QDR SRAM Controller Reference Design

in APEX II Devices

October 2001, ver. 2.0 Application Note 133

Introduction

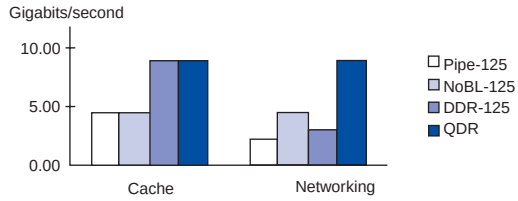
The explosive growth of the Internet has boosted the demand for high-speed data communications systems that require fast processors and high-speed interfaces to peripheral components. While the processors in these systems have improved in performance, static memories have not kept pace. New SRAM architectures are evolving to support the high throughput requirements of current systems. One such architecture is quad data rate (QDR) SRAM, which can provide bandwidth improvements more than four times greater than other SRAM architectures.

Most existing SRAM solutions were designed for PCs and have interfaces that transfer data efficiently for PC-type single input/output (I/O) applications. In contrast, most communications applications require continuous data transfer between the SRAM and the memory controller (e.g., continuous transitions between read and write cycles through the memory). Single I/O devices like standard synchronous pipelined SRAMs do not perform well in these applications.

The QDR Consortium—comprised of Cypress Semiconductor, Hitachi, Integrated Device Technology, Inc., Micron Technology, NEC, and Samsung—designed the QDR SRAM architecture for high-performance communications systems such as routers and ATM switches. QDR SRAMs are designed to handle the transfer of four data words through the SRAM in a single clock cycle. The memory provides simultaneous reads and writes, as well as zero latency and increased data throughput, guaranteeing simultaneous access to the same address location.

Figure 1 compares the performance of QDR SRAMs versus other SRAM architectures. The comparison assumes that the QDR interface operates at 166.67 MHz. The figure shows that the QDR SRAM outperforms other memory architectures by up to four times in a networking application.

Figure 1. Performance Comparison



This application note describes the functionality of the QDR SRAM controller reference design and explains how to synthesize, place-and-route, and simulate it.

Reference Design Description

When using QDR SRAM in a system, a memory controller generates all of the signals needed for the SRAM and serves as the interface between the SRAM and the rest of the system. Altera® APEX™ II devices have been designed to connect at high speeds with memories such as QDR SRAM. I/O buffers, which are compliant with the HSTL Class I I/O standard, enable fast data transfers between the APEX II device and the QDR SRAM. Additionally, the high-density APEX II devices provide up to 67,200 logic elements (5 million system gates) that can be utilized for custom logic connecting to the memory controller. These advantages make APEX II devices the ideal solution for memory-intensive applications.

The Altera QDR SRAM controller reference design shows one implementation of a QDR SRAM controller using the EP2A15F672C7 device. You can use this design as:

- A memory interface module, which you can incorporate into a larger system-on-a-programmable-chip (SOPC) design
- An example implementation that you can reference when targeting a different APEX II device

The main advantage of QDR SRAM is its ability to be written and read independently on both the rising and falling edges of the clock. This ability quadruples the throughput of the SRAM. To take advantage of the high throughput of QDR SRAM, the Altera QDR SRAM controller uses the double data rate (DDR) I/O registers in the APEX II device. These structures allow the APEX II device to input and output data on both the positive and negative clock edges.

Refer to the *APEX II Programmable Logic Device Family Data Sheet* for a detailed description of the DDR I/O feature.

The reference design implements several optional pipeline registers in the core of the APEX II device. These registers maintain 166.67-MHz operation for the standalone controller. If, however, you add custom logic to the APEX II device in addition to the controller, Altera recommends that you remove these pipeline stages to reduce latency through the controller.

The reference design provides an interface to the Cypress CY7C1302V25-167 device, a 9-MByte pipelined QDR SRAM with a 2-word burst on all reads and writes. You can implement controllers for larger QDR SRAMs, QDR SRAMs with a 4-word burst, or QDR SRAMs from other vendors with little or no change to the reference design.

Controller Structure & Operation

Figure 2 shows a block diagram of the controller reference design.

Figure 2. Block Diagram

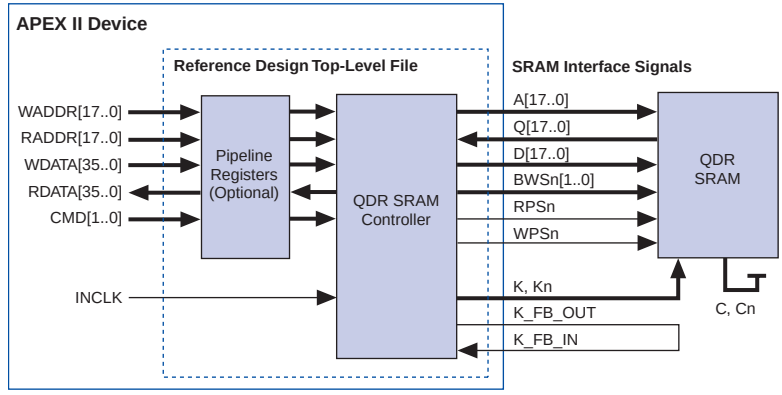


Table 1 describes the function of each controller pin on the QDR SRAM interface.

Table 1. QDR SRAM Interface Signals

Signal Type	Signal Direction	Signal Name	Description
Clock	Output	K, Kn	K and Kn are output by the APEX II device and are used as clock inputs to the QDR SRAM. All transactions are initiated synchronously on the rising edge of K or Kn. These clocks are generated from the rising and falling edges of WRITE_CLK_90.
		K_FB_OUT	K_FB_OUT is fed back to the controller as K_FB_IN at the QDR SRAM to imitate the data flight times to and from the QDR SRAM. (See "Clock Generation" on page 8 for more details on the clocking scheme)
	Input	K_FB_IN	The controller uses the K_FB_IN clock to generate READ_CLK for clocking in data from the QDR SRAM.
Control	Output	RPSn	This active-low read port select signal is clocked out on the rising edge of WRITE_CLK and sampled by the QDR SRAM on the rising edge of K.
		WPSn	This active-low write port select signal is clocked out on the rising edge of WRITE_CLK and sampled by the QDR SRAM on the rising edge of K.
		BWSn[1..0]	This active-low byte write select signal is clocked out on the rising edge of WRITE_CLK and sampled by the QDR SRAM on the rising edge of K. You can use this signal to individually select which bytes are written or read. For the reference design, both bytes are active on writes. You can add logic for individual byte writes if desired.
Address	Output	A[17..0]	The QDR SRAM uses the same address signals for the read and write ports. The address inputs to the QDR SRAM are clocked out of the controller using WRITE_CLK and sampled on the rising edge of K for reads and on the rising edge of Kn for writes.
Data	Input	Q[17..0]	Read data output from QDR SRAM. Two words can be transferred from the QDR SRAM during each clock cycle because the QDR SRAM outputs data on the rising edges of both K and Kn. On the subsequent falling edge of READ_CLK, the controller captures data output by the SRAM on the rising edge of K. The controller captures data output on the rising edge of Kn on the subsequent rising edge of READ_CLK.
	Output	D[17..0]	Write data input to the QDR SRAM. The controller clocks data out on the rising and falling edges of WRITE_CLK and the QDR SRAM captures it on the next rising edges of K and Kn. Therefore, two words can be transferred to the QDR SRAM during each clock cycle.

Table 2 shows the user interface signals for the controller.

Table 2. User Interface Signals

Signal Type	Signal Direction	Signal Name	Description
Clock	Input	INCLK	User input clock, which is used to generate WRITE_CLK and WRITE_CLK_90. This clock is 166.67 MHz in the reference design.
Control	Input	CMD[1..0]	User command input, which is sampled on the rising edge of WRITE_CLK.
Address	Input	RADDR[17..0] WADDR[17..0]	User read and write address inputs, which are sampled on the rising edge of WRITE_CLK.
Data	Input	WDATA[35..0]	Data input for write operations, which is sampled on the rising edge of WRITE_CLK
Data	Output	RDATA[35..0]	Data output from read operations; it is output from the controller on the rising edge of READ_CLK.

Table 3 shows the commands accepted by the controller.

Table 3. Controller Commands

Command	Code (CMD[1..0])
Idle	00
Read	01
Write	10
Read/Write	11

The Altera QDR SRAM controller is a synchronous interface. Because the read and write data paths for the controller are independent, the controller can perform reads and writes together or separately.

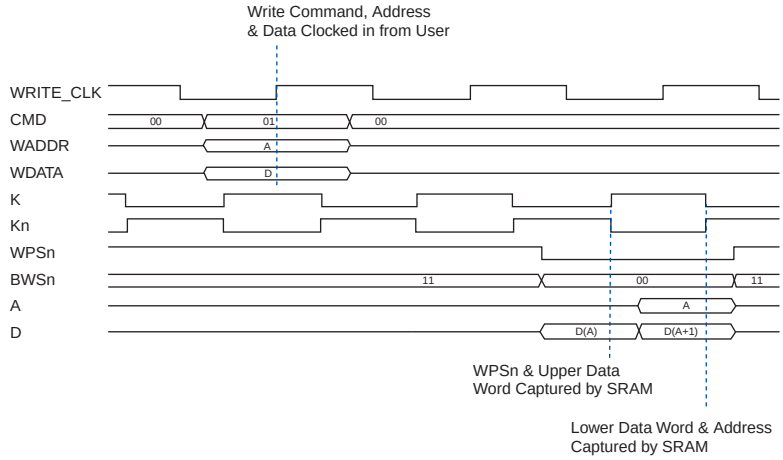
To use the controller, you must first provide an input clock (INCLK), which is fed into two APEX II phase locked loops (PLLs). Once the PLLs have locked onto the input clock signal, they generate the WRITE_CLK and WRITE_CLK_90 clocks for the controller. A third PLL generates READ_CLK. For more information, see "Clock Generation" on page 8.

At the rising edge of WRITE_CLK, the controller should receive an idle, read, write, or read/write command, as shown in Table 3. The controller should receive the appropriate read address (RADDR) simultaneously with a read or read/write command. Similarly, the controller should receive the write address (WADDR) and write data (WDATA) simultaneously with a write or read/write command.

At the SRAM side of the controller, the APEX II DDR I/O registers output data, address, and control signals as well as the K and Kn clocks, which are generated using WRITE_CLK_90. WRITE_CLK_90 is also used to output another clock, K_FB_OUT, which is fed back to the controller as K_FB_IN. K_FB_IN is sent to a third PLL, which generates READ_CLK.

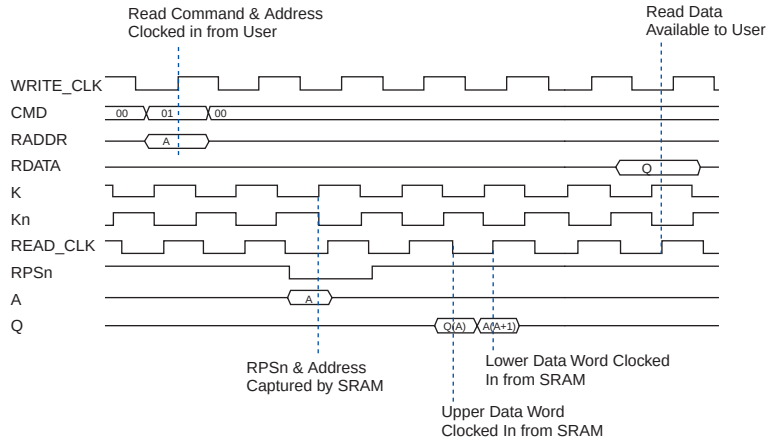
For write operations, the upper data word (D) is output and the write port select (WPSn) and byte write select (BWSn) signals are asserted on the rising edge of WRITE_CLK. The SRAM captures these signals on the rising edge of K. On the falling edge of WRITE_CLK, the address (A) and the lower data word are sent to the SRAM. These signals are captured on the rising edge of Kn. Figure 3 shows the functionality of the controller for write operations.

Figure 3. Write Cycle Waveform



For read operations, the address (A) and read port select (RPSn) signals are output on the rising edge of WRITE_CLK. After the SRAM captures this information on the rising edge of K, the upper and lower data words at that address are driven to Q on the next rising edges of K and Kn, respectively. The controller captures the words on the READ_CLK clock's falling and rising edges, respectively. The data is then sent back to the read data (RDATA) ports where it is output on the rising edge of READ_CLK. Figure 4 shows the functionality of the controller for read operations.

Figure 4. Read Cycle Waveform



Constraints

To ensure proper pin placement and I/O buffer configuration for the controller, you must use a Quartus II constraint file with the following assignments:

- The following outputs from the controller must be placed on column pins: `A[17..0]`, `D[17..0]`, `RPSn`, `WPSn`, `BWSn[1..0]`, `K`, `Kn`, and `K_FB_OUT`.
- `K`, `Kn`, and `K_FB_OUT` must be assigned to consecutive column pins. There should be two reserved pins on each side of the three consecutive pins to preserve signal integrity. Search for "Reserving a Pin with the Assignment Organizer" in Quartus II Help for instructions on how to make this assignment.
- You can place the `Q[17..0]` inputs to the controller on either column or row pins.
- Add the assignment Decrease Input Delay to Input Register = On to the `Q[17..0]` inputs to minimize the t_{SU} time on those pins.

The QDR SRAM interface requires the use of the HSTL I/O standard. Characterization data has shown that APEX II devices can drive out and receive HSTL I/O signals at greater than 166.67 MHz.

Refer to *AN 117: Using Selectable I/O Standards in Altera Devices* for more details on HSTL.

To implement the HSTL I/O interface, perform the following steps.

1. Use the **Assignment Organizer** in the Quartus II software to make an I/O standard assignment of HSTL Class I on all APEX II pins that interface with the SRAM (A[17..0], D[17..0], Q[17..0], RPSn, WPSn, BWSn[1..0], K, Kn, K_FB_IN, and K_FB_OUT).

2. To implement the reference voltage pins (VREF pins) required by the HSTL Class I standard, place VREF pin assignments in the Quartus II software on all I/O banks that hold the interface pins. Refer to *AN 117: Using Selectable I/O Standards in Altera Devices* for further instruction on how to place HSTL VREF pins.

The reference design shows examples of the appropriate pin placement as well as the I/O standard and VREF assignments for HSTL. You can view the assignments the reference design needs by viewing the current assignments floorplan in the Quartus II software.

APEX II devices other than the EP2A15F672C7 may require additional location assignments to ensure high-speed operation of the controller.

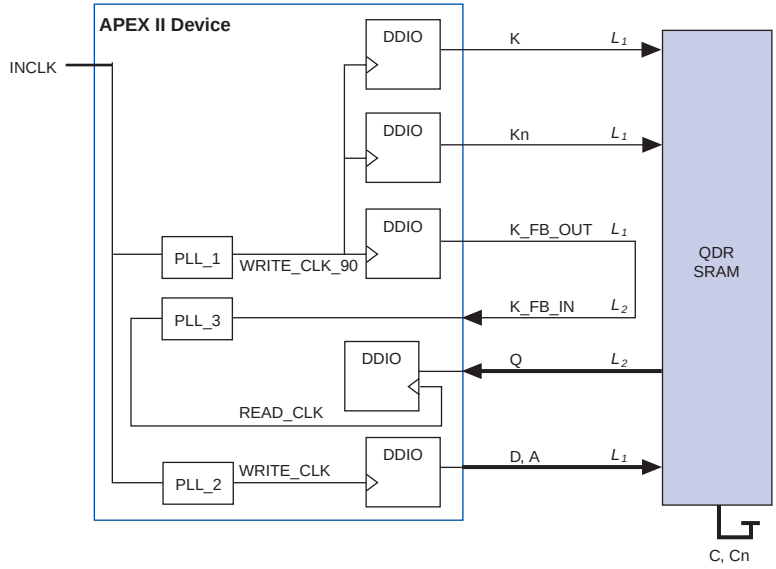
Clock Generation

The controller clocking scheme maintains consistent and robust high-frequency operation. The Altera QDR SRAM controller reference design requires three PLLs and two global clock resources in the APEX II device to perform clock generation. The design uses the following clocks:

- INCLK—Input clock from the user
- WRITE_CLK and WRITE_CLK_90—True and 90-degree-shifted controller clocks
- K and Kn—SRAM clocks
- K_FB_IN and K_FB_OUT—Controller feedback clock
- READ_CLK—Read data capture clock

Figure 5 on page 9 shows how these clocks are generated and used for the QDR SRAM interface.

***Figure 5. Clock Generation** Note (1)*



Note:

(1) All traces marked L_1 should be of equal length; all traces marked L_2 should be of equal length.

You must supply an input clock, nominally 166.67 MHz, to the design. This clock feeds two on-chip PLLs, which generate the true (non-phase-shifted) clock for data and address (`WRITE_CLK`) and the 90-degree-shifted clock (`WRITE_CLK_90`) for the `K` and `Kn` outputs.

The controller uses `WRITE_CLK_90` and the APEX II DDR I/O registers with the inputs tied to V_{CC} and ground to generate a differential clock signal for the QDR SRAM device. The result is a clock signal (`K`) and a 180-degree-phase-shifted clock signal (`Kn`), each with the same frequency as the `WRITE_CLK`.

The APEX II device outputs `K` and `Kn` and sends them to the SRAM along with the data, address, and control lines. This action negates the effect of signal skew on the write and read request operations, because the propagation delays for `K` and `Kn` from the APEX II device to the SRAM are equal to the delays on the data signals. For the controller to operate properly, you should equalize the trace lengths (and therefore the flight times) of the data in, address, and control signals with the `K` and `Kn` clocks.

Because data is transported both to and from the SRAM, you should use a similar strategy to eliminate signal skew on read operations. One method is to feed the K clock back into the controller and then use this feedback clock to capture the read data. In this case, the length of the feedback trace between the SRAM and the controller should be equal to the read data (Q) trace length. The disadvantage of this method is that the additional tap on the K trace can cause skew between the K and Kn clock signals. Because of this issue, the reference design outputs an additional clock, K_FB_OUT, to emulate the effect of feedback from K. For this method to work properly, the feedback trace length must match the sum of the D and Q trace lengths.

Additionally, Altera recommends that you place the APEX II device adjacent to the SRAM on the circuit board. This positioning keeps the trace length to a minimum and further minimizes any skew caused by board delay.

Timing

Because data is transferred between the controller and the SRAM at high speeds, you should take special care to avoid setup or hold violations for the SRAM or APEX II device. This section discusses the timing issues that may arise when designing a high-speed QDR SRAM interface.

Write Cycle

When designing for correct write-cycle timing, meeting the setup and hold requirements of the SRAM is the primary concern. Setup and hold specifications for the CY7C1302V25-167 device are 0.7 ns each.

The controller drives both the QDR clock and data signals; therefore, the clock-to-output delay from the APEX II device pins is the same for both sets of pins. As determined by characterization, the clock-to-output delay from the APEX II pins under worst-case temperature and voltage conditions can range from 2.628 ns to 3.253 ns, depending on the pin placement. The board delays for the clock and data are assumed to be roughly equal, because the signal trace lengths should be matched (as discussed in "Clock Generation" on page 8).

At a clock speed of 166.67 Mhz, the bit period—or length of time between each data bit—is 3 ns. Because K and Kn are clocked out using WRITE_CLK_90, while data and address are clocked out using WRITE_CLK, there is a timing cushion of one half of the bit period each way to meet setup and hold times at the SRAM.

The following calculations apply for 166.67-MHz controller-to-SRAM data transfers. The calculations allow for the characterized difference between the clock t_{CO} and data/address t_{CO}, and up to 0.1 ns of board skew.

[t_{CO}(APEX II Clock) – t_{CO}(APEX II Data and Address)] + Board Skew (Clock – Data) + t_{SU} (SRAM) < (Bit Period)/2

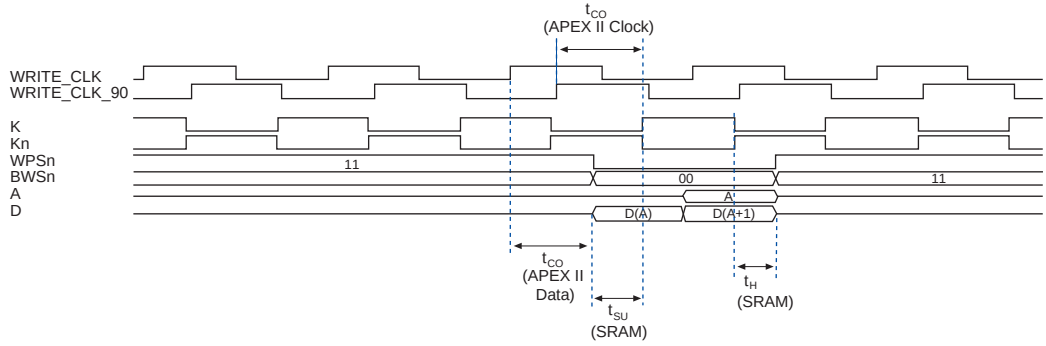
[3.253 ns – 2.628 ns] + 0.1 ns + 0.7 ns = 1.425 ns < 1.5 ns

[t_{CO}(APEX II Data and Address) – t_{CO}(APEX II Clock)] + Board Skew (Clock – Data) + t_{H}(SRAM) < (Bit Period)/2

[3.253 ns – 2.628 ns] + 0.1 ns + 0.7 ns = 1.425 ns < 1.5 ns

Figure 6 shows the write cycle timing waveform for the SRAM interface pins at 166.67 MHz.

Figure 6. Write Cycle Timing Waveform



In addition to setup and hold times, an additional concern is the clock-to-clock skew between K and Kn (t_{KHKH}). The 167-MHz QDR SRAM specification allows for up to 0.6 ns difference between the rising edges of K and Kn. Because APEX II clock-to-out times can vary with pin position, K and Kn should be placed on adjacent pins and their t_{CO} times should be checked carefully in the Quartus II Timing Analyzer. Characterization has shown that the controller meets the t_{KHKH} specification when the K and Kn pins are adjacent.

Read Cycle

The read request and address signals are sent to the SRAM along with the K and Kn clocks in a similar manner as the write data. Therefore, the write timing parameters apply to these signals as well.

Additionally, when the SRAM sends read data to the controller, the design must meet the APEX II device setup and hold times. Altera has determined through device characterization that for the Q pins in the reference design, the worst-case setup times are 1.1 ns and worst-case hold times are -0.4 ns.

The clock-to-output specification for the SRAM determines the arrival time of the Q signal. For the CY7C1302V25-167 device, the maximum t_{CO} value is 2.5 ns and the minimum t_{CO} value (i.e., the data output hold time t_{DOH}) is 1.2 ns. Board delay can be largely ignored, because flight times for the K_FB signal and the Q bus are roughly equal. Regardless, the timing calculation allows for some skew between the clock and data lines.

The SRAM sends data out on the rising edge of K, and the controller captures it on the falling edge of READ_CLK. For a clock speed of 166.67 MHz, there is a window of 3 ns between the rising and falling edges. Subtracting the SRAM clock-to-output delay of 2.5 ns leaves only 0.5 ns of margin to cover the APEX II setup times and board and clock skew. Because 0.5 ns is insufficient, a PLL shifts READ_CLK to capture the data.

For the reference design, the following calculations apply for data transfers from the SRAM to the controller:

t_{CO} (SRAM) + Board Skew (K_fb – Data) + Clock Skew (K – Kn) + t_{SU} (APEX II) – Clock Shift (READ_CLK) < Bit Period

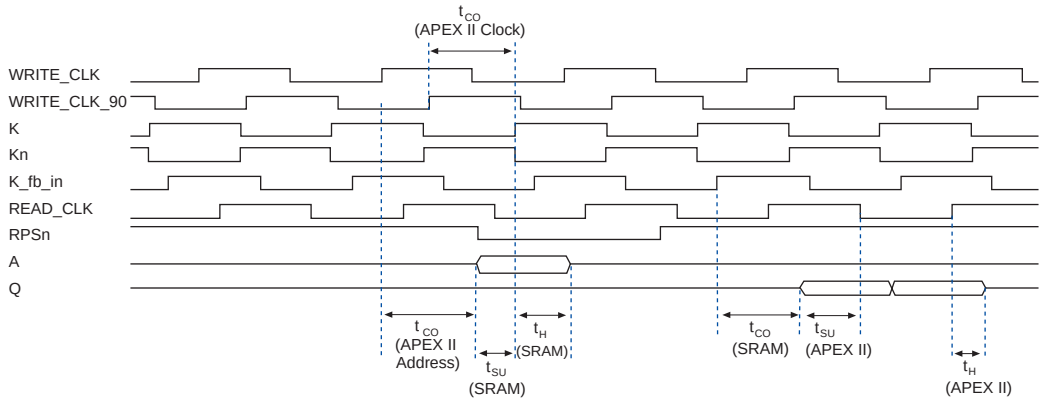
2.5 ns + 0.1 ns + 0.3 ns + 1.1 ns – 1.1 ns = 2.9 ns < 3 ns

t_{DOH} (SRAM) - Board Skew (K_fb - Data) - Clock Skew (K – Kn) – t_{H} (APEX II) – Clock Shift (READ_CLK) > 0 ns

1.2 ns – 0.1 ns – 0.3 ns – (-0.4 ns) – 1.1 ns = 0.1 ns > 0 ns

Figure 7 shows the read cycle timing waveform for the SRAM interface pins at 166.67 MHz.

Figure 7. Read Cycle Timing Waveform



Read/Write Cycle

The QDR SRAM controller has independent read and write paths. Therefore, timing does not change for a standalone read or write versus a combined read/write operation.

Getting Started

This section describes how to install the QDR SRAM controller reference design and walks through the design flow. This description is for the Verilog HDL version of the reference design; the VHDL version is similar, however, the filenames have different extensions.

Hardware & Software Requirements

To use the QDR SRAM controller reference design, you must have the following software installed on your system:

- Quartus II software version 1.1 Service Pack 1 or later
- LeonardoSpectrum-Altera software version 2001.1d or later, LeonardoSpectrum software version 2001.1d or later, or Synplify version 6.2.4 or later
- ModelSim software version 5.5c or ModelSim-Altera software version 5.5b or later

This walkthrough uses the LeonardoSpectrum Altera Edition software version 2001.1d and the ModelSim software version 5.5c on a PC running Windows NT.

Design Installation

Altera provides the QDR SRAM controller reference design as two executable files, one for VHDL and one for Verilog HDL. To install the files, perform the following steps:

You can download either the Verilog HDL or VHDL version of the reference design as an executable file from the Altera web site at http://www.altera.com.

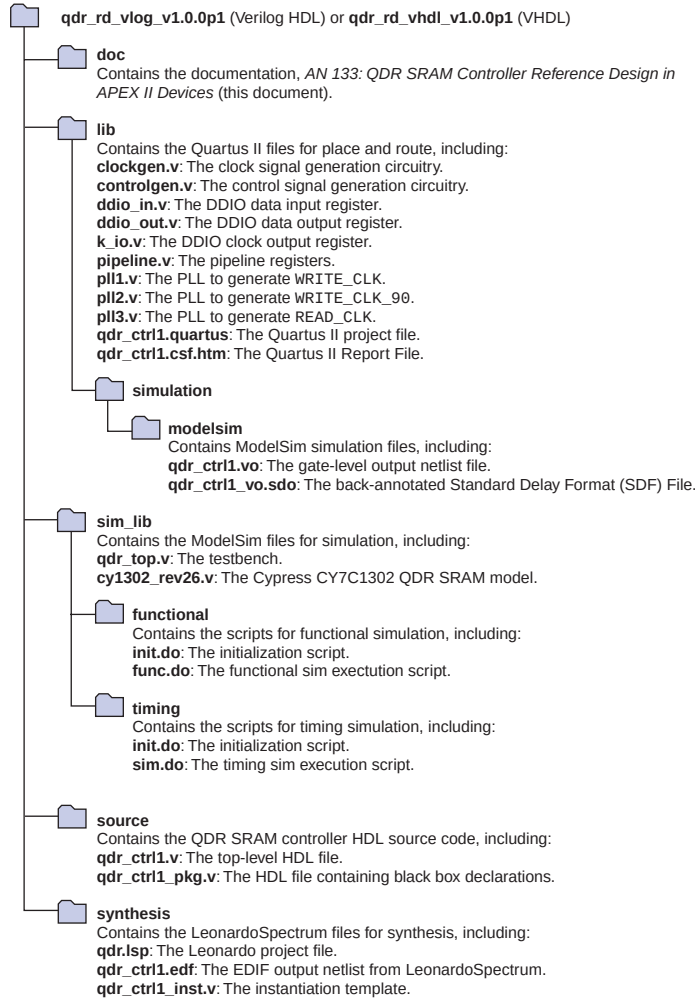
1. Save the executable file, **qdr_rd_vlog_v1.0.0p1.exe** (Verilog HDL) or **qdr_rd_vhdl_v1.0.0p1.exe** (VHDL), onto your hard disk. You can delete this file after you finish installing.

2. Double-click the **qdr_rd_vlog_v1.0.0p1.exe** (Verilog HDL) or **qdr_rd_vhdl_v1.0.0p1.exe** (VHDL) file in the Windows Explorer to launch the installer.

3. Follow the on-line instructions to complete installation.

Figure 8 shows the directory structure created by the reference design installer. It also describes selected files (Verilog HDL design files only; the VHDL files have similar functionality). The default installation directory is **c:\Altera\qdr_rd_vlog_v1.0.0p1** (Verilog HDL) or **c:\Altera\qdr_rd_vhdl_v1.0.0p1** (VHDL).

Figure 8. QDR SRAM Controller Directory Structure



Design Walkthrough

Altera provides the source files of the reference design, which you can use to synthesize, place-and-route, and simulate the design. This section walks you through the design flow for the reference design. The steps include:

- Synthesize in the LeonardoSpectrum Software
- Compile in the Quartus II Software
- Simulate in the ModelSim Software

This walkthrough describes the process for the Verilog HDL reference design files and assumes that the files are installed into the **c:\Altera\qdr_rd_vlog_v1.0.0p1** directory. If you installed the files into another directory or want to use the VHDL files (for which the default installation directory is **c:\Altera\qdr_rd_vhdl_v1.0.0p1**), adjust the path names accordingly.

The results for each step are included in the reference design; therefore, you do not need to perform each step unless you have altered the design files. For example, you can run a timing simulation without first compiling the design because the Quartus II software place-and-route results are shipped with the reference design.

Synthesize in the LeonardoSpectrum Software

Altera provides the reference design source files, which you can synthesize in a third-party tool before performing place-and-route. The following source files are included in the **source** directory when you install the reference design:

- **c:\Altera\qdr_rd_vlog_v1.0.0p1\source\qdr_ctrl1.v**—Top-level of the QDR SRAM controller.
- **c:\Altera\qdr_rd_vlog_v1.0.0p1\source\qdr_ctrl1_pkg.v**—Declarations for the black box modules.

The QDR SRAM controller, as shipped with the reference design, includes pipeline registers so that the design meets the $\mathbf{f_{MAX}}$ performance requirements. The source code (**qdr_ctrl1.v** for Verilog HDL or **qdr_ctrl1_pkg.vhd** for VHDL) includes the parameter `INCLUDE_PIPELINE_REGS`, which controls whether the extra pipeline stages are added to the write and read paths. You can set this parameter to `FALSE` to eliminate the pipeline stages.

The **synthesis** directory contains Exemplar LeonardoSpectrum-Altera project files, which you can use to synthesize the controller source code.

You can change the device or clock frequency in the LeonardoSpectrum software to customize the design. You do not need to make any special synthesis settings.

Follow the steps below to synthesize the design:

1. Run the LeonardoSpectrum software.
2. Choose **Open Project** (File menu).
3. Browse to the **c:\Altera\qdr_rd_vlog_v1.0.0p1\synthesis** directory.
4. Select the file **qdr.lsp** and click **Open**.
5. Choose **Quick Setup** (Tools menu).
6. Update the path for the input and output files to reflect the location in which you installed the design by adding the input and output files to the project. The files to add are:

   Input files:
   c:\Altera\qdr_rd_vlog_v1.0.0p1\source\qdr_ctrl1.v
   c:\Altera\qdr_rd_vlog_v1.0.0p1\source\qdr_ctrl1_pkg.v

   Output file:
   c:\Altera\qdr_rd_vlog_v1.0.0p1\synthesis\qdr_ctrl1.edf
7. Click the **Run Flow** button to begin synthesis.

Compile in the Quartus II Software

After you synthesize the design, you are ready to place-and-route the design. The **lib** directory contains the Quartus II software version 1.1 Service Pack 1 project files, including the constraint files needed for the design to meet the required clock frequencies and I/O timing in the EP2A15F672C7 device.

Altera provides several source files in the **lib** directory: **pll1.v**, **pll2.v**, **pll3.v**, **ddio_out.v**, **ddio_in.v**, **k_io.v**, **clockgen.v**, **controlgen.v**, and **pipeline.v**. These files are treated as black boxes in the synthesis tool and are synthesized by the Quartus II software.

- **pll1.v**, **pll2.v**, and **pll3.v** are phase-locked loop (PLL) instantiation files created by the Quartus II MegaWizard® Plug-In Manager. The files instantiate the parameterized `altclocklock` function, which generates a PLL in the APEX II device. In the reference design, **pll1.v**

generates WRITE_CLK, **pll2.v** generates WRITE_CLK_90, and **pll3.v** generates READ_CLK.

- **k_io.v**, **ddio_out.v**, and **ddio_in.v** are DDR I/O instantiation files created by the Quartus II MegaWizard Plug-In Manager. **k_io.v** generates the clocks, **ddio_out.v** generates the output data, and **ddio_in.v** generates the input data.
- **clockgen.v** and **controlgen.v** contain the clock signal (K, Kn, K_FB_IN) and control signal (BWSn, RPSn, and WPSn) generation circuitry, respectively.
- **pipeline.v** adds three pipeline stages to both the read and write paths so that the design meets the reference design f_{MAX} timing. You can remove the pipeline stages by changing a parameter in the source code. See "Synthesize in the LeonardoSpectrum Software" on page 16 for details.

To compile the Altera-provided project files, follow the steps below:

1. Run the Quartus II software.
2. Choose **Open Project** (File menu).
3. Browse to the **c:\Altera\qdr_rd_vlog_v1.0.0p1\lib** directory.
4. Select the project file **qdr_ctrl1.quartus** and click **Open**.
5. Choose **Compile Mode** (Processing menu).
6. Choose **Start Compliation** (Processing menu).

Simulate in the ModelSim Software

The **sim_lib** directory contains an HDL testbench file (**qdr_top.v**) that instantiates the QDR SRAM controller and the QDR SRAM model (**c:\Altera\qdr_rd_vlog_v1.0.0p1\sim_lib\cy1302_rev26.v**). The testbench implements four pipelined writes, four pipelined reads, a standalone write operation, a read/write operation, and a standalone read operation to demonstrate the functionality of the controller. The testbench adds delay to the board traces to model the propagation delay between the APEX II device and the QDR SRAM. You can model different board delay scenarios by changing these values.

Altera provides the following scripts to perform functional and timing simulation in the ModelSim software.

- **init.do**—This script creates the a work library and pre-compiles the correct simulation libraries for functional or timing simulation.
- **func.do**—This script, which is located in the **functional** subdirectory, compiles the controller source files, the model, and the testbench, and displays the appropriate waveforms.
- **sim.do**—This script, which is located in the **timing** subdirectory, compiles the gate-level HDL output netlist file generated by the Quartus II software (**c:\Altera\qdr_rd_vlog_v1.0.0p1\lib\simulation\modelsim\qdr_ctrl1.vo**), the model, and the testbench, and performs a timing simulation with a back-annotated Standard Delay Format (SDF) File (**c:\Altera\qdr_rd_vlog_v1.0.0p1\lib\simulation\modelsim\qdr_ctrl1_v.sdo**).

Before using the **init.do** script, you should update it so that the paths in the script point to the locations in which you installed the Quartus II software.

If you are using the ModelSim-Altera software, Altera recommends that you use the pre-compiled APEX II models instead of compiling the models provided with the Quartus II software. Before using the pre-compiled models, you must change the **init.do** script; refer to the ModelSim-Altera documentation for more information.

To perform functional simulation, perform the following steps:

1. Run the ModelSim version 5.5c software.
2. Change your working directory to the **c:\Altera\qdr_rd_vlog_v1.0.0p1\sim_lib\functional** directory.
3. Type the following commands in the Command Window:

```
do init.do ↵
do func.do ↵
```

To perform timing simulation, perform the following steps:

1. Copy the files in the **c:\Altera\qdr_rd_vlog_v1.0.0p1\lib\simulation\modelsim** directory to the **c:\Altera\qdr_rd_vlog_v1.0.0p1\sim_lib\timing** directory.

2. Run the ModelSim version 5.5c software.

3. Change your working directory to the **c:\Altera\qdr_rd_vlog_v1.0.0p1\sim_lib\timing** directory.

4. Type the following commands in the **Command Window**:

```
do init.do ↵
do sim.do ↵
```

For Verilog HDL simulation, the ModelSim software may show setup violations at the start of simulation as the initial values settle through the controller. However, there should be no setup violations during actual operation of the controller.

Instantiation within an SOPC Design

You can instantiate the QDR SRAM controller design files in an HDL file and integrate it into an SOPC design. This section describes the steps you should follow to integrate the controller into an SOPC design. Generally, you should use "Design Walkthrough" on page 16 as a reference when integrating the controller into your system.

This section describes the process for the Verilog HDL reference design files and assumes that the files are installed into the **c:\Altera\qdr_rd_vlog_v1.0.0p1** directory. If you installed the files into another directory or want to use the VHDL files (for which the default installation directory is **c:\Altera\qdr_rd_vhdl_v1.0.0p1**), adjust the path names accordingly.

Synthesis

You can synthesize the source files **c:\Altera\qdr_rd_vlog_v1.0.0p1\source\qdr_ctrl1.v** and **c:\Altera\qdr_rd_vlog_v1.0.0p1\source\qdr_ctrl1_pkg.v** in any tool that can generate an Altera-targeted netlist; however, you may need to make some minor changes to the source code, depending on which synthesis tool you use. Altera provides a template, **c:\Altera\qdr_rd_vlog_v1.0.0p1\synthesis\qdr_ctrl1_inst.v**, for instantiating the top-level file **c:\Altera\qdr_rd_vlog_v1.0.0p1\source\qdr_ctrl1.v** in your design.

Because synthesis tool results vary, you should always black box the source files in the **lib** directory in the synthesis tool and so that the source files are synthesized in the Quartus II software.

Place & Route

Before compiling in the Quartus II software, you must add the **lib** directory to your Quartus II project as a user library. Search for "User Libraries" in Quartus II Help for more information. You should also either copy the netlist file output by your synthesis tool into the **lib** directory or use the **Add Files to Project** command (Project menu) in the Quartus II software to add the file to your project.

To ensure that your design meets the QDR timing requirements, you should generate an appropriate constraint file as described in "Constraints" on page 7. Refer to the Quartus II project provided in the **lib** directory for example project assignments.

Simulation

The simulation testbench and scripts shipped with the reference design are intended to demonstrate the correct operation of the stand-alone QDR SRAM interface. Altera recommends that you generate your own simulation environment that is better-suited to your SOPC design and EDA tools.

Resource Usage

The QDR controller reference design requires the resources shown in Table 4 for an APEX II device:

Table 4. Resource Usage

Logic Cells	PLLs	Global Clocks	I/O Pins
220	3	2	62 *(1)*

Note:

(1) 62 I/O pins, plus additional VREF pins, are needed to interface to the QDR SRAM. An additional 120 pins are necessary if you want to interface with the controller from outside the APEX II device, as implemented in the reference design.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
lit_req@altera.com