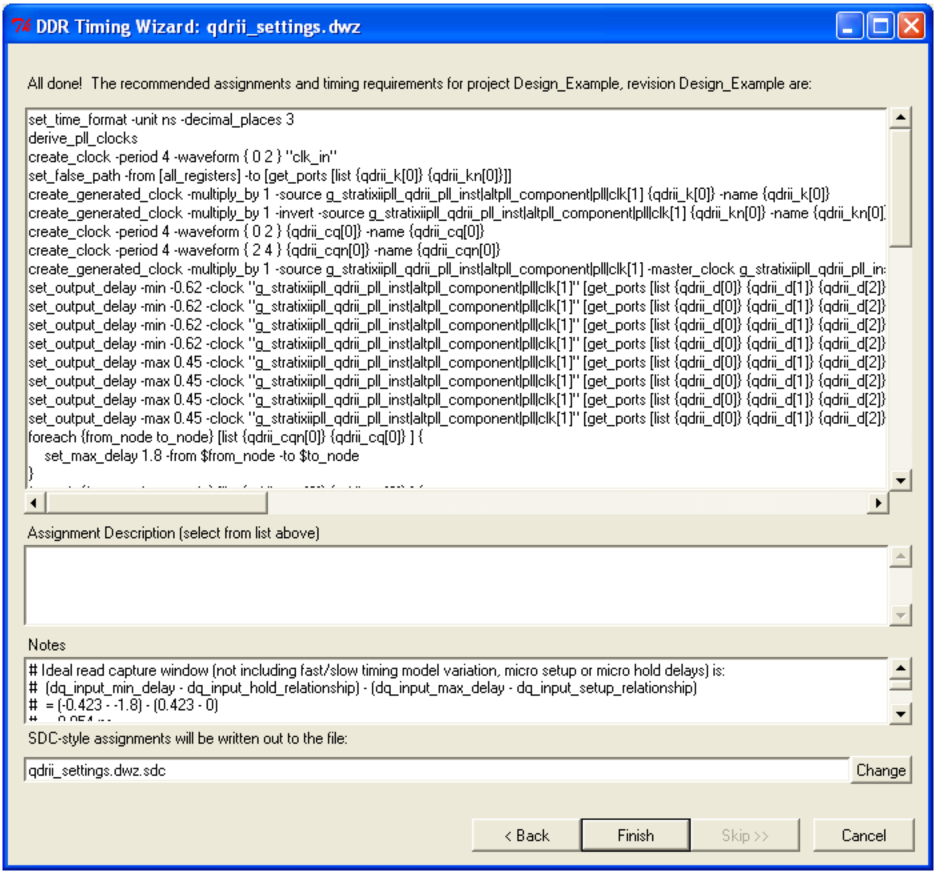


Note to Figure 30:

(1) FPGA timing parameters are automatically filled based on the default numbers for the target device. These timing parameters are hard-coded in the MegaWizard.

11. The final window to appear is shown in Figure 31.

Figure 31. DDR Timing Wizard Finish Window

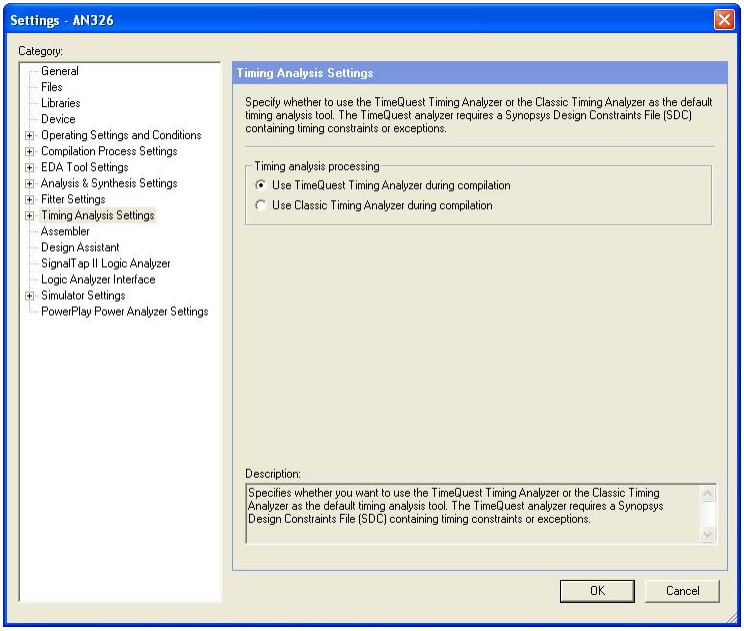


12. Click **Finish**.

Set the TimeQuest Timing Analyzer

1. On the Assignments menu, click **Timing Analysis Settings**.
2. In the **Timing Analysis Settings** dialog box, select **Use TimeQuest Timing Analyzer during compilation** (Figure 32). (The default is to use the Classic Timing Analyzer.)
3. Click **OK**.

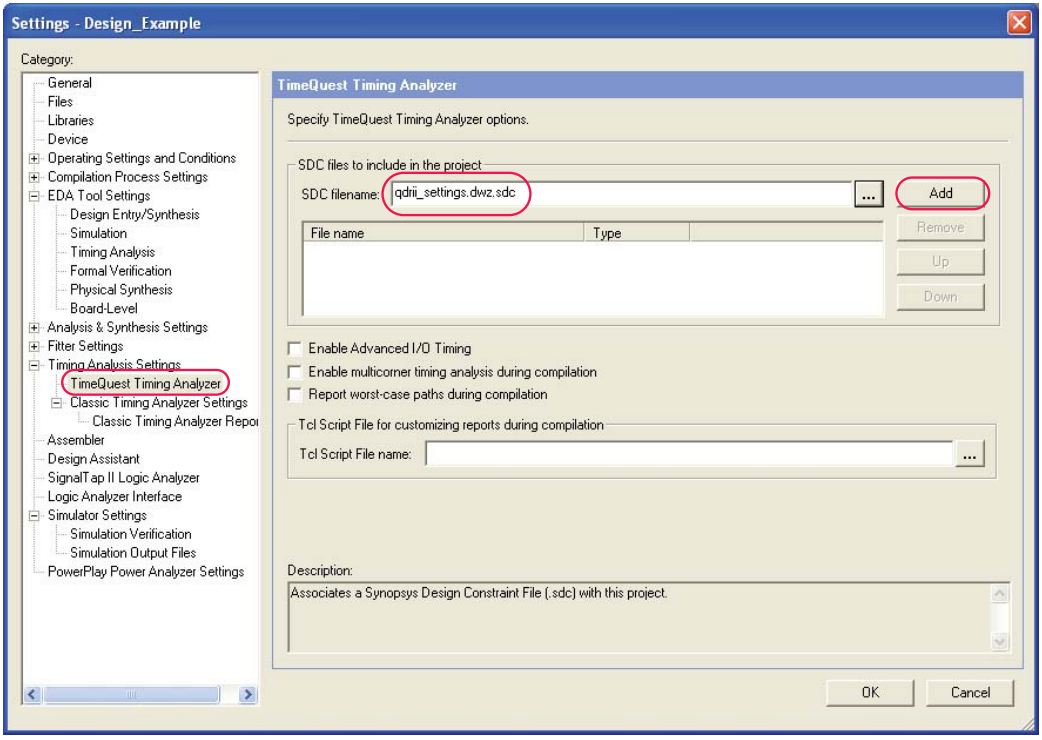
Figure 32. TimeQuest Timing Analyzer



Add the SDC File

1. On the **Assignments** menu, click **Timing Analysis Settings**.
2. Under **Category**, expand **Timing Analysis Settings**.
3. Click **TimeQuest Timing Analyzer**.
4. Under **SDC file to include in the project**, click **<*variation_name*>_settings.dwz.sdc** (Figure 33). This file contains all of the necessary timing constraints for the memory interface portion of your design.

Figure 33. TimeQuest Timing Analyzer Settings



5. Click **Add**.

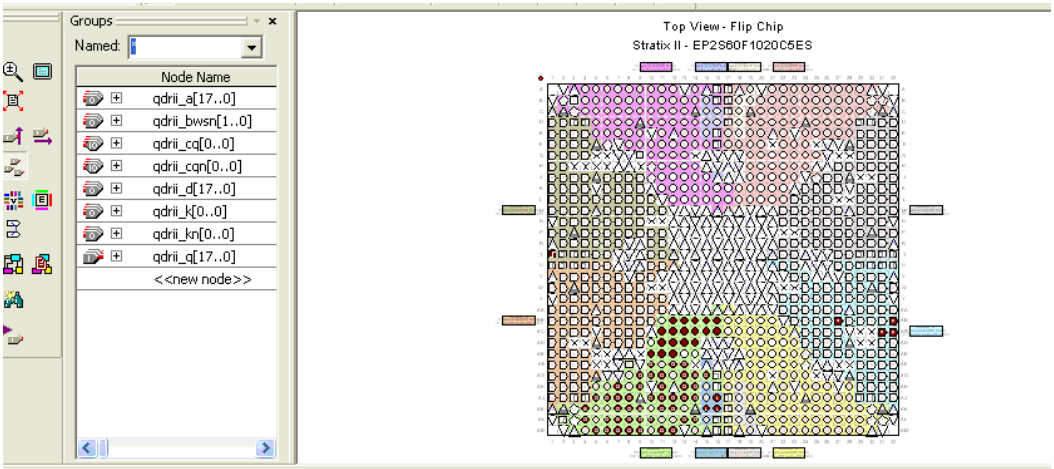
6. Click **OK**.

Assign Pin Locations and I/O Standards

Assign pin locations and I/O standards to your pins. Make sure the bank you are choosing is the right bank connected to your memory device on the board. In the design walkthrough example, bank 7 is selected because this was the bank chosen for the target board we are using.

To view the pin assignments, on the **Assignments** menu, click **Pins** (Figure 34).

Figure 34. Pin Assignments



	Node Name	Direction	Location	I/O Bank	Vref Group	I/O Standard	R
1	clk_in	Input	PIN_T1	5	B5_N0	3.3-V LVTTL (default)	
2	fail	Output	PIN_AC32	1	B1_N1	3.3-V LVTTL (default)	
3	fail_permanent	Output	PIN_AK15	10	B7_N0	3.3-V LVTTL (default)	
4	qdrii_a[17]	Output	PIN_AE12	7	B7_N1	1.8-V HSTL Class I	
5	qdrii_a[16]	Output	PIN_AE11	7	B7_N2	1.8-V HSTL Class I	
6	qdrii_a[15]	Output	PIN_AE10	7	B7_N2	1.8-V HSTL Class I	
7	qdrii_a[14]	Output	PIN_AD14	7	B7_N0	1.8-V HSTL Class I	
8	qdrii_a[13]	Output	PIN_AD13	7	B7_N1	1.8-V HSTL Class I	
9	qdrii_a[12]	Output	PIN_AD12	7	B7_N1	1.8-V HSTL Class I	
10	qdrii_a[11]	Output	PIN_AD11	7	B7_N2	1.8-V HSTL Class I	
11	qdrii_a[10]	Output	PIN_AC16	7	B7_N0	1.8-V HSTL Class I	
12	qdrii_a[9]	Output	PIN_AC15	7	B7_N0	1.8-V HSTL Class I	
13	qdrii_a[8]	Output	PIN_AC14	7	B7_N0	1.8-V HSTL Class I	
14	qdrii_a[7]	Output	PIN_AC13	7	B7_N1	1.8-V HSTL Class I	

When assigning pins, take into consideration that Stratix II and Stratix II GX devices offer DQ and DQS pins on all sides of the device. However, the top/bottom I/O banks support DLL-based higher performance interfaces. For interfaces running at lower speeds, a PLL-based implementation is supported on all Stratix II and Stratix II GX I/O banks.

For more information about PLL-based implementation, refer to "Appendix A: Interface Architecture" on page 57.

For high-performance interfaces, use all DQ groups on one top/bottom side of the device. When all DQ groups on that side are used up, use the DQ groups on the opposite side. Note that the DLL signals may not be shared between the top and bottom. Therefore, each side should have its own DLL instantiation.

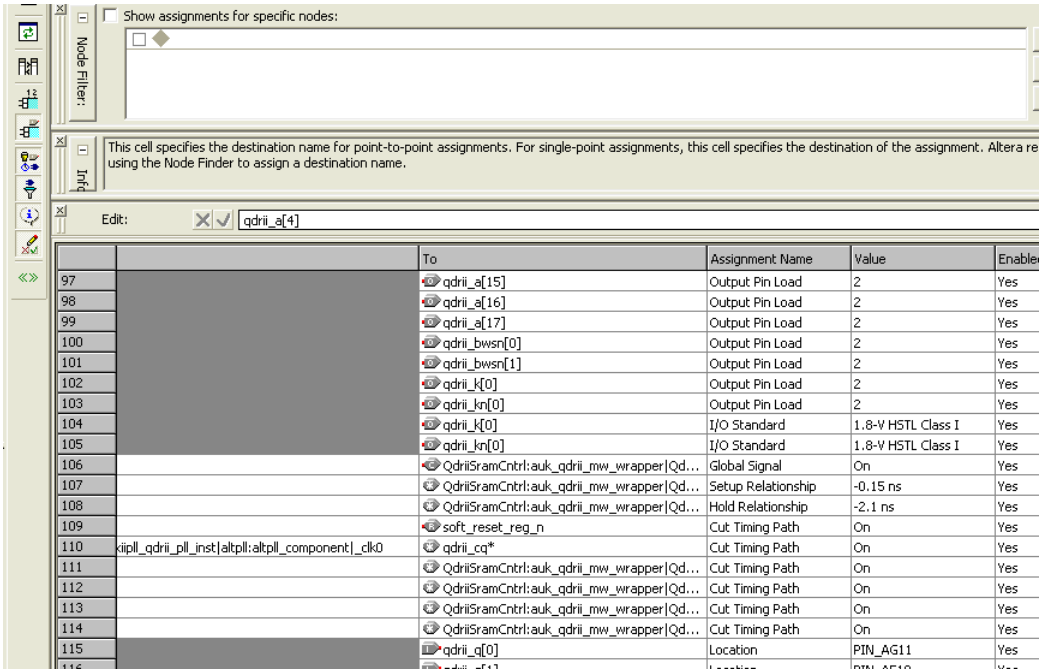
The QDRII Memory Controller always generates one address bus per memory chip. Loading on the address bus could be high if the address buses are shared. This results in timing violations. You have the option to use one address bus per chip. It is up your discretion whether to use them or not. If you choose not to use the additional address bus, you can leave the second bus unconnected, and the Quartus II software optimizes it away. You could instead remove the second bus from the clear text path.

For more information, refer to the *QDRII SRAM Controller MegaCore Function User Guide*.

Adjust Constraints

On the **Assignments** menu, click **Assignment Editor** (or click the **Assignment Editor** icon). The **Assignment Editor** tab appears (Figure 35). On this tab, you can edit the data that appears in assignment editor if you want to modify the constraints and repeat Analysis & Synthesis before compilation.

Figure 35. Assignment Editor



Compile Design and Report Timing

Now that the constraints are added and adjusted, the design is ready for compilation.

1. On the **Processing** menu, click **Start Compilation**. Alternately, on the menu bar, click the **Start Compilation** button ( ).

   Compilation performs Analysis and Synthesis, Fitter, Assembler, and Timing Analysis all in one step.

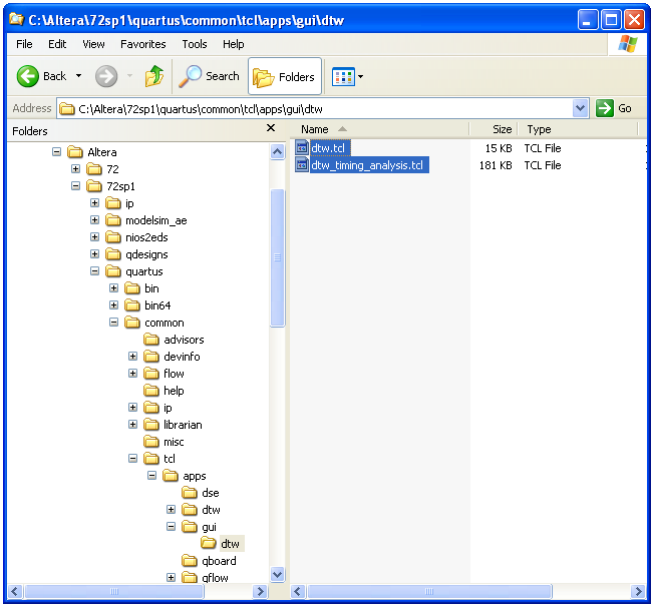
2. When the compilation is complete, the **Compilation Was Successful** window appears.

Run DTW

You should run **dtw_timing_analysis.tcl** to review the memory interface timing numbers. To do this, follow these steps:

1. From the Quartus II directory in your local directory, navigate to the **dtw** folder (Figure 36).

Figure 36. dtw Folder

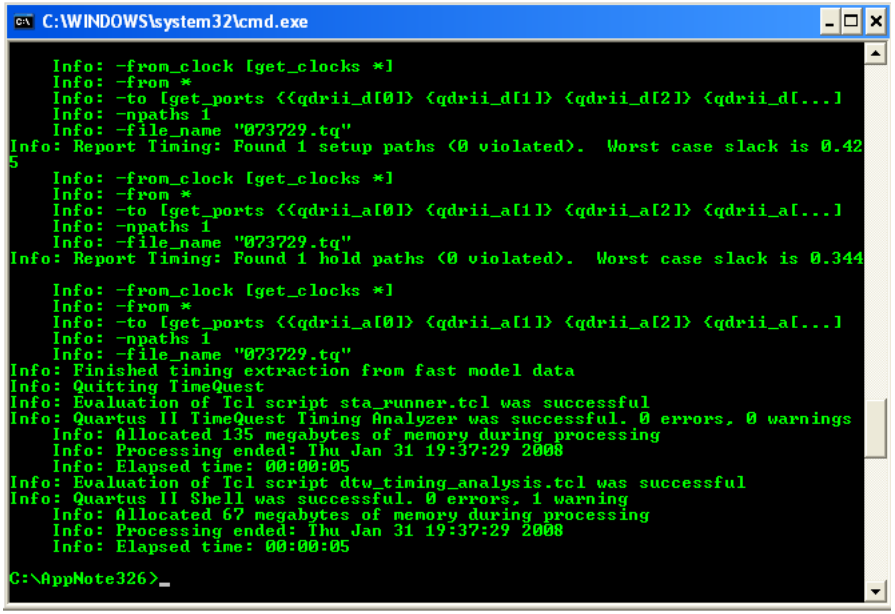


2. Copy this folder into your project directory.
3. Open a command window:
   a. In your PC, click **Start**.
   b. Click **Run**.
   c. Type: `cmd` ↵
4. Type the following command at the prompt:

   `quartus_sh -t dtw_timing_analysis.tcl -dwz_file \`
   `<variation name>.dwz` ↵

   The text shown in the **cmd** window shown in Figure 37 appears.

Figure 37. Running DTW Timing Analysis Script

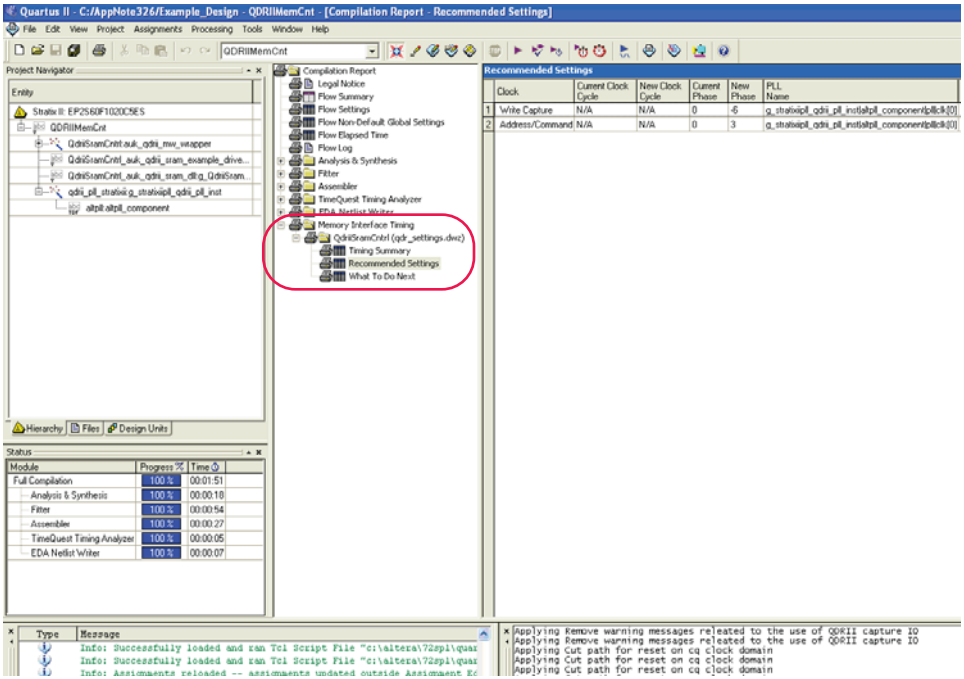


5. When the **dtw_timing_analysis** Tcl script is complete, go back to the Quartus II software, close the summary report, then reopen it. Memory interface timing is now added to the reports.

6. Click the memory interface timing as shown in Figure 38 and review the data. If your design's timing requirements are still not met, click **What To Do Next**.

For more details about DTW, refer to the *DDR Timing Wizard User Guide*.

Figure 38. Memory Interface Timing Report



7. In the **Compilation Report**, click the **TimeQuest Timing Analyzer** report.

If your design does not meet the timing requirements, the timing violations are indicated by red text. The most common timing violations are setup, hold, removal, and recovery violations. To investigate and correct these violations, you must identify and fix them, then re-compile the design (refer to "Report Timing Violations and Adjusting Constraints").

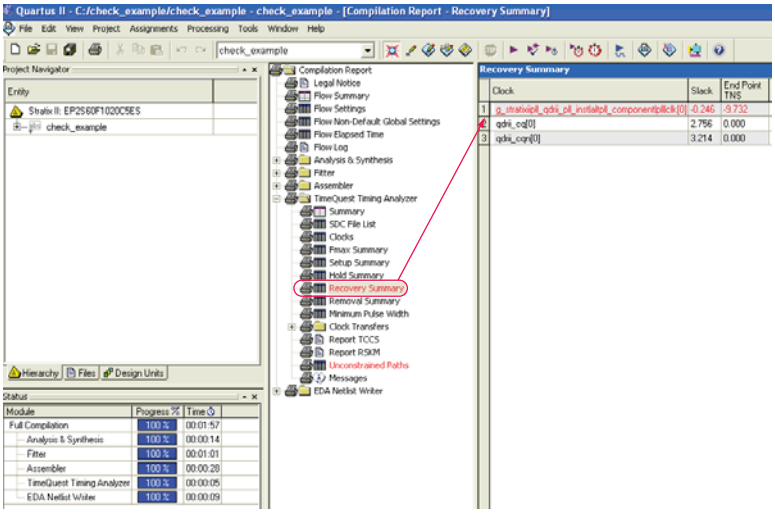
You might need to repeat these steps several times to fix all of the timing violations.

Report Timing Violations and Adjusting Constraints

In this example, there are two timing violations—Recovery and Unconstrained Paths.

1. Double-click the specific violation. The details of the violation appear to the right of the Compilation Report (Figure 39).

Figure 39. TimeQuest Timing Analyzer Report

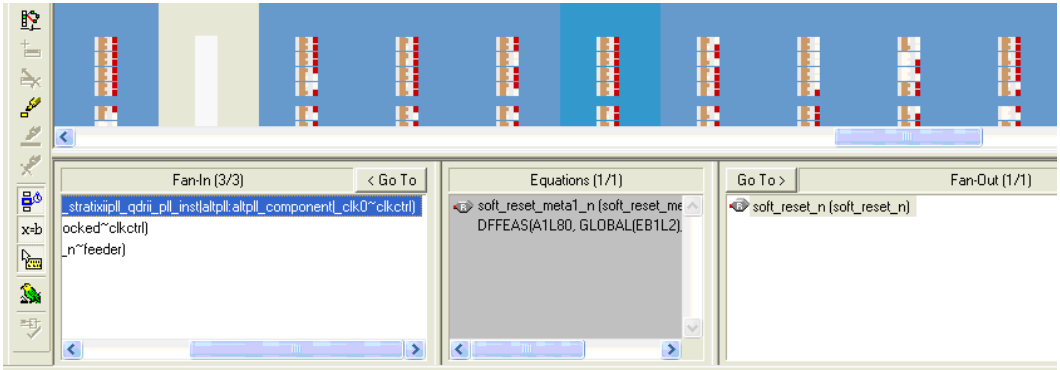


2. Click **Recovery Summary**. In the summary on the right, it indicates that the signal affected is `pllclk[0]`. To debug the violation you must track down and follow the affected signal (Figure 39).

3. Right-click the signal and select **Locate**.

4. Click **Locate in Chip Planner**. The Chip Planner shows all the related signals, as shown in Figure 40.

Figure 40. Recovery Summary Timing Violation



5. To investigate the fan-in and fan-out signals further, right-click the signal and locate it in the **Assignment Editor**.

   In this example, the signal `soft_global_reset`, which is the reset signal coming into the chip, is a global signal. To correct this violation, add the following line to the QSF file:

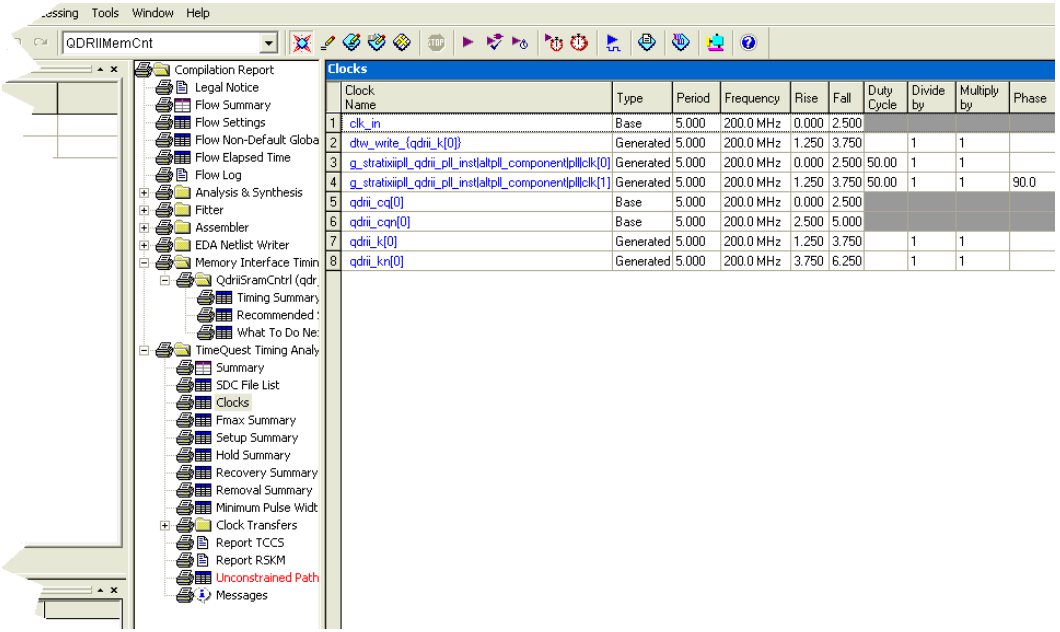
   ```
   set_instance_assignment -name GLOBAL_SIGNAL OFF \
   -to soft_reset_n
   ```

6. Compile the project again.

You can click on each summary to review the timing numbers for clocks, setup, hold, and so forth.

In this example, the recovery timing violation resolved and the design was compiled. The TimeQuest Timing Analyzer report now shows **Unconstrained Paths** as the only timing requirement violation, as shown in Figure 41. This is addressed on page 53.

Figure 41. Unconstrained Paths Violation



TimeQuest Timing Analysis

You can use the TimeQuest Timing Analyzer to review the timing data in more detail.

1. To run the TimeQuest Timing Analyzer, on the menu bar, click the clock icon ( ). The **Quartus II TimeQuest Timing Analyzer** window appears.

2. In the **Tasks** window on the left are three tasks:

   - **Create Timing Netlist**—This creates the slow model timing netlist; to run the fast model (if you have already run the default netlist), type the following in the **Tcl** window:

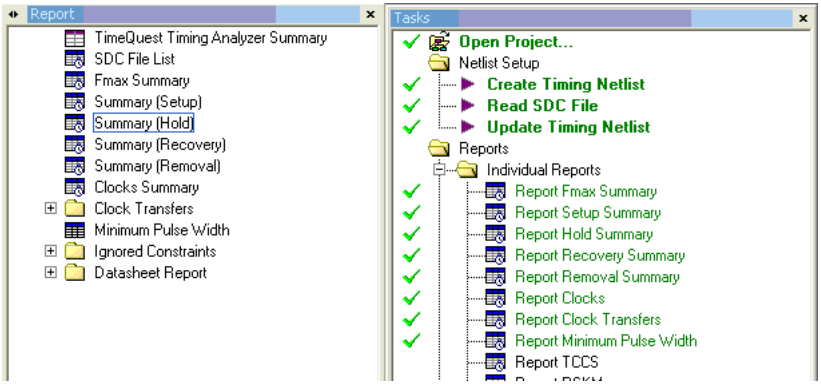
     ```
     delete_timing_netlist
     create_timing_netlist -model fast
     ```

   - **Read SDC File**
   - **Update Timing Netlist**

   Double-click each of these tasks in the order shown.

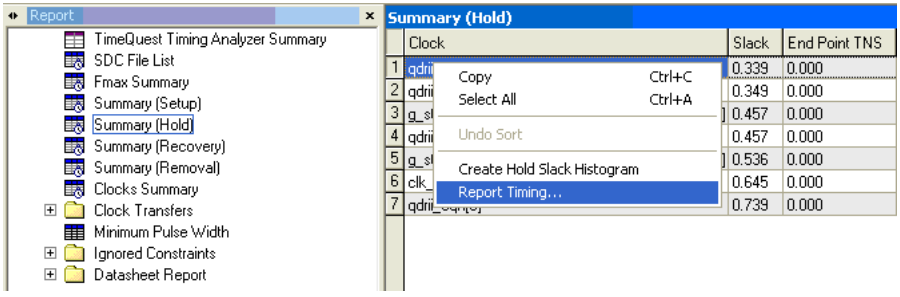
3. When these tasks are complete, you can open each individual report by double-clicking each one. Each report that is opened in the **Tasks** window appears in the **Report** window (Figure 42).

Figure 42. The TimeQuest Timing Analyzer Report and Tasks Windows



4. Left-click each item and find the timing report specific to that signal, as shown in Figure 43.

Figure 43. Specific Timing Report



5. Click **Report Timing**. The **Report Timing** dialog box appears (Figure 44). In this dialog box, you can specify a combination of **from** and **to** clocks and the number of reported paths in the window as shown in Figure 44.

6. When you have indicated the parameters you require, click the **Report Timing** button.

Figure 44. Report Timing Dialog Box

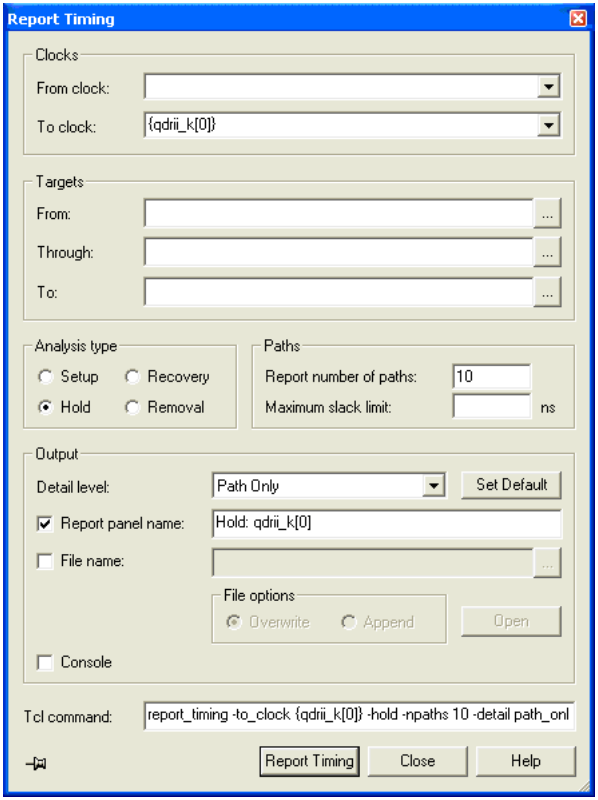
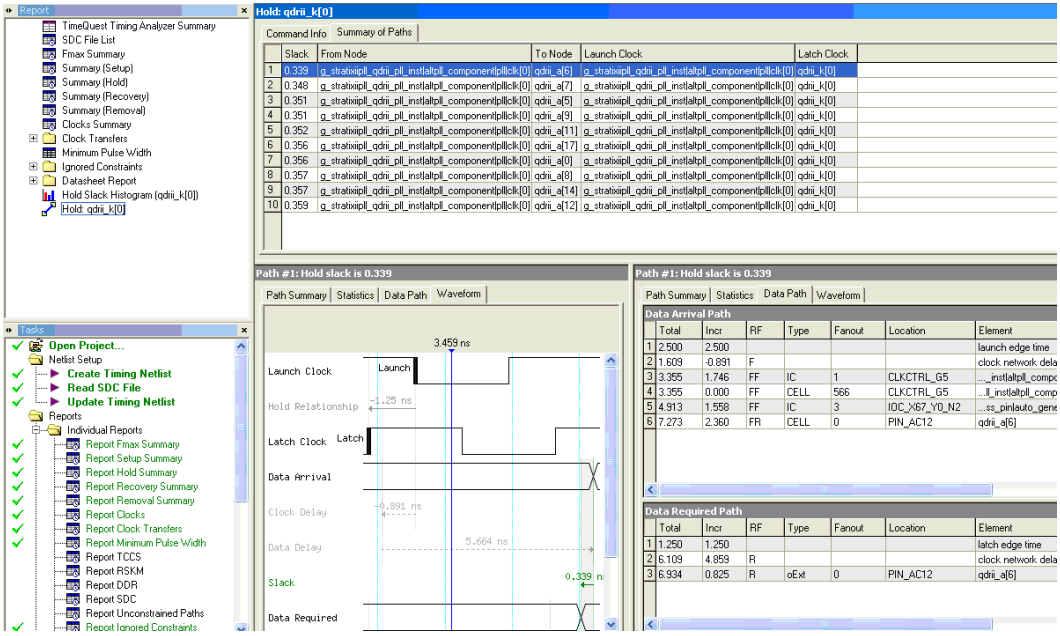


Figure 45 shows a detailed report of ten paths in the design in which the K clock is the latch clock. Note that the software has the ability to show the data in different GUI formats.

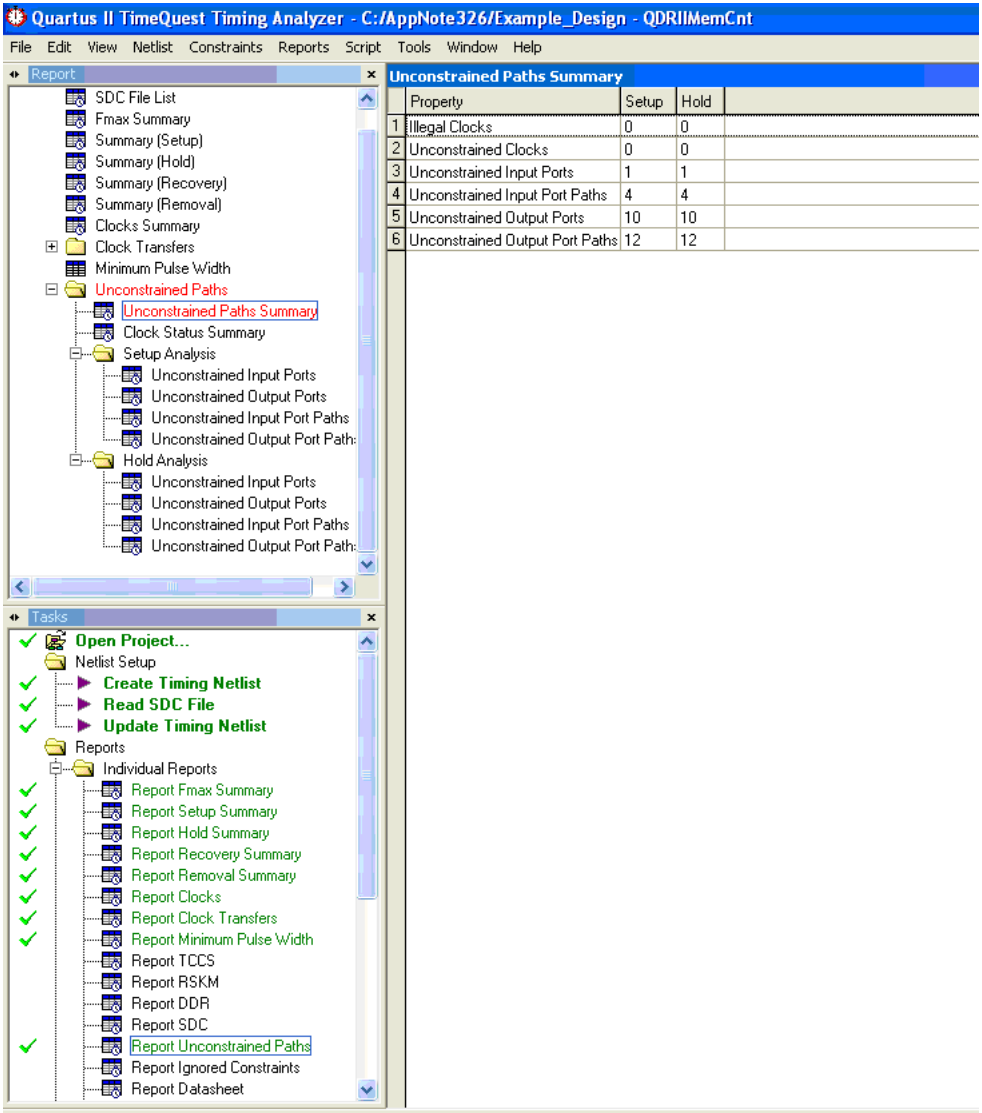
Figure 45. Detailed Report



Remember that the TimeQuest Timing Analyzer report shows **Unconstrained Paths** (in red in Figure 46) as the only timing requirement violation.

The **Unconstrained Paths Summary** shows that the paths that are not constrained are the outputs going into LEDs on the board to flag the test results and the reset input. Because constraining these pins is not crucial to the functionality and performance of this design, in this example this violation is ignored.

Figure 46. Unconstrained Paths Summary



Hardware Verification

This design was programmed and verified on a device mounted on an internal board used for internal purposes—a Stratix II memory board II. The Stratix II memory board II features an EP2S60F1020C5ES device and a QDRII SRAM memory device.

Figure 47 shows the Stratix II memory board II block diagram.

Figure 47. Stratix II Device Memory Board

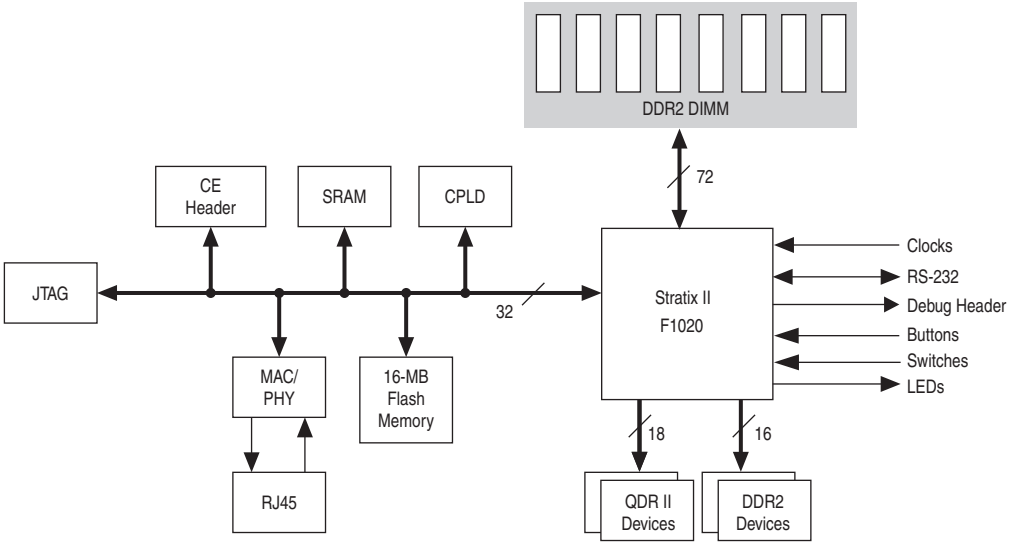
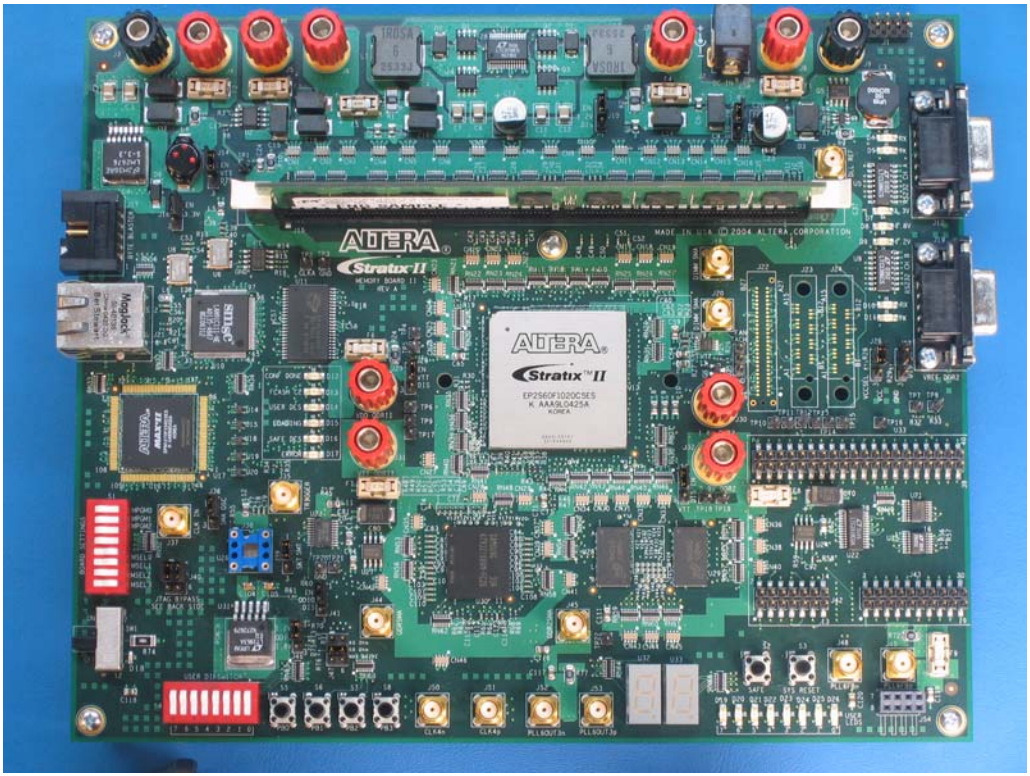


Figure 48 shows the Stratix II memory board II.

Figure 48. Stratix II Memory Board II



Conclusion

QDRII+ and QDRII SRAM devices offer enhanced timing margins and flexibility over QDR SRAM devices. Designed for high-bandwidth communications, networking, and DSP applications, QDRII+ and QDRII SRAM devices and Altera's Stratix II, Stratix II GX, Stratix, and Stratix GX devices help you take advantage of QDRII+ and QDRII SRAM technology and achieve high-memory bandwidth through a simple, proven interface.

Appendix A: Interface Architecture

Altera provides two different methods for implementing the read-side interface to QDRII+ and QDRII SRAM devices. Stratix and Stratix GX devices use a read-side PLL to center-align the CQ echo clock with the read data (Q). Stratix II and Stratix II GX devices also offer PLL-based implementation, similar to Stratix and Stratix GX devices. Stratix II and Stratix II GX devices can also use an enhanced DLL and delay-shift circuitry to center-align the echo clocks (CQ and CQ#) with read data (Q).

Write-side implementation is identical for both PLL- and DLL-based implementations. A write-side PLL outputs two clocks that generate the write data (D) and center-aligned system clocks (K and K#) using the dedicated double data rate input/output (DDIO) circuits. This implementation results in matched propagation delays for clock and data signals from the FPGA to the QDRII or QDRII+ SRAM device, minimizing skew.

For detailed information about the DDR I/O and DLL circuits, refer to the *External Memory Interfaces in Stratix II and Stratix II GX Devices* chapter of the *Stratix II Device Handbook*.

DLL-Based Data-Path Architecture

The QDRII+ and QDRII SRAM interface implementations in Stratix II and Stratix II GX devices use the following:

- A write-side PLL to generate K and K# system clocks and clock out address, command, and data signals
- A read-side DLL-based phase-shift circuitry to register read data from the memory using echo clocks CQ and CQ#

Altera recommends the following read capture implementation for data captures from QDRII+ and QDRII SRAM devices when using complementary echo clocks (CQ and CQ# signals).

The Stratix II and Stratix II GX IOE contains two input registers and a latch. The CQ and CQ# echo clock signals clock the positive and negative half-cycle registers during reads. The latch holds the negative half-cycle data until the next rising edge on CQ. However, additional IP is required when the complementary clocks have non-50% duty cycle or skew. This is because the latch, controlled by the CQ clock, is transparent until just after the register clocked on the CQ# signal captures the data.

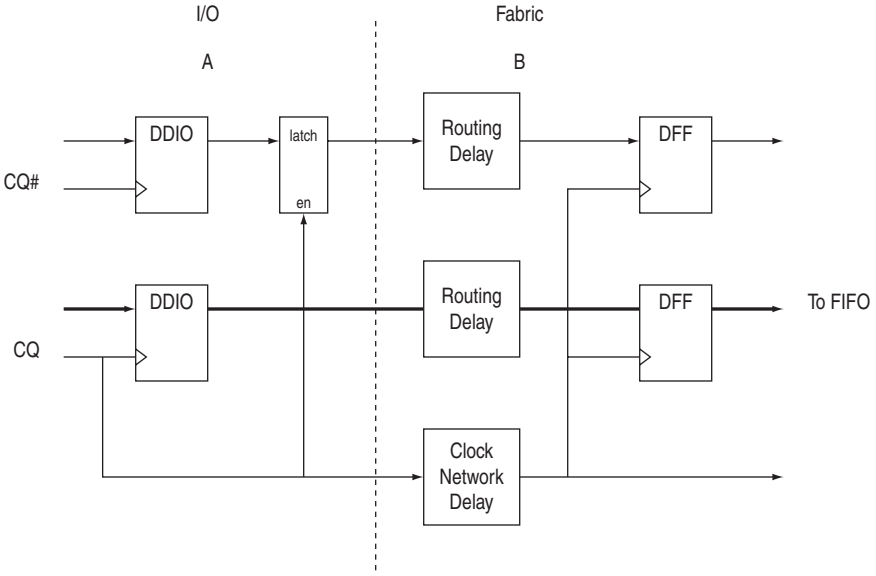
Instead of just capturing the data in the latch, the captured read data is recaptured using the CQ echo clock in the FPGA fabric using a zero-cycle path. The CQ echo clock is routed into the FPGA fabric using dedicated clock routing (Altera recommends global routing) to provide minimum clock skew across all recapture registers. If you do not have enough

global clock network resources, you have the option of using the regional clock network, though routing the CQ over a clock network adds delay. When **Optimize Hold Timing** is set to all paths, the Quartus II software fitter places and routes the recapture registers so that the data delay is sufficient to meet the setup and hold requirements at the logic array registers.

You should only use regional routing if you run out of global clock networks.

Figure 49 shows a block diagram of the new read capture implementation.

Figure 49. Block Diagram of the New Read Capture Implementation

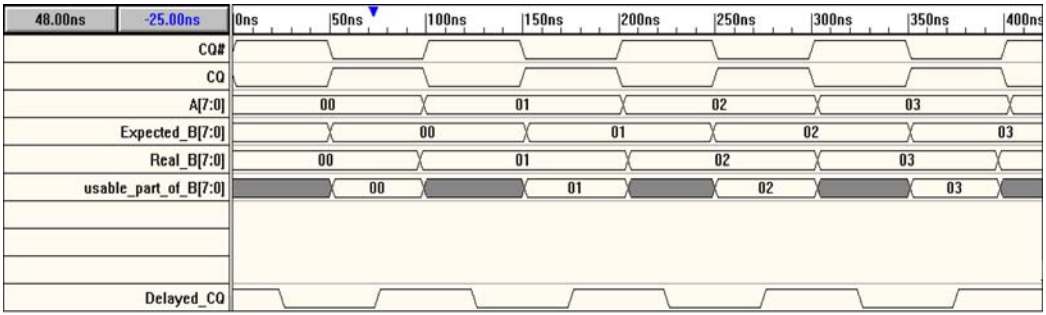


The data from the latch becomes valid following the rising edge of the CQ signal (when the latch becomes transparent) and, in a worst-case condition, becomes invalid following the rising edge of the CQ# signal (when roughly half a cycle equals $t_{KHK\#H}$). This is done by creating a zero-cycle path between the latch and a logic array register. The data is recaptured in the logic array using the same edge of the CQ signal that makes the latch transparent. Both the CQ signal and the data cross the IOE-to-logic array boundary where they are delayed. The CQ signal is delayed by slightly more than the data needed to meet the setup time for this register. However, the delay is not enough to violate the register's

hold time, which is related to the rising edge of the CQ# signal. The data is recaptured in the FPGA logic array while the latch is valid, so the IOE capture register timing margins are not impacted.

Figure 50 shows an IOE timing diagram that assumes the latch is still transparent on the rising edge of CQ#. The `Real_B`, `Expected_B`, and `Delayed_CQ` signals represent the data and clock to the recapture registers. The output of latch B is either `Real_B` or `Expected_B`, depending on the relationship between CQ and CQ#. To cover both cases, the `usable_part_of_B` signal should be captured before going to the resynchronization FIFOs. Routing delay aligns the data with the clock.

Figure 50. OE Timing Diagram



To implement this new capture data path, complete the following in the Quartus II Assignment Editor:

- Add a set of registers following the IOE capture registers/latch
- Constrain the delayed CQ to go on a global or regional clock network
- Enable the **Optimize Hold Timing** feature of the Quartus II fitter
- Add setup and hold timing constraints on the new recapture registers

The Tcl commands for the assignments above are as follows:

Example 2.

```
set_global_assignment -name OPTIMIZE_HOLD_TIMING "ALL PATHS"
set_instance_assignment -name GLOBAL_SIGNAL "ON" -to "dqsbusout"
set_instance_assignment -name SETUP_RELATIONSHIP "-0.2 ns" -from * to
resync*
set_instance_assignment -name HOLD_RELATIONSHIP "-1.6 ns" -from * to
resync*
```

If you want to use a regional clock network, replace the second assignment with the following command:

```
set_instance_assignment -name GLOBAL_SIGNAL
"regional clock" -to "dqsbusout".
```

Usage details and guidelines for the above assignments are as follows:

- The CQ recapture clock must be the DQS-delayed output fed through the global clock network to slow it down enough to meet the setup requirement (this has the added benefit of reducing the clock skew).
- The **OPTIMIZE_HOLD_TIMING** option forces the fitter to use about 2.5 to 3.0 ns to route data to the recapture register. This time is relatively matched to the routing delay of the recapture clock path (~4 ns), allowing positive timing margins at both the fast and slow timing models.
- `dqsbusout` is the DLL-delayed CQ clock output to the FPGA fabric when using the WYSIWYG interface. When using the ALTDQS megafunction, the CQ clock output to the logic array is called `dqinclk`.
- `resync[]` are the new logic array recapture registers that are added.
- The setup and hold relationship assignments are negative because the logic array recapture occurs using the "same" CQ edge that signifies the opening of the data valid window that is available in the IOE. (The negative half-cycle data latch is made transparent by this CQ edge; the positive half-cycle data is captured in the IOE input register by this CQ edge.)
- The minimum data valid window for this zero-cycle recapture path is equal to $t_{KHK\#H}$. The window opens with a rising edge on the CQ signal and closes with a rising edge on the CQ# signal.
- The **SETUP_RELATIONSHIP** parameter is ideally 0 ns, but Altera recommends a 200-ps adder as a conservative guard against possible on-chip variations.
- The **HOLD_RELATIONSHIP** parameter is defined as:
  $-[1 \times \text{data valid time}] = -[t_{KHK\#H}]$
  Again, add 200 ps as a guard against any on-chip variations, making the parameter 200 ps – $[t_{KHK\#H}]$.

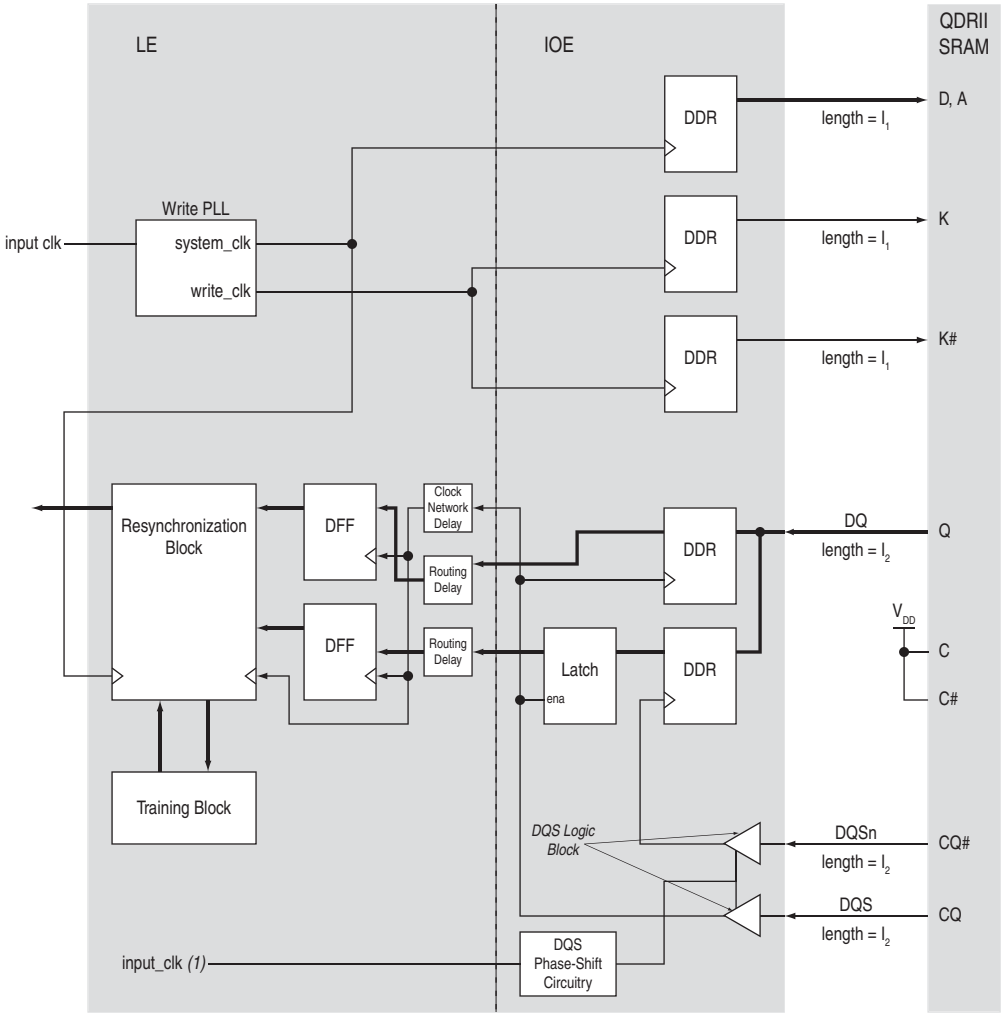
- The setup and hold relationship requirements apply to all synchronous inputs to the recapture registers, including synchronous clears and enables.

Figure 51 shows the Stratix II memory interface data-path architecture. Specifically, it shows how to connect the clocks, data, address, and control pins in Stratix II devices when interfacing with QDRII+ and QDRII SRAM devices.

The write PLL generates two clock outputs, `system_clk` and `write_clk`, that have a 90° phase offset. The `system_clk` output clocks out the address, command, and data signals to the QDRII+ and QDRII SRAM; the `write_clk` output generates the `K` and `K#` memory input clocks. This architecture center aligns the `K` and `K#` write clock edges to the output data (`D`) and address (`A`) signals. All outputs to the memory (including the clock) use the DDIO circuitry in the I/O cell, significantly minimizing the skew between clock and data channels.

The read-side DQS phase-shift circuitry generates a center-aligned version of CQ and CQ# echo clocks for read data capture. The captured data can then be resynchronized to the system clock.

Figure 51. QDRII+ and QDRII SRAM Interface Data Path Using DLL-Based Dedicated DQS Circuitry



Note to Figure 51:
(1) The input reference clock can be from the `input_clk`, another pin, or a PLL output.

PLL-Based Data-Path Architecture

The QDRII+ and QDRII SRAM implementation without using the DQS circuitry uses two PLLs in Stratix II, Stratix II GX, Stratix, and Stratix GX devices:

- A write-side PLL to generate K and K# system clocks and clock out address, command, and data signals.
- A read-side PLL to register data from memory using phase-shifted versions of echo clock CQ.

In source synchronous mode, enhanced PLLs compensate for clock delay to top and bottom I/O registers; Fast PLLs compensate for clock delay to side I/O registers. While implementing source synchronous receivers in these I/O banks, use the corresponding PLL type for best matching between clock and data delays. In the Quartus II software, set the input pin to register delay chain within the IOE to zero for all data pins.

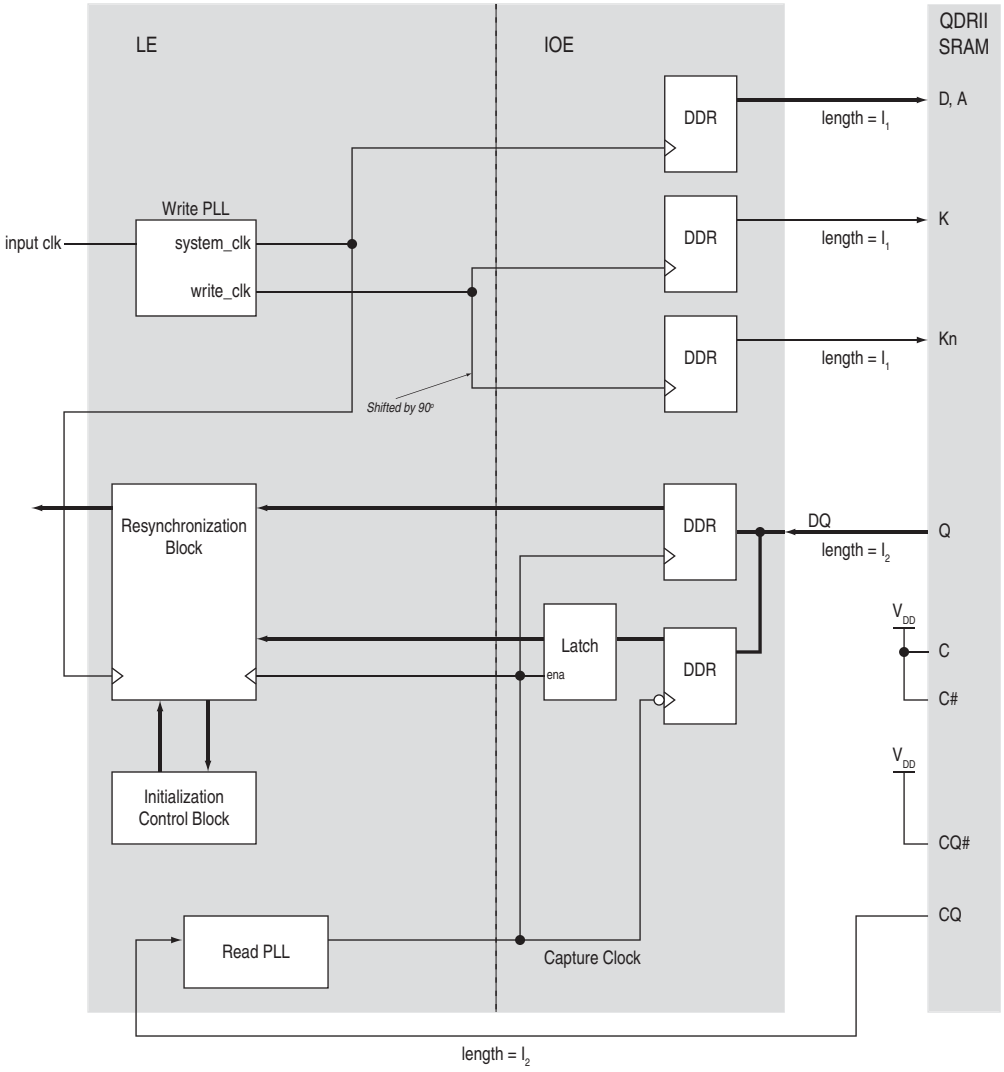
Figure 52 on page 65 shows the data-path architecture using non-DQS circuitry in Stratix II, Stratix II GX, Stratix, and Stratix GX devices. Specifically, it shows how to connect the clock, data, address, and control pins in Stratix II, Stratix II GX, Stratix, and Stratix GX devices when interfacing with QDRII+ and QDRII SRAM devices. The write-side PLL generates two clock outputs, system_clk and write_clk, that have a 90° phase offset. The system_clk output clocks out the address, command, and data signals to the QDRII+ and QDRII SRAM; the write_clk output generates the K and K# memory input clocks. This architecture center aligns the K and K# write-clock edges to the output data (D) and address (A) signals. All outputs to the memory (including the clock) use the DDIO circuitry in the I/O cell, significantly minimizing the skew between clock and data channels.

The read-side PLL generates a phase-shifted version of the CQ echo clock to capture read data (Q) signals. The captured data can then be resynchronized to the system clock. The read-side PLL is configured in the source-synchronous mode to ensure the CQ echo clock; the read Data (Q) maintain the same phase relationship at the clock and data parts of the capture registers.

In source-synchronous mode, enhanced PLLs compensate for clock delay to the top and bottom I/O registers; fast PLLs compensate for clock delay to the side I/O registers. While implementing source-synchronous receivers in these I/O banks, use the corresponding PLL type for best matching between clock and data delays (from input pins to register ports).

The Quartus II software associates a particular PLL with a particular I/O bank for source-synchronous operation, so you must ensure that the read PLL is on the same side of the device as the data pins. The clock delay to the worst-case I/O registers in this I/O bank are fully compensated and result in closely matched data delays and clock delays from the pin to the I/O registers across process, voltage, and temperature (PVT). When using I/O registers in the non-compensated I/O banks, clock delays and data delays are less closely matched. Use a fast PLL for implementing the interface on side I/O banks; use an enhanced PLL for implementing the interface on the top or bottom I/O banks for best clock and data delay matching. You must also set the input pin delay-to-register option to 0 in the Quartus II software.

Figure 52. QDRII+ and QDRII Data Path Architecture without DQS Circuitry Note (1)



Note to Figure 52:

(1) CQ# of the QDRII+ and QDRII SRAM is not used in this configuration and is left unconnected.

Appendix B: Manual Interface Timing Analysis

When designing an external memory interface for your FPGA, you must analyze timing margins for several paths. All memory interfaces require analysis of the read and write capture timing paths.

This section of the application note describes Altera's recommended timing methodology using write and read capture timing paths as examples. Altera recommends using DTW for timing analysis as shown in the example design, but if you are not using DTW, you should use this methodology for analyzing timing for all applicable timing paths (including address/command, resynchronization, and so on). While these analyses account for all FPGA-related timing effects, you should design in adequate margin to account for board-level effects. To start, you can analyze the read and write capture timing margins for a QDRII+ or QDRII SRAM interface with a Stratix II device. The timing analysis methodology example in this section uses a EP2S60F1020C5ES FPGA interfaced with a Cypress CY7C1313AV18 QDRII memory device for DLL- based read and timing margins calculation. For the rest of the calculations in this section, a EP2S90F1020C3 FPGA interfaced with a Cypress CY7C1315BV18-300 QDRII memory device is used.

However, you can use the same methodology with your preferred FPGA and memory device. You can analyze the write and read capture timing margins for a QDRII+ SRAM interface with a Stratix II device by substituting the QDRII memory specifications used in the timing analysis methodology example with the QDRII+ memory specifications.

Methodology Overview

To analyze any timing path you must consider the data and clock arrival times at the destination register. Figure 53 illustrates a simplified block diagram to analyze timing at any register. The setup time margin is defined as the time between "earliest clock arrival time" and "latest valid data arrival time" at the register ports. Similarly, the hold-time margin is defined as the time between the "earliest invalid data arrival time" and the "latest clock arrival time" at the register ports. These arrival times are calculated based on propagation delay information with respect to a common reference point (such as a CQ or CQ# edge or system clock edge).

Figure 53. Simplified Block Diagram for Timing Analysis

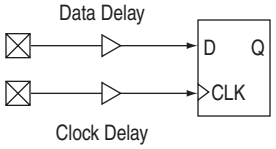
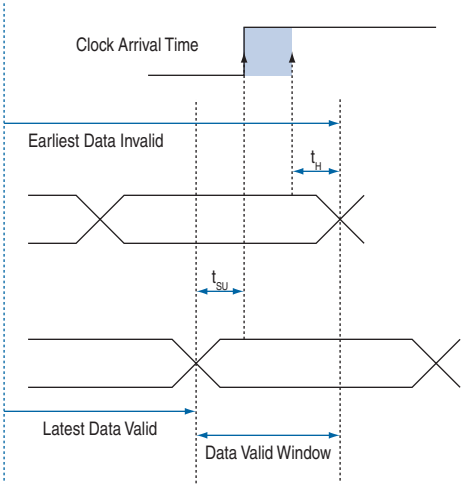


Figure 54 shows how a data valid window is derived based on arrival times.

Figure 54. Data Valid Window Timing Waveform



FPGA Timing Information

To ensure that your design is robust, evaluate the timing margins at all process, voltage, and temperature (PVT) conditions. To facilitate this, Altera provides two device timing models (slow corner model and fast corner model) in the Quartus II software:

- The slow corner model provides timing delays between two nodes within the FPGA with slow silicon, high temperature, and low voltage. Therefore, the model provides the slowest possible delay for that timing path on any device for that particular speed grade.
- The fast corner model provides timing delays between two nodes within the FPGA with fast silicon, low temperature, and high voltage. Therefore, the model provides the fastest possible delay for that timing path on any device.

While almost all FPGA timing delays and uncertainties are modeled in the Quartus II software, a handful of factors are not modeled and you should account for them during margin analysis. For example, clock jitter on PLL and DLL outputs at a given PVT operating condition is not modeled. Such unmodeled uncertainties are specified in the Stratix II device data sheet. These timing uncertainties or adder terms, when used in conjunction with the Quartus II reported timing data, provide the most

accurate device timing information. The following analysis details the use of these timing adder terms. Simultaneous switching noise (SSN) and other signal integrity effects on timing are not included in this analysis.

Read Timing Margins for DLL-Based Implementation

During read operations, the QDRII+ or QDRII memory device provides a pair of complementary echo clocks (CQ and CQ#) that are edge-aligned with the data bus (Q). The memory controller (FPGA) shifts the clock edge to the center of the data valid window and captures the Q input data. Figure 55 illustrates the timing relationship between the echo clocks and Q outputs from the memory during a read operation.

Figure 55. 10 QDRII+ and QDRII SRAM Read-Cycle Timing Waveform

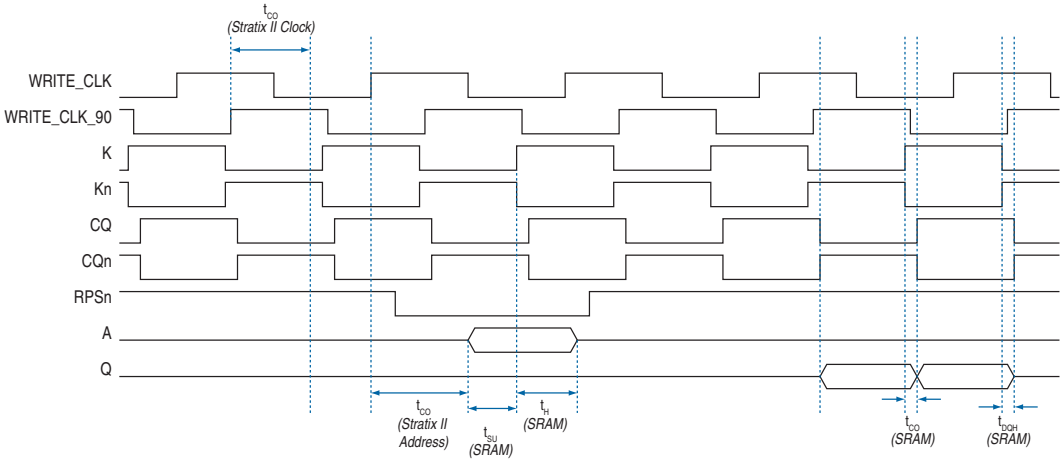
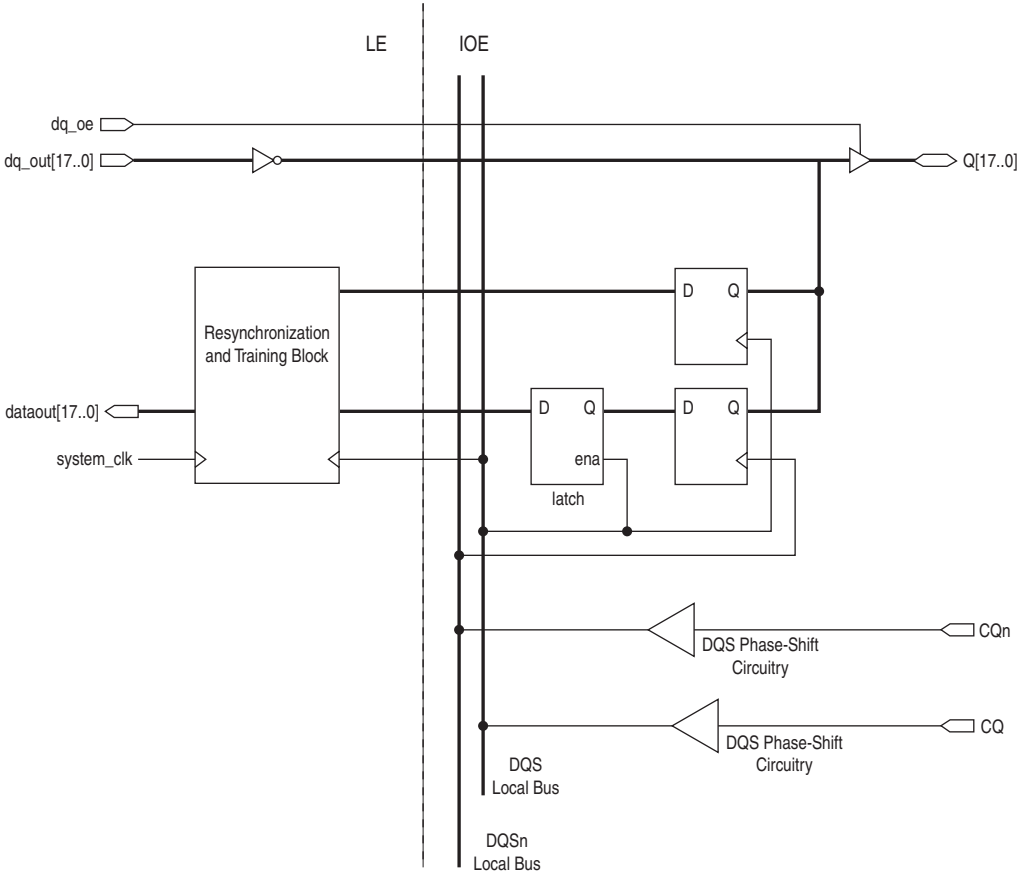


Figure 56 shows the read data path from the Stratix II device. The CQ and CQ# signals go to the DQS phase-shift circuitry and are shifted. The shifted CQ and CQ# signals go to the DQS and DQSn bus to clock the DQ input registers.

Figure 56. 11 QDRII+ and QDRII SRAM Read Data Path in Stratix II Devices



During the system initialization, the Altera QDRII SRAM Controller MegaCore function performs a dummy write followed by a read to calculate the latency required by the resynchronization register. The data capture, training block, and resynchronization logic is available from the Altera QDRII SRAM Controller MegaCore function. To use your own controller, create the necessary logic to connect this logic to the read and write control logic

Memory Timing Parameters

Start the read timing analysis by obtaining the timing relationship between the data (Q) and echo clock (CQ, CQ#) outputs from the QDRII SRAM memory device. You are analyzing timing for 200 MHz clock speeds or 400 Mbps data rates, so the half clock period is 2200 ps. This is specified as $t_{KHK\#H}$ in the memory data sheet. In addition to $t_{KHK\#H}$, the memory also specifies t_{CQHQV} and t_{CQHQX}. The former specifies the maximum time from an echo clock (CQ or CQ#) edge to the last valid data (Q). The latter specifies the earliest time at which the data can become invalid.

Using these memory timing parameters, you can calculate the data valid window at the memory to be $t_{KHK\#H} - t_{CQHQV} + t_{CQHQX} = 1500$ ps. Assuming the board trace length variations among all CQ, CQ#, and Q traces are not more than ±20 ps, the data valid window present at the FPGA input pins is 1460 ps.

FPGA Timing Parameters

FPGA timing parameters are obtained from two sources: Quartus II timing analyzer and the Stratix II data sheet. While the former provides all clock/data propagation delays, the data sheet specifies all clock uncertainties and skew adder terms.

The timing analysis methodology outlined earlier suggests you calculate the earliest and latest arrival times for clock and data. Begin with the clock (CQ and CQ#).

The Stratix II features dedicated DQS phase-shift circuitry in the top/bottom IO banks of the device, which center-aligns the echo clocks edge with respect to the Q input signals. This phase-shift circuitry has a coarse and fine delay resolution. The coarse delay feature is self-compensating over PVT and has a resolution of 22.5° to 36° of the reference clock frequency (based on the DLL mode of operation). The target memory speed is 200 MHz, so you can select between DLL modes 1 and 2. DLL mode 2 gives you a coarse phase resolution of 30°, while mode 1 gives you 22.5° resolution. You can further fine tune this phase shift with a DLL offset implement using uncompensated delay chains or the DLL's phase-offset feature.

You can analyze timing with a 90° phase shift on the DQS strobe (DLL mode 2), knowing that the phase shift (and DLL mode) can always be adjusted at the end of this timing analysis for balanced setup and hold margins on the read capture register.

The DQS phase-shift circuitry uses a DLL to provide the self-compensating coarse delay shift. This means you have to account for any jitter and phase-shift error on the DQS signal. From the *DC & Switching Characteristics* chapter of the *Stratix II Handbook*, you obtain $t_{DQS\_PHASE\_JITTER}$ (± 45 ps) and t_{DQS_PSERR} (± 52.5 ps) timing parameters for DLL mode 2 with three stages of delay chains.

After encountering the phase-shift circuitry, the DQS travels on a dedicated local clock bus to the DQ capture registers. The fanout of this local clock bus could range from ×4 to ×36. While the Quartus II software provides clock propagation delays to each of these DQ register clock ports, you account for additional uncertainties with the $t_{DQS\_SKEW\_ADDER}$ term listed in the data sheet. For the ×18 mode used by this Cypress QDRII SRAM device, the skew adder is ± 38 ps.

To obtain the Quartus II software timing data for the target device, you should instantiate and compile the QDRII+ or QDRII SRAM Controller MegaCore function. If you are using your own controller logic, you should instantiate the clear-text QDRII+ or QDRII data path instead to obtain timing delays. For the read interface, the Quartus II software reports individual setup and hold times for each Q pin.

If you are using TAN, selecting the List Paths option in the timing report provides data and clock propagation delays for that Q pin. Select the worst-case setup and hold Q registers and extract the minimum and maximum propagation delays.

If you are using the TimeQuest Timing Analyzer, you must extract the delays. To extract minimum and maximum propagation delays, refer to "TimeQuest Timing Analysis" on page 50. You must follow those steps to get the detailed timing report for `cq` and `cqn`, like the one shown in Figure 45 on page 53. When you generate these reports, find the minimum and maximum delays. By default, the TimeQuest Timing Analyzer generates data for a slow model. To extract propagation delays for a fast model, you must delete the existing timing netlist and generate a new one. To do so, type in the commands shown in Example 3.

Example 3.

```
delete_timing_netlist and hit enter
create_timing_netlist -model fast hit enter
read_sdc hit enter
update_timing_netlist hit enter
```

Now, you can generate the detailed reports for the fast model.

Minimum and maximum propagation delays on the clock and data path are extracted and present in Table 8. This timing extraction is done twice, once with each device model (fast corner and slow corner). The difference between minimum and maximum delays is very small because of the matched routing paths within the die and package.

Table 8. Minimum and Maximum Propagation Delays

FPGA TIming Delays	Fast Corner (ns)	Slow Corner (ns) (C5 Speed Grade)
Data delay (minimum)	1.162	2.345
Data delay (maximum)	1.192	2.372
Clock delay (minimum)	2.058	3.191
Clock delay (maximum)	2.089	3.253
Micro setup	0.068	0.163
Micro Hold	0.037	0.096

Setup & Hold Margins Calculations

After obtaining all relevant timing information from the memory, FPGA, and board you are ready to calculate the setup and hold margins at the DQ capture register during read operations.

For more information on the FPGA specifications used in these calculations, refer to the External Memory Interface Specifications listed in the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II Handbook*.

For more information about the QDRII SRAM specifications used in these calculations, refer to CY7C1313 data sheet, "18-Bit QDR(TM)-II SRAM 4-Word Burst Architecture".

(1) Earliest clock arrival time = Minimum clock delay within the FPGA – DQS uncertainties

$= (\text{Clock Delay (minimum)} - t_{DQS\_PHASE\_JITTER} - t_{DQS\_PSERR} - t_{DQS\_SKEW\_ADDER})$

$= 3.191 - 0.045 - 0.038 - 0.043$

$= 3.055\,\text{ns}$ (at slow corner)

(2) Latest data valid time = Memory DQS-to-DQ valid + maximum data delay in FPGA

$= t_{CQHQX} + \text{data delay (maximum)}$

$= 2.372 + 0.35$

$= 2.722\,\text{ns}$ (at slow corner)

(3) Setup margin = Earliest clock arrival – latest data valid – micro setup – board uncertainty

$= t_{EARLY\_CLOCK} + - t_{LATE\_DATA\_VALID} - \mu t_{SU} - t_{EXT}$

$= 3.055 - 2.722 - 0.163 - 0.02$

$= 0.150$ ns (at slow corner)

Repeating these calculations with the fast corner timing model, you derive 0.150 ns of setup margin.

(4) Latest clock arrival time = Maximum clock delay within FPGA + DQS Uncertanties

$=$ Clock Delay (maximum) $+ t_{DQS\_PHASE\_JITTER} + t_{DQS\_PSERR} + t_{DQS\_SKEW\_ADDER}$

$= 3.253+0.045+0.053+0.038$

$= 3.389$ ns (at slow corner)

(5) Earliest data invalid time = memory DQS-to-DQ invalid + minimum data delay in FPGA

$= (t_{KHK\#H} + t_{CQHQV})$ + data delay (minimum)

$= 2.2 + (-0.35) + 2.345$

$= 4.195$ ns (at slow corner)

(6) Hold margin = Earliest data invalid time – latest clock arrival time–micro hold – board uncertainty

$= t_{LATE\_CLOCK} - t_{EARLY\_DATA\_INVALID} - \mu hold - \mu t_{EXT}$

$= 4.195 - 3.389 - 0.096 - 0.02$

$= 0.690$ ns (at slow corner)

Repeating these calculations with the fast corner timing model, you derive 0.0.730 ns of hold margin.

Table 9 shows the calculation that was described in a spreadsheet format.

Table 9. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP2S60F102C5ES Using DQS Mode (Part 1 of 3)

	Parameter	Fast Corner Model	Slow Corner Model	Description
Memory Specifications *(1)*	$t_{KHK\#H}$	2.20	2.20	K clock rising edge to K# clock rising edge
	t_{CQHQV}	0.35	0.35	Clock-to-out time for read data with respect to echo clock
	t_{CQHQX}	–0.35	–0.35	Read data hold with respect to echo clock

Table 9. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP2S60F102C5ES Using DQS Mode (Part 2 of 3)

	Parameter	Fast Corner Model	Slow Corner Model	Description
FPGA Specifications *(2)*	$t_{DQS\_PHASE\_JITTER}$	0.045	0.045	Phase jitter on CQ/CQ# output delayed by DLL (three delay stages = ± 3 × 15)
	t_{DQS_PSERR}	0.053	0.053	Phase-shift error on CQ output delayed by DLL
	$t_{DQS\_SKEW\_ADDER}$	0.038	0.038	Clock delay skew adder for × 8
	Clock delay (minimum) *(3)*	2.058	3.191	Minimum CQ/CQ# pin to IOE register delay from Quartus II (with 90° DLL-based phase shift)
	Clock delay (maximum) *(3)*	2.089	3.253	Maximum CQ/CQ# pin to IOE register delay from Quartus II (with 90° DLL-based phase shift)
	Data Delay (minimum) *(3)*, *(4)*	1.162	2.345	Minimum Q pin to IOE register delay from Quartus II
	Data Delay (maximum) *(3)*, *(4)*	1.192	2.372	Maximum Q pin to IOE register delay from Quartus II
	μt_{SU}	0.068	0.163	Intrinsic setup time of the IOE register (rounded up).
	μt_{H}	0.037	0.096	Intrinsic hold time of the IOE register (rounded up).
Board Specifications	t_{EXT}	0.020	0.020	Board trace variations on the DQ and DQS lines

Table 9. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP2S60F102C5ES Using DQS Mode (Part 3 of 3)

	Parameter	Fast Corner Model	Slow Corner Model	Description
Timing Calculations	t_{EARLY_CLOCK}	1.922	3.055	Earliest possible clock edge after DQS phase-shift circuitry and uncertainties: minimum clock delay – $t_{DQS\_PHASE\_JITTER} - t_{DQS\_PSERR} - t_{DQS\_SKEW\_ADDER}$
	t_{LATE_CLOCK}	2.225	4.195	Latest possible clock edge after DQS phase-shift circuitry and uncertainties: maximum clock delay – $t_{DQS\_PHASE\_JITTER} + t_{DQS\_PSERR} + t_{DQS\_SKEW\_ADDER}$
	$t_{EARLY\_DATA\_INVALID}$	3.012	3.38	Time for earliest data to become invalid for sampling at FPGA flop: $t_{KHK\#H} + t_{DQS\_PSERR} + t_{CQHQV}$ + minimum data delay
	$t_{LATE\_DATA\_VALID}$	1.542	2.722	Time for latest data to become valid for sampling at FPGA flop: t_{CQHQX} + maximum data delay
Results	Read setup timing margin	0.292	0.150	$t_{EARLY\_DATA\_INVALID} - t_{LATE\_CLOCK} - \mu t_H - t_{EXT}$
	Read hold timing margin	0.730	0.690	$t_{EARLY\_DATA\_INVALID} - t_{LATE\_CLOCK} - \mu t_H - t_{EXT}$
	Total margin	1.022	0.840	Setup + hold-time margin

Notes for Table 9:

(1) The memory numbers used here for 200-MHz come from Cypress CY7C1313BV18 data sheet, Rev. C.

(2) This analysis is performed with FPGA timing parameters for an EP2S60F1020C5ES device. You should use this template to analyze timing for your preferred Stratix II density-package combination. For more information on FPGA specifications, refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II Device Handbook*.

(3) These numbers are from the Quartus II software, version 7.2 SP1 using the Altera IP MegaCore function v7.2 SP1 for a 18-bit QDRII SRAM interface.

(4) Package trace skews are modeled by the Quartus II software.

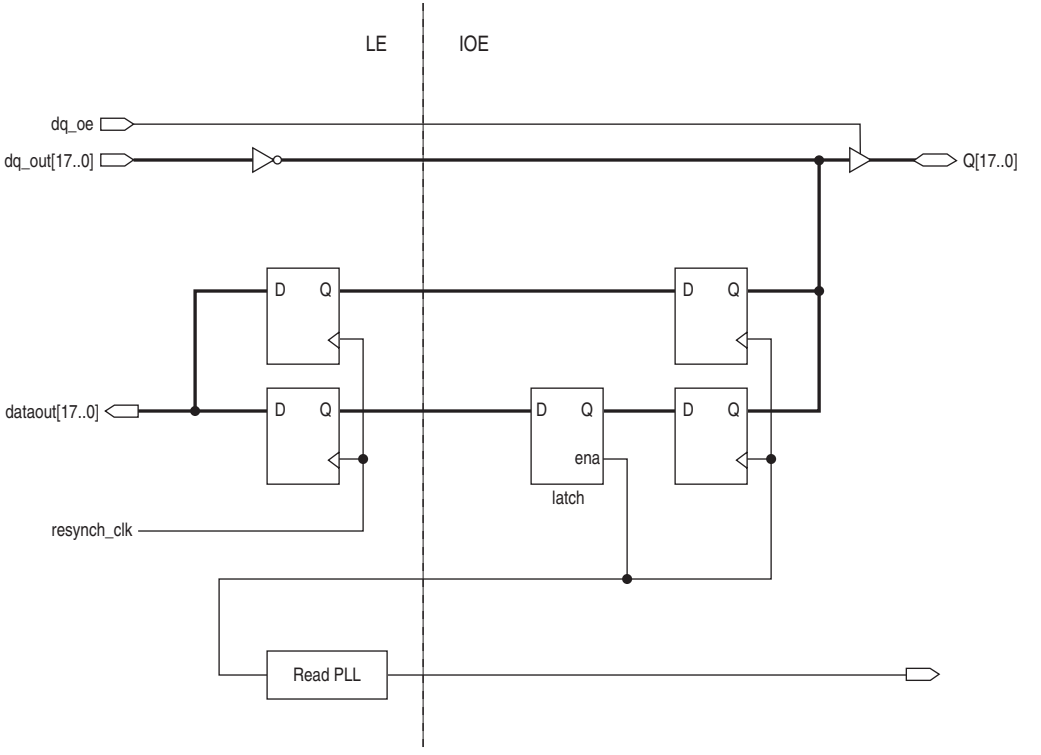
As you can see, the paper calculation correlates with the DTW results as shown in Figure 57. TimeQuest is automatically calculates the margins.

Read Timing Margins for PLL-Based Implementation

Timing margin analysis for a PLL-based implementation is very similar to the previously described DLL-based implementation. The only differences are the capture clock used and related clock uncertainties. In this mode, the echo clock, CQ signal is fed into a PLL inside the FPGA. The timing margin analysis presented here uses Stratix and Stratix GX devices.

Figure 57 shows the read data path from the Stratix and Stratix GX device. The CQ signal goes to the read PLL and is shifted. The shifted CQ signal goes to the DQS local bus and it clocks the DQ input registers. The register outputs then go to the resynchronization register in the logic array. The `resynch_clock` signal clocks the resynchronization register. The `resynch_clock` signal can come from the system clock, the write clock, or the write PLL clock.

Figure 57. 13 QDRII+ & QDRII SRAM Read Data Path in Stratix & Stratix GX Devices



Memory Timing Parameters

The timing relationship of data (Q) with respect to the echo clock, CQ is governed by the t_{CQHQV} and t_{CQHQX} parameters. For the QDRII memory device under consideration, this timing parameter is ± 350 ps.

FPGA Timing Parameters

When the echo clock, CQ, is fed into the PLL for read capture, uncertainties introduced on this clock include jitter, phase-shift error, and compensation error. While jitter and phase-shift error parameters have been defined before, the compensation error is a measure of the PLL's ability to regenerate a clock output that tracks the input reference. For the source-synchronous mode of the PLL, this parameter is typically $t_{PLL\_COMP\_ERROR}$ = ± 100 ps.

For more information about PLL specifications, refer to the chapter *PLLs in Stratix II & Stratix II GX Devices* in volume 2 of the *Stratix II Device Handbook*.

PLL-based read implementation uses a single global clock network to distribute the phase-shifted clock signal to DQ capture registers in the IOE. The difference in clock arrival times to these registers (clock skew) is modeled in the Quartus II software and is reflected in the minimum/maximum propagation delays for the clock. Additionally, the Quartus II software models the package trace delays for every pin in the device. You therefore do not account for such skews separately in our timing margin analysis. The extracted minimum/maximum clock and data delays account for these uncertainties.

Table 10 shows the read timing margin calculation for a PLL-based QDRII SRAM interface using an EP2S90C3 device.

Table 10. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP2S90F1020C3 Using a Read PLL (Part 1 of 3)

	Parameter	200 MHz		Description
		Fast Corner Model	Slow Corner Model	
Memory Specifications *(1)*	$t_{KHK\#H}$	2.200	2.200	K clock rising edge to K# clock rising edge
	t_{CQHQV}	0.350	0.350	Clock-to-out time for read data with respect to echo clock
	t_{CQHQX}	–0.3350	–0.350	Read data hold with respect to echo clock

Table 10. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP2S90F1020C3 Using a Read PLL (Part 2 of 3)

	Parameter	200 MHz		Description
		Fast Corner Model	Slow Corner Model	
FPGA Specifications *(2)*	t_{PLL_JITTER}	0.125	0.125	Stratix II PLL output jitter
	t_{PLL_PSERR}	0.030	0.030	Stratix II PLL phase-shift error
	$t_{PLL\_COMP\_ERROR}$	0.100	0.100	PLL compensation error (high bandwidth
	Clock delay (minimum) *(3)*	1.535	2.125	Minimum CQ/CQ# pin to IOE register delay from Quartus II (with 75° PLL-based phase shift)
	Clock delay (maximum) *(3)*	1.569	2.170	Maximum CQ/CQ# pin to IOE register delay from Quartus II (with 75° PLL-based phase shift)
	Data Delay (minimum) *(3)*, *(4)*	0.599	0.988	Minimum Q pin to IOE register delay from the Quartus II software
	Data Delay (maximum) *(3)*, *(4)*	0.659	0.988	Maximum Q pin to IOE register delay from the Quartus II software
	μt_{SU}	0.068	0.122	Intrinsic setup time of the IOE register (rounded up)
	μt_{H}	0.037	0.072	Intrinsic hold time of the IOE register (rounded up).
Board Specifications	t_{EXT}	0.020	0.020	Board trace variations on the DQ and DQS lines
Timing Calculations	t_{EARLY_CLOCK}	1.280	1.870	Earliest possible clock edge after DQS phase-shift circuitry and uncertainties: minimum clock delay – t_{PLL_JITTER} – t_{PLL_PSERR} – $t_{PLL\_COMP\_ERROR}$
	t_{LATE_CLOCK}	1.824	2.435	Latest possible clock edge after DQS phase-shift circuitry and uncertainties: maximum clock delay – t_{PLL_JITTER} + t_{PLL_PSERR} + $t_{PLL\_COMP\_ERROR}$
	$t_{EARLY\_DATA\_INVALID}$	2.449	2.778	Time for earliest data to become invalid for sampling at FPGA flop: $t_{KHK\#H}$ + t_{CQHQX} + minimum data delay
	$t_{LATE\_DATA\_VALID}$	1.009	1.338	Time for latest data to become valid for sampling at FPGA flop: t_{CQHQV} + maximum data delay

Table 10. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP2S90F1020C3 Using a Read PLL (Part 3 of 3)

	Parameter	200 MHz		Description
		Fast Corner Model	Slow Corner Model	
Results	Read setup timing margin	0.183	0.390	$t_{EARLY\_CLOCK} - t_{LATE\_CLOCK} - t_{LATE\_DATA\_VALID} - \mu t_{SU} - t_{EXT}$
	Read hold timing margin	0.568	0.261	$t_{EARLY\_DATA\_INVALID} - t_{LATE\_CLOCK} - \mu t_{H} - t_{EXT}$
	Total margin	0.751	0.651	Setup + hold-time margin

Notes for Table 10:
(1) The memory numbers used here for 200 MHz come from the Cypress CY7C1315BV18 data sheet, Rev. B.
(2) PLL phase shift is user defined and adjustable if you need to balance the setup and hold time margins.
(3) These numbers are from the Quartus II software, version 6.0 using the Altera IP MegaCore function 1.3.0 for a 36-bit QDRII SRAM interface.

Table 11 shows the read timing margin calculation for a PLL-based QDRII interface using an EP1S25C5 device.

Table 11. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP1S25C5 Using a Read PLL (Part 1 of 3)

	Parameter	200 MHz		Description
		Fast Corner Model	Slow Corner Model	
Memory Specifications *(1)*	$t_{KHK\#H}$	2.200	2.200	K clock rising edge to K# clock rising edge
	t_{CQHQV}	0.350	0.350	Clock-to-out time for read data with respect to echo clock
	t_{CQHQX}	–0.350	–0.350	Read data hold with respect to echo clock

Table 11. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP1S25C5 Using a Read PLL (Part 2 of 3)

	Parameter	200 MHz		Description
		Fast Corner Model	Slow Corner Model	
FPGA Specifications *(2)*	t_{PLL_JITTER}	0.100	1.000	Stratix II PLL output jitter
	t_{PLL_PSERR}	0.050	0.050	Stratix II PLL phase-shift error
	Clock delay (minimum) *(3)*	1.910	1.974	Minimum K pin to IOE register delay from Quartus II (with 75° PLL-based phase shift)
	Clock delay (maximum) *(3)*	1.980	2.117	Maximum K pin to IOE register delay from Quartus II (with 75° PLL-based phase shift)
	Data Delay (minimum) *(3)*	0.516	1.089	Minimum D pin to IOE register delay from the Quartus II software
	Data Delay (maximum) *(3)*	0.516	1.089	Maximum D pin to IOE register delay from the Quartus II software
	μt_{SU}	0.133	0.280	Intrinsic setup time of the IOE register (rounded up)
	μt_{H}	0.0332	0.068	Intrinsic hold time of the IOE register (rounded up)
Board Specifications	t_{EXT}	0.020	0.020	Board trace variations on the DQ and DQS lines
Timing Calculations	t_{EARLY_CLOCK}	1.780	1.844	Earliest possible clock edge after DQS phase-shift circuitry and uncertainties: minimum clock delay – t_{PLL_JITTER} – t_{PLL_PSERR}
	t_{LATE_CLOCK}	2.110	2.247	Latest possible clock edge after DQS phase-shift circuitry and uncertainties: maximum clock delay + t_{PLL_JITTER} + t_{PLL_PSERR}
	$t_{EARLY\_DATA\_INVALID}$	2.316	2.889	Time for earliest data to become invalid for sampling at FPGA flop: $t_{KHK\#H}$ + t_{CQHQX} + minimum data delay
	$t_{LATE\_DATA\_VALID}$	0.916	1.489	Time for latest data to become valid for sampling at FPGA flop: t_{CQHQV} + maximum data delay

Table 11. Read Timing Analysis for 200-MHz QDRII SRAM Interface in EP1S25C5 Using a Read PLL (Part 3 of 3)

	Parameter	200 MHz		Description
		Fast Corner Model	Slow Corner Model	
Results	Read setup timing margin	0.711	0.055	$t_{EARLY\_CLOCK} - t_{LATE\_CLOCK} - t_{LATE\_DATA\_VALID} - \mu t_{SU} - t_{EXT}$
	Read hold timing margin	0.154	0.554	$t_{EARLY\_DATA\_INVALID} - t_{LATE\_CLOCK} - \mu t_{H} - t_{EXT}$
	Total margin	0.865	0.609	Setup + hold-time margin

Notes for Table 11:
(1) The memory numbers used here for 200 MHz come from the Cypress CY7C1315BV18 data sheet, Rev. B.
(2) PLL phase shift is user-defined and adjustable if you need to balance the setup and hold time margins.
(3) These numbers are from the Quartus II software, version 6.0 using the Altera IP MegaCore function 1.3.0 for a 36-bit QDRII SRAM interface.

Write Data Timing Analysis

Timing margin analysis for write data and address/control signals are very similar. This section analyzes timing for the write data signals. You can use the same approach to repeat this for the address/control signals.

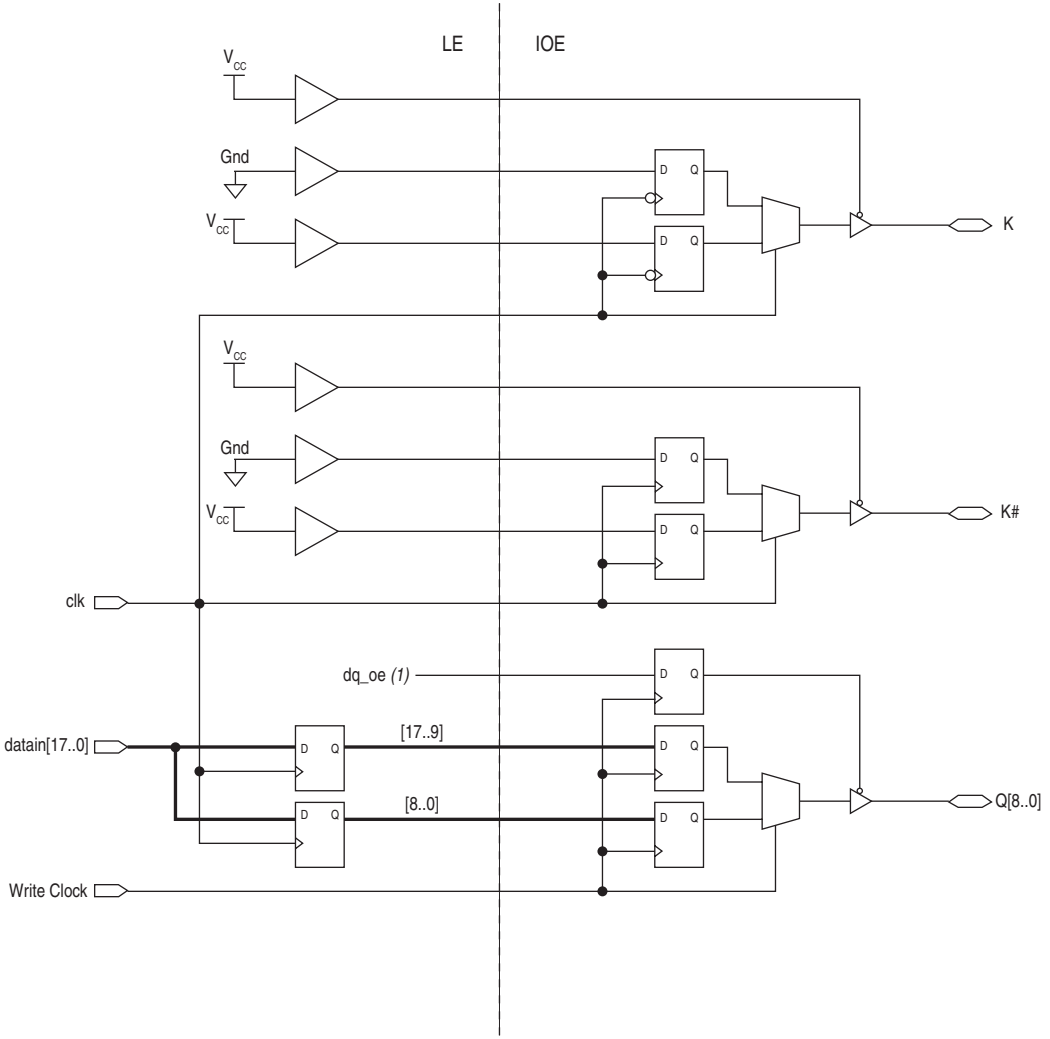
For write operations, the QDRII and QDRII+ memory requires the input clock (K and K#) to be center-aligned with the data bus (D). This is implemented in Stratix II using the PLL phase-shift feature. Two output clocks are created from the PLL, with a relative 90° phase offset. The leading clock edge clocks out the D write data output pins to the memory, while the lagging clock edge generates the input clock to the memory, K/K#.

Figure 58 shows the write side clocks that the PLL generates. Table 12 lists these clocks.

Table 12. Write Side PLL Clocks

Clock	Description
System Clock	This is used in the data path to generate the command, address, and data write signals.
Write clock (90° shifted from system clock)	This is used for the memory controller and to generate K signals, K# signals, or both.

Figure 58. QDRII+ and QDRII Read Data Path in Stratix II Devices



Notes to Figure 58:

(1) The `dq_oe` is active low in silicon. However, the Quartus II software implements it as active high and adds the inverter automatically during compilation.

(2) The QDRII and QDRII+ devices have separate ports for data read and data. Connect Q of the QDRII+ or QDRII SRAM device to the DQ pins of the Stratix II device and connect `dq_oe` to GND. You can use any of the user I/O pins in I/O banks 3, 4, 7, or 8 to connect to the D pins of the QDRII or QDRII+ SRAM device.

Memory Timing Parameters

When writing to a memory, the FPGA must ensure that setup and hold times are met. These specifications (t_{HD} and t_{SD}) are obtained from the QDRII SRAM data sheet (400 ps for the 200-MHz QDRII example). Additionally, the FPGA must provide a memory clock (K/K#) that meets the clock's high and low time specifications. Although the last parameter doesn't affect timing margins, the specifications must be met for successful memory operation.

FPGA Timing Parameters

The timing paths within the FPGA for the DQ and DQS outputs to memory are matched by design. Dedicated clock networks drive DDR I/O structures to generate DQ and DQS. This results in minimal skew between these outputs. These skew parameters are:

- Phase-shift error
- Clock skew
- Package skew

The two clock networks used are driven by the same PLL, but with a 90° relative phase shift. The 0° clock generates the write data, `Q[]`, commands and addresses, while a 90° clock generates the input clock, K and K#. Typical PLL uncertainties such as jitter and compensation error affect both clock networks equally. Therefore, these timing parameters do not affect write timing margins. However, because the clock generating the input clocks is phase-shifted, you account for the PLL phase-shift uncertainty (t_{PLL_PSERR} = ± 15 ps, listed in the *DC & Switching Characteristics* chapter of the *Stratix II Device Handbook*, while calculating the clocks arrival times at the memory pins.

The Quartus II software models the intra-clock skew, i.e. skew between nodes driven by the same dedicated clock network. However, skew between two such clock networks is not modeled and specified in the data sheet as an adder term. You should add this skew component to the propagation delays extracted from the Quartus II software.

For the 18-bit QDRII interface that spans two I/O banks in the top or bottom of the device, the clock skew adder between two clock networks is specified as ± 75 ps ($t_{CLOCK\_SKEW\_ADDER}$). We account for this uncertainty while calculating DQS arrival times at the memory pins.

The final skew component is package skew. As noted earlier, the Quartus II software models package trace delay for each pin on the device. Extracted propagation delays reflect any skew between output signals to the memory.

Table 13. Write Timing Analysis for 200-MHz QDRII SRAM Interface in EP2S60F1020C5ES Using DQS Mode (Part 1 of 2)

	Parameter	Fast Corner Model	Slow Corner Model	Description
Memory Specifications *(1)*	t_{SD}	0.400	0.400	Memory data setup requirement
	t_{HD}	–0.400	–0.400	Memory data hold requirement
FPGA Specifications *(2)*	t_{PLL_JITTER}	0.000	0.000	Does not affect margin because the same PLL generates both write clocks (0° and 90°)
	t_{PLL_PSERR}	0.030	0.030	PLL phase-shift error (on 90° clock output)
	$t_{CLOCK\_SKEW\_ADDER}$	0.050	0.050	Clock skew between two dedicated clock networks feeding I/O banks on the same side of the FPGA
	Minimum clock t_{CO} *(3)*	3.563	6.099	Minimum K/K# t_{CO} from the Quartus II software
	Maximum clock t_{CO} *(3)*	3.573	6.109	Maximum K/K# t_{CO} from the Quartus II software
	Minimum data t_{CO} *(3)*, *(4)*	2.281	4.815	Minimum D t_{CO} from the Quartus II software
	Maximum data t_{CO} *(3)*, *(4)*	2.340	4.905	Maximum D t_{CO} from the Quartus II software
Board Specifications	t_{EXT}	0.020	0.020	Board trace variations on the DQ and DQS lines

Table 13. Write Timing Analysis for 200-MHz QDRII SRAM Interface in EP2S60F1020C5ES Using DQS Mode (Part 2 of 2)

	Parameter	Fast Corner Model	Slow Corner Model	Description
Timing Calculations	t_{EARLY_CLOCK}	3.483	6.019	Earliest possible clock edge after DQS phase-shift circuitry and uncertainties: minimum clock delay – t_{PLL_JITTER} – $t_{CLOCK\_SKEW\_ADDER}$ – t_{PLL_PSERR}
	t_{LATE_CLOCK}	3.823	6.359	Latest possible clock edge after DQS phase-shift circuitry and uncertainties: t_{PLL_DCD} + $t_{OUTHALF\_JITTER}$ = 0.017 maximum clock delay + t_{PLL_JITTER} + t_{PLL_DCD} + $t_{OUTHALF\_JITTER}$ + $t_{CLOCK\_SKEW\_ADDER}$ + t_{PLL_PSERR}
	$t_{EARLY\_DATA\_INVALID}$	4.781	7.315	Time for earliest data to become invalid for sampling at FPGA flop: Launch CLK edge time + t_{CQHQX} + minimum data delay
	$t_{LATE\_DATA\_VALID}$	2.340	4.905	Time for latest data to become valid for sampling at FPGA flop: t_{CQHQV} + maximum data delay
Results	Read setup timing margin	0.723	0.694	t_{EARLY_CLOCK} – $t_{LATE\_DATA\_VALID}$ – t_{SD} – t_{EXT}
	Read hold timing margin	0.538	0.536	$t_{EARLY\_DATA\_INVALID}$ – t_{LATE_CLOCK} – t_{HD} – t_{EXT}
	Total margin	1.261	1.230	Setup + hold-time margin

Notes for Table 13:

(1) The memory numbers used here for 200-MHz come from Cypress CY7C1313BV18 data sheet, Rev. C.

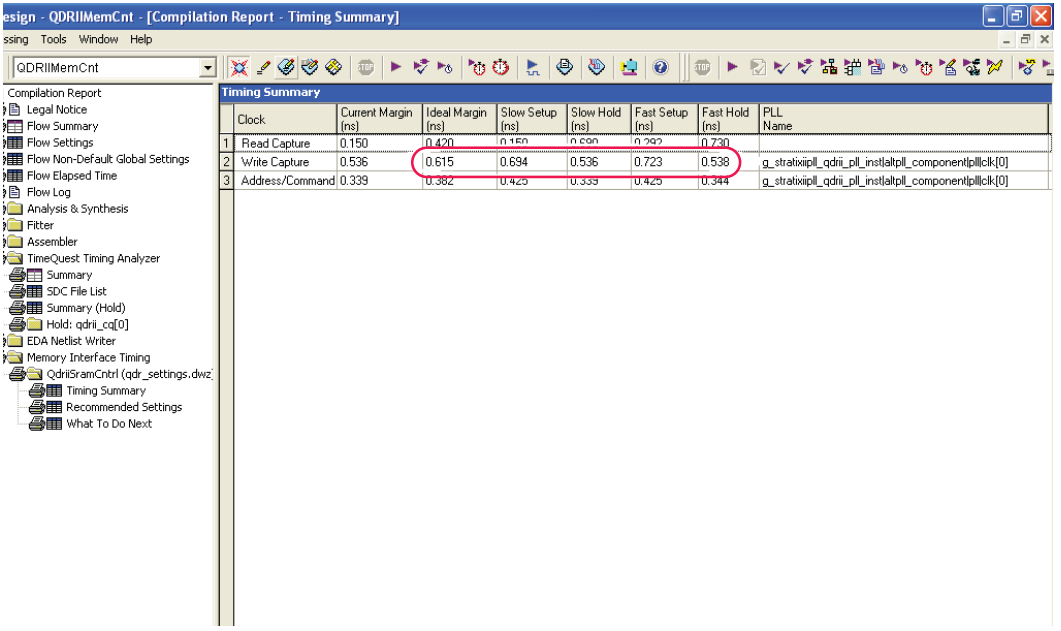
(2) This analysis is performed with FPGA timing parameters for an EP2S60F1020C5ES device. You should use this template to analyze timing for your preferred Stratix II density-package combination. For more information on FPGA specifications, refer to the *DC & Switching Characteristics* chapter in volume 1 of the *Stratix II Handbook*.

(3) These numbers are from the Quartus II software, version 7.2 SP1 using the Altera IP MegaCore function v7.2 SP1 for a 18-bit QDRII SRAM interface.

(4) Package trace skews are modeled by the Quartus II software.

As you can see, the paper calculation is correlating with DTW results as shown in Figure 59. TimeQuest is automatically calculates the margins.

Figure 59. Write Timing Analysis in the TimeQuest TIming Analyzer



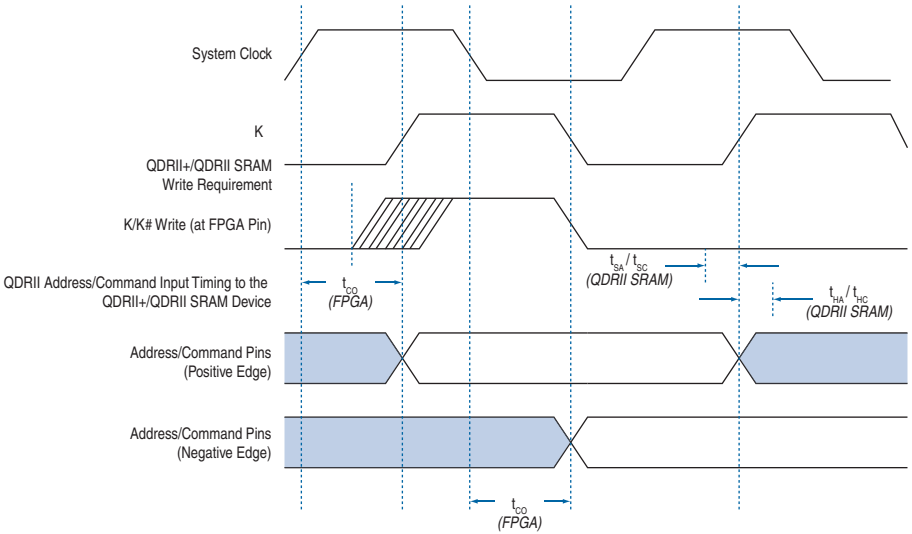
Command and Address Timing

Command and address signals are generated from the system clock (or another clock) in single data rate. The command and address signals must meet the setup and hold time requirement with respect to the rising edge of the K signal at the QDRII or QDRII+ SRAM device. The FPGA also generates the K signal from the system clock either directly or via an external clock buffer. Depending on the location of the registers for the commands and addresses, you may need to use a different system clock edge or add a phase shift on the system clock to make sure that these signals meet the setup and hold time requirement at the QDRII or QDRII+ SRAM device.

This section outlines the commands and addresses timing considerations. For example, if you place the command and address registers clocked by the system clock in the IOE, the command and address signals at the FPGA change at the same time as the K signal since they are both generated from the system clock either directly or via an external clock buffer. The delays from the IOE registers to the pins are then similar (exactly the same if there is no clock skew).

In this example, the command and address signals are edge-aligned with the K signal (if there are no variations in the package or board trace length of the system for the different signals). This means that the address and command signals cannot meet the setup time requirements of the QDRII+ or QDRII SRAM device. To meet the setup and hold time requirement, you must use the negative edge of the system clocks to generate the command and address signals if you want to place the command and address registers in the IOE. Figure 60 shows the command and address timing and how the system clock edge affects how the signals meet the QDRII or QDRII+ SRAM t_{SA}/t_{SC} and t_{HA}/t_{HC} requirements.

Figure 60. Command and Address Timing Using the IOE Registers



Note to Figure 60:
(1) In this waveform, there is no clock skew or any variations in package and board trace length.

References

- *AN 392: Implementing Multiple Legacy DDR/DDR2 SDRAM Controller Interfaces*
- *AN 449: External Memory Interface Design Guidelines for Stratix II, Stratix II GX, and Arria GX Devices*
- *DDR Timing Wizard User Guide*
- *External DDR Memory PHY Interface (ALTMEMPHY) User Guide*
- *External Memory Interfaces in Stratix II and Stratix II GX Devices* in the *Stratix II Device Handbook*
- *External Memory Interfaces in Stratix & Stratix GX Devices* in the *Stratix Device Handbook*
- *QDRII SRAM Controller MegaCore Function User Guide*
- *TB 091: External Memory Interface Options for Stratix II Devices*

Revision History

Table 14 shows the revision history for this application note.

Table 14. Document Revision History (Part 1 of 2)

Date and Document Version	Changes Made	Summary of Changes
May 2008 v5.1	Fixed code in Example 2.	—
March 2008 v5.0	Added example design and summarized technical text per recent Application Note standards.	Significant new content added. Significant reorganization of existing content.
April 2006 v4.0	• Changed the title of the application note from *AN 326: Interfacing QDRII SRAM With Stratix II, Stratix & Stratix GX Devices* to *AN 326: Interfacing QDRII & QDRII+ SRAM With Stratix II, Stratix & Stratix GX Devices* • Updated "Introduction" section. • Updated "QDRII & QDRII+ SRAM Overview" section. • Updated "Altera Memory Controller IP" section. • Updated "Setup & Hold Margins Calculations" section. • Updated Figure 13. • Updated Figure 14. • Updated Tables 6, 7, 8, and 11.	—
November 2005 v3.2	• Updated Figure 3 and Figure 5. • Updated "Interface Architecture" section.	—

Table 14. Document Revision History (Part 2 of 2)

Date and Document Version	Changes Made	Summary of Changes
September 2005 v3.1	• Updated PLL-based implementation information in Table 1. • Updated Quartus II software version in Tables 1 through 3. • Updated Figure 5. • Updated Figure 6. • Updated Figure 11. • Updated Table 6. • Updated the Setup & Hold Margins Calculations section. • Updated Table 7. • Updated the FPGA Timing Parameters section. • Updated the Write Data Timing Analysis section.	—
May 2005 v.3.0	• Reorganized major headings in document to be consistent with other memory application notes presentations. • Updated read and write table information. • Removed redundant sections.	—
October 2004 v2.0	• No information available	—
February 2004 v1.0	• Initial Release.	—

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