XAPP253 (v2.3) June 1, 2004

Synthesizable 400 Mb/s DDR SDRAM Controller

Author: Lakshmi Gopalakrishnan

Summary

This application note describes how to use a Virtex-II™ device to interface to a Double Data Rate (DDR) SDRAM device. The reference design targets a DDR SDRAM device at a clock rate of 200 MHz with data transfers at 400 Mb/s.

Introduction

This application note describes a DDR SDRAM controller design implemented in a Virtex-II device. The first section briefly reviews DDR SDRAM functionality while the following sections describe implementation and timing analysis details.

DDR SDRAM Overview

DDR SDRAM consists of high-speed CMOS, dynamic random-access memory (RAM). It is internally configured as a quad-bank DRAM. All inputs are compatible with the JEDEC SSTL_2 standard and all outputs are SSTL_II Class II compatible.

DDR SDRAM uses a double-data rate architecture to achieve high-speed operation. It uses data transfers on both edges of each clock cycle, effectively doubling the data throughput of the memory device. DDR SDRAM operates with a differential clock: CLK and $\overline{\text{CLK}}$. Commands (address and control signals) are registered at every CLK positive edge.

A bidirectional data strobe (DQS) is transmitted along with data, for use in data capture. It is transmitted by the controller during Write cycles and by the memory during Read cycles. DQS is edge-aligned with data during Read cycles and center-aligned with data during Write cycles.

Read and Write accesses to the DDR SDRAM are burst oriented; accesses start at a selected location and continue for a programmed number of locations in a programmed sequence. Accesses begin with the registration of an Active command, which is then followed by a Read or Write command. The address bits registered coincident with the Read or Write command are used to select the bank and the starting column location for the burst access.

DDR SDRAM provides locations for 2, 4, 8, or full-page programmable Read or Write burst lengths. An auto-precharge function can be enabled to provide a self-timed precharge that is initiated at the end of the burst access. As with standard SDR SDRAMs, the pipelined, multibank architecture of DDR SDRAMs allows for concurrent operation, thereby providing high effective bandwidth by hiding row precharge and activation time.

Mode Register

The mode register is used to define the specific DDR SDRAM mode of operation. This definition includes the selection of a burst length, a burst type, a CAS latency, and an operating mode. The mode register is programmed via the MODE REGISTER SET command and retains the stored information until it is programmed again or the device loses power. The mode register must be loaded when all banks are idle and no bursts are in progress.

Burst Length

Mode register bits A0 – A2 specify the burst length. The burst length determines the maximum number of column locations to be accessed for a given Read or Write command. Burst lengths of 2, 4, or 8 locations are available for both sequential and interleaved burst types. Full-page burst is supported in sequential mode only.

Burst Type

Accesses within a given burst can be programmed to be either sequential or interleaved; this is referred to as the burst type and is selected via bit M3.

Read Latency

Read latency is the delay, in clock cycles, between the registration of a Read command and the availability of the first bit of output data. The latency can be set to 2, 3, or 4 clocks. A Read latency of 3 clocks, required by the 200 MHz memory device, is used in this reference design.

Operating Mode

The normal operating mode is selected by issuing a MODE REGISTER SET command with bits A7 – A11 each set to zero, and bits A0 – A6 set to the desired values. A DLL reset is initiated by issuing a MODE REGISTER SET command with bits A7 and A9 – A11 each set to zero, but A8 set to one and bits A0 – A6 set to the desired values.

Extended Mode Register

The extended mode register controls functions beyond those controlled by the mode register; these additional functions are DLL enable/disable. The DLL must be enabled for normal operation. DLL enable is required during power-up initialization and upon returning to normal operation after having disabled the DLL for the purpose of debug or evaluation. The extended mode register is programmed via the LOAD MODE REGISTER command to the mode register (with BA0 = 1 and BA1 = 0). The extended mode register must be loaded when all banks are idle and no bursts are in progress. The DDR SDRAM data sheet shows the configuration of the extended mode register.

Commands

This section describes commands used with the DDR SDRAM controller.

DESELECT

The DESELECT function prevents new commands from being executed by the DDR SDRAM. The DDR SDRAM is effectively deselected. Operations already in progress are not affected.

NO OPERATION (NOP)

The NO OPERATION (NOP) command is used to instruct the selected DDR SDRAM to perform a NOP. This prevents unwanted commands from being registered during idle or wait states. Operations already in progress are not affected.

LOAD MODE REGISTER

The mode registers are loaded via inputs A0 – A11. The LOAD MODE REGISTER command can be issued only when all banks are idle, and a subsequent executable command cannot be issued until t_{MRD} is met.

Active Command

The Active command is used to open or activate a row in a particular bank for subsequent access. The value on inputs BA0 and BA1 selects the bank, and the address provided on inputs A0 – A7 selects the starting column location. The value on input A8 determines whether or not auto precharge is used. If auto precharge is selected, the row being accessed is precharged at the end of the Read burst, and if auto precharge is not selected, the row remains open for subsequent accesses. The Active command is followed by a Read or a Write command.

The Read command is used to initiate a burst Read access to an active row and the Write command is used to initiate a burst Write access to an active row. The value on inputs BA0 and BA1 selects the bank, and the address provided on inputs A0 – A7 selects the starting column location.

PRECHARGE

The PRECHARGE command is used to deactivate the open row in a particular bank or the open row in all banks. The bank(s) are available for a subsequent access at a specified time (t_{RP}) after the PRECHARGE command is issued. Input A8 determines whether one or all banks are to be precharged, and in the case where only one bank is to be precharged, inputs BA0 and BA1 select the bank.

BURST TERMINATE

The BURST TERMINATE command is used to truncate Read bursts. The most recently registered Read command prior to the BURST TERMINATE command is truncated. The DDR SDRAM data sheet shows how these commands are generated using the ddr_rasb, ddr_casb and ddr_web signals by the controller.

DDR SDRAM Controller Implementation Details

Design Features

DDR SDRAM controller design features include:

- FIFO backend user interface
- Programmable burst lengths of 2, 4, 8, or full-page (available only in certain DRAMs)
- Programmable CAS latency of 3
- Burst length applies to both Read and Write cycles
- Interfaces with the DDR SDRAM at 200 MHz, DDR (400 Mb/s)
- Uses DQS to receive data from the memory

User Interface

The backend interface of the DDR SDRAM controller is a FIFO implementation. Four synchronous FIFOs that operate at 200 MHz are used: the Write Address FIFO, Write Data FIFO, Read Address FIFO, and the Read Data FIFO. A configuration register along with a command register is provided by the user interface.

Write Data FIFO

The Write Data FIFO serves as a buffer for the backend interface to store data that needs to be written into the DDR SDRAM memory. The Write Data FIFO is 72 bits wide and 32 words deep. This includes the rising and the falling edge data as well as the Data mask inputs.

Configuration of the 72 bits is as follows:
{rising edge data, falling edge data, rising edge data mask, falling edge data mask}.

For a burst length of 2, each location in the Write Data FIFO constitutes data for each burst. For a burst length of 4, two locations in the Write Data FIFO constitute the data required. Similarly, four locations in the Write Data FIFO constitute data for one burst of eight words.

Write Address FIFO

The Write Address FIFO serves as the buffer for the backend interface to store addresses corresponding to the write data. The Write Address FIFO is 22 bits wide. The Write Address constitutes 12 bits which form the row address, 8 bits which form the column address, and 2 bits for the bank address.

Configuration of the 22 bits is as follows:
{row address, column address, bank address}.

Read Address FIFO

The Read Address FIFO serves as the buffer for the backend interface to store address locations in the memory from where it wants to read data from. The Read Address FIFO configuration is similar to the Write Address FIFO configuration and is 22 bits wide and 16 words deep.

Configuration of the Read Address data is as follows:
{row address, column address, bank address}.

Read Data FIFO

The Read Data FIFO serves as the buffer for the controller to write data it has read from the memory. This FIFO is 64 bits wide and constitutes only the Rising and the Falling edge data.

Configuration for the Read data FIFO is as follows:
{Rising edge data, Falling edge data}.

The controller provides a data valid signal which can be used as the read enable to read data from the FIFO by the backend.

Generating the FIFOs

The FIFOs used in the reference design are synchronous FIFOs generated using Coregen. These FIFOs can be accessed from the Xilinx Foundation directory:
Xilinx ISE 6 -> Accessories -> CORE Generator -> Memory & Storage Elements -> FIFOs ->Synchronous FIFO.

The required width and depth need to be specified. The netlist that is generated through Coregen needs to be saved in the same directory as that of the design netlist(.edf)

Figure 1 provides a top-level block diagram of the DDR SDRAM controller.

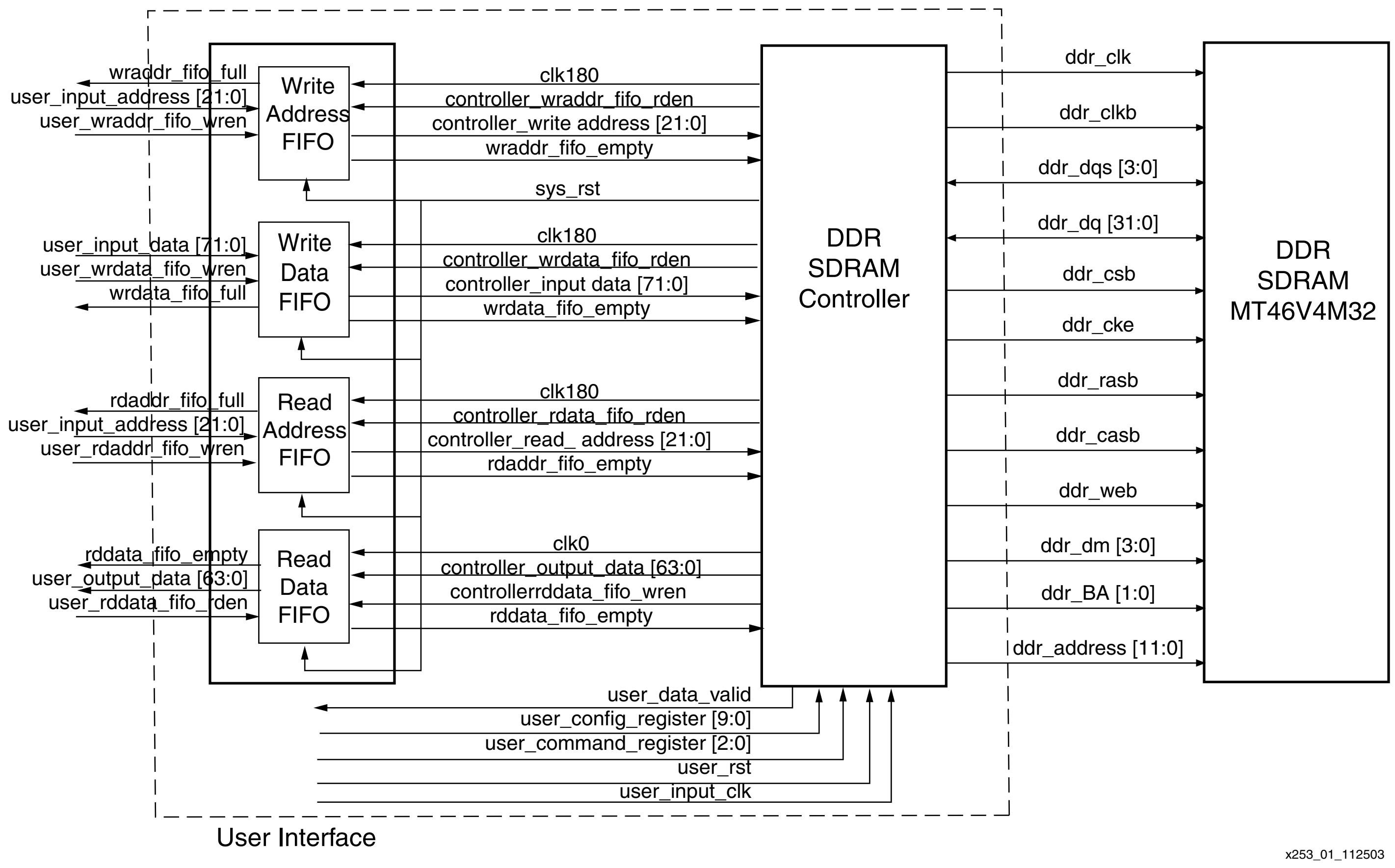

Figure 1: **Top-Level Block Diagram of DDR SDRAM Controller**

Table 1 provides pin descriptions for the DDR SDRAM controller.

Table 1: **ddr_ctrl Pin Descriptions**

	Pin Name	Pin Direction	Width	Description
Interface to User Logic	user_input_clk	In	1	Input clock
	user_rst	In	1	Input reset signal
	user_config_register	In	10	Configuration commands for the controller
	user_command_register	In	3	Commands for the controller
	user_input_address	In	22	Write or Read Address (Row address, column address, bank address)
	user_input_data	In	72	Write data and write mask
	user_wraddr_fifo_wren	In	1	Write enable for the write_address_fifo
	user_wrdata_fifo_wren	In	1	Write enable for the write_data_fifo
	user_rdaddr_fifo_wren	In	1	Write enable for the read_address_fifo
	user_rddata_fifo_rden	In	1	Read enable for the read_data_fifo
	user_output_data	Out	64	Output Read data
	user_data_valid	Out	1	Signal to indicate valid output data
	wraddr_fifo_full	Out	1	Full flag for write_address_fifo
	wrdata_fifo_full	Out	1	Full flag for write_data_fifo
	rdaddr_fifo_full	Out	1	Full flag for read_address_fifo
	rddata_fifo_empty	Out	1	Empty flag for read_data_fifo
Interface to DDR SDRAM	ddr_address	Out	12	Address
	ddr_dq	In/Out	32	Data
	ddr_dqs	In/Out	4	Data strobe
	ddr_rasb	Out	1	Command
	ddr_casb	Out	1	Command
	ddr_web	Out	1	Command
	ddr_ba	Out	2	Bank address
	ddr_clk	Out	1	Clock
	ddr_clkb	Out	1	Clock (inverted)
	ddr_csb	Out	1	Chip select
	ddr_cke	Out	1	Clock enable
	ddr_dm	Out	4	Data mask

Configuration Register

The reference design uses a CAS latency of 3 which is the supported value for 200 MHz operation. The design includes test benches that provide data for burst 2, 4, and 8 cases.

Table 2 includes the 10-bit user configuration register in the proper order = {EMR, BMR operation, BMR/EMR, CAS Latency, Burst type, Burst length}.

Table 2: **User Configuration Register Definition**

Parameter Name	Width	Description
EMR	1	Extended Mode Register 0 – Enable DLL 1 – Disable DLL
BMR Operation	1	Base Mode Register 0 – Normal BMR operation 1 – Normal operation/Reset DLL
BMR/EMR	1	0 – MR 1 – EMR
CAS Latency	3	010 – 2 011 – 3 100 – 4 Others – Reserved
Burst Type	1	0 – Sequential 1 – Interleaved
Burst Length	3	001 – 2 010 – 4 011 – 8 111 – Full Others – Reserved

Command Register

Table 3 shows the command register definitions.

Table 3: **Command Register**

Command	Description
101	Load Mode
111	Burst Terminate
Others	NOP

The controller assumes that the burst lengths used during the Initialization sequence do not change during the controller operation. Hence, if the user needs to change the burst lengths or CAS latencies, a load mode command needs to be issued.

BURST TERMINATE command is to terminate a READ burst. READ and WRITE bursts are done based on the states of the FIFOs and, hence, the commands are not issued through the command register.

Memory Initialization

The memory must be powered up and initialized in a predefined manner. The initialization sequence is handled by the controller, as follows:

1. After all power supply and reference voltages are stable and the clock is stable, the DDR SDRAM requires a 200 μs delay prior to applying an executable command.
2. Once the 200 μs delay has passed, the initialization sequence begins:
    a. A DESELECT or NOP command is applied and CKE is set High.
    b. A PRECHARGE ALL command is applied.

c. A LOAD MODE REGISTER command is issued for the extended mode register (BA1 = 0 and BA0 = 1) to enable the DLL.

d. Another LOAD MODE REGISTER command is issued to the mode register (BA0 and BA1, both 0) to reset the DLL and to program the operating parameters.

e. A PRECHARGE ALL command is applied, placing the device in an all-banks-idle state.

f. Once in the IDLE state, two AUTO REFRESH cycles are performed.

Additionally, a LOAD MODE REGISTER command is recommended for the mode register with the deactivated reset DLL bit. This command can be given at any time after issuing the LOAD MODE REGISTER command to reset the DLL.

Two hundred clock cycles are required between the DLL reset and any Read command. This needs to be handled by the user interface before sending any Read command to the controller. After meeting these requirements, the DDR SDRAM is ready for normal operation.

Figure 2 illustrates the DDR SDRAM controller state machine.

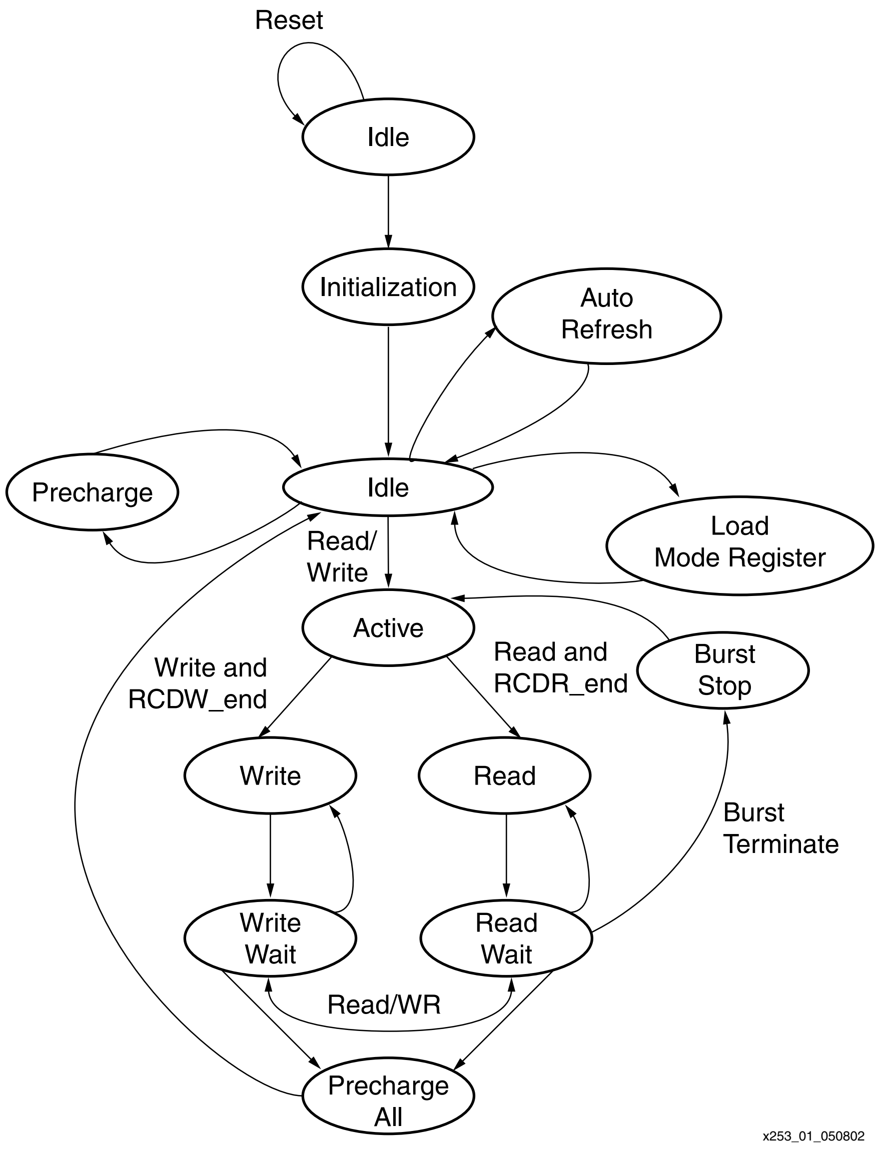

Figure 2: **DDR SDRAM Controller State Machine**

Memory Accesses

To open a row in a specified bank, an ACTIVE command needs to be given after the LOAD MODE REGISTER command. The ACTIVE command selects both the bank and the row to be activated. After a row is opened with an ACTIVE command, a Read or Write command can be issued to that row, subject to the t_{RCD} specification. A subsequent ACTIVE command to a different row in the same bank can be issued only after the previous active row has been closed (precharged).

AUTO REFRESH Counter

The memory requires AUTO REFRESH cycles at an average interval of 15.6 µs. A clock divided from the system clock of 200 MHz is used in a counter to generate the AUTO REFRESH cycles. The AUTO REFRESH commands are generated when the controller state machine is in the IDLE state.

The AUTO REFRESH controller circuitry generates an AUTO REFRESH flag when it is time to generate an AUTO REFRESH command after 15.6 µs. When the AUTO REFRESH flag is High, any pending Read/Write state is not interrupted. After the state machine enters the IDLE state, the AUTO REFRESH command is executed and the AUTO REFRESH flag is set Low.

Controller Write Operation

1. The controller detects a write operation by detecting the write address FIFO being non-empty
2. The controller compares the input row and bank address to the active row and bank address to make sure there is no address conflict. If there is a conflict, the controller closes the bank by issuing a PRECHARGE command followed by an ACTIVE and FIIRST_WRITE commands.
3. The controller then asserts the read enable signal to the Write Data FIFO. Data from the FIFO is then registered on the WCLK domain and in the DDR IOB registers and then written into the memory. The corresponding write commands are also issued to the memory by the controller.
4. Steps 1 through 3 are continued until the write FIFOs are empty.
5. The controller can detect write requests in the IDLE, Write burst, or READ_WAIT states.
6. After the write burst is completed, the controller stays in the WRITE_WAIT state for WAIT_CYCLES, which is the number of wait cycles after the last write command has been issued. This can be modified to a desirable number of cycles. Once the WAIT_CYCLES number of cycles has passed with no new incoming write requests, the controller closes the bank and moves to the IDLE state. Any pending AUTO REFRESH requests detected by the AUTO_REF command will then be issued.

See Figure 3, Figure 4, and Figure 5.

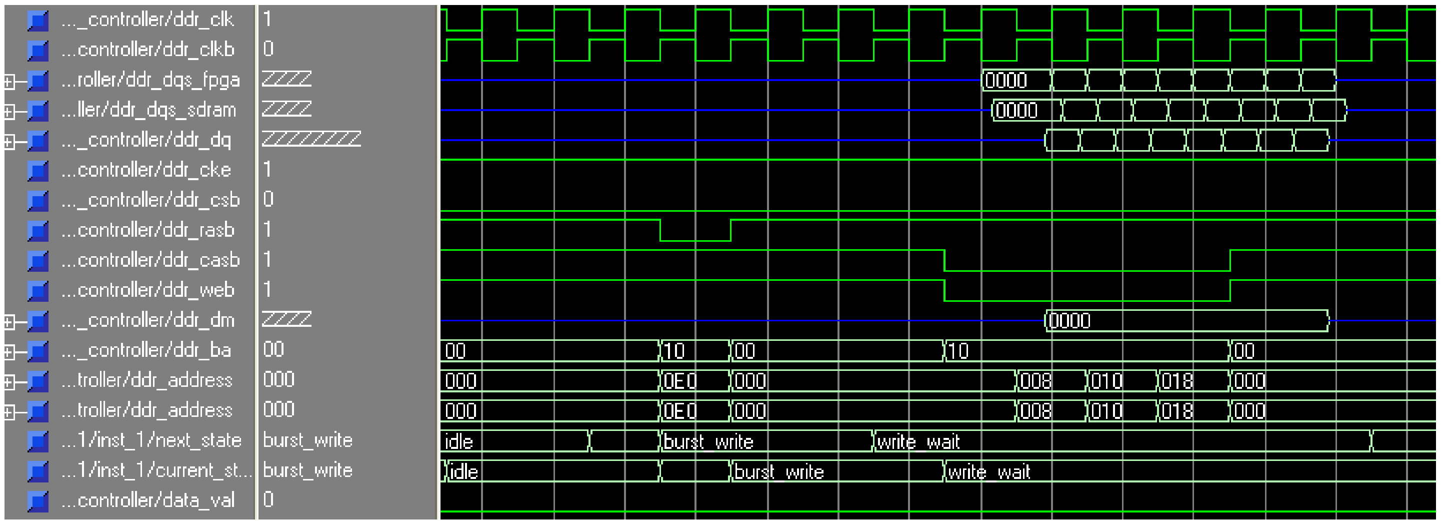
..._controller/ddr_clk 1
...controller/ddr_clkb 0
...roller/ddr_dqs_fpga ZZZZ
...ller/ddr_dqs_sdram ZZZZ
..._controller/ddr_dq ZZZZZZZZ
...controller/ddr_cke 1
..._controller/ddr_csb 0
...controller/ddr_rasb 1
...controller/ddr_casb 1
...controller/ddr_web 1
..._controller/ddr_dm ZZZZ
..._controller/ddr_ba 00
...troller/ddr_address 000
...troller/ddr_address 000
...1/inst_1/next_state burst_write
...1/inst_1/current_st... burst_write
...controller/data_val 0
0000
0000
0000
00 10 00 10 00
000 0E0 000 008 010 018 000
000 0E0 000 008 010 018 000
idle burst_write write_wait
idle burst_write write_wait

Figure 3: **Consecutive Burst Writes on a Burst Length of 2**

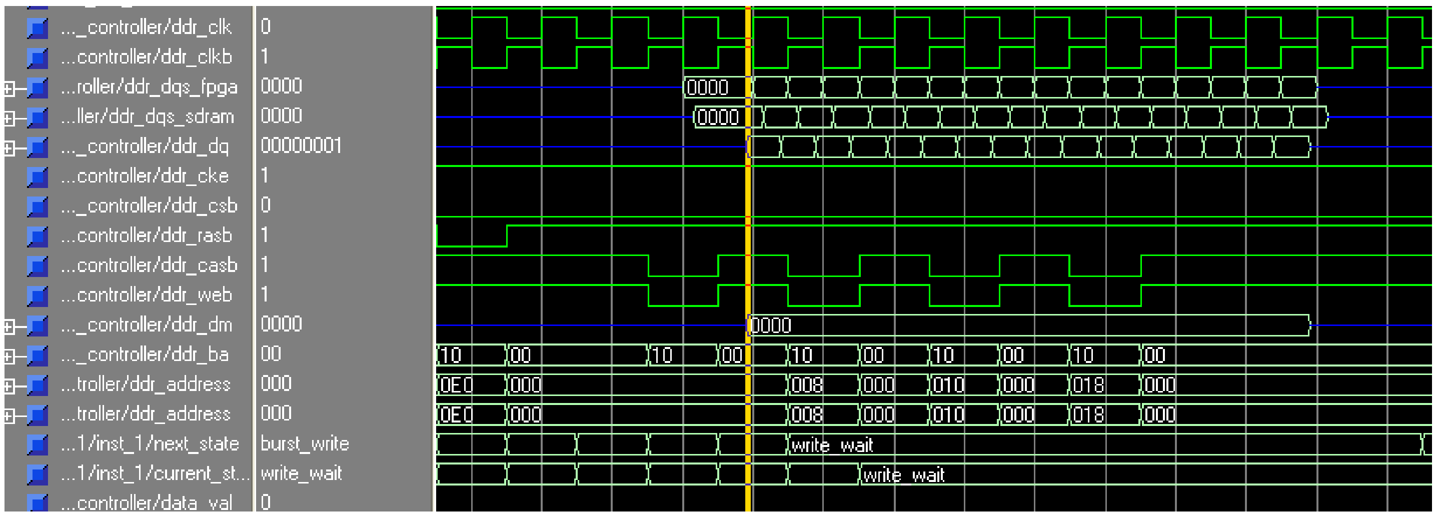
..._controller/ddr_clk 0
...controller/ddr_clkb 1
...roller/ddr_dqs_fpga 0000
...ller/ddr_dqs_sdram 0000
..._controller/ddr_dq 00000001
...controller/ddr_cke 1
..._controller/ddr_csb 0
...controller/ddr_rasb 1
...controller/ddr_casb 1
...controller/ddr_web 1
..._controller/ddr_dm 0000
..._controller/ddr_ba 00
...troller/ddr_address 000
...troller/ddr_address 000
...1/inst_1/next_state burst_write
...1/inst_1/current_st... write_wait
...controller/data_val 0
0000
0000
0000
10 00 10 00 10 00 10 00 10 00 10 00
0E0 000 008 000 010 000 018 000
0E0 000 008 000 010 000 018 000
write_wait
write_wait

Figure 4: **Consecutive Burst Writes on a Burst Length of 4**

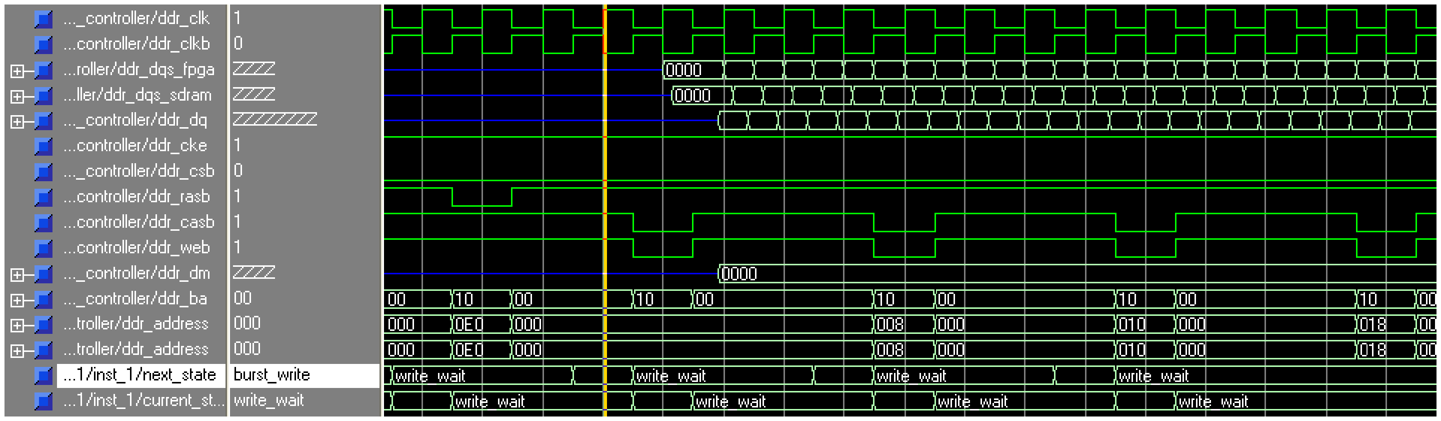
..._controller/ddr_clk 1
...controller/ddr_clkb 0
...roller/ddr_dqs_fpga ZZZZ
...ller/ddr_dqs_sdram ZZZZ
..._controller/ddr_dq ZZZZZZZZ
...controller/ddr_cke 1
..._controller/ddr_csb 0
...controller/ddr_rasb 1
...controller/ddr_casb 1
...controller/ddr_web 1
..._controller/ddr_dm ZZZZ
..._controller/ddr_ba 00
...troller/ddr_address 000
...troller/ddr_address 000
...1/inst_1/next_state burst_write
...1/inst_1/current_st... write_wait
0000
0000
0000
00 10 00 10 00 10 00 10 00 10 00
000 0E0 000 008 000 010 000 018 00
000 0E0 000 008 000 010 000 018 00
write_wait write_wait write_wait write_wait
write_wait write_wait write_wait write_wait

Figure 5: **Consecutive Burst Writes on a Burst Length of 8**

Controller READ Operation

1. The controller detects an incoming read request by detecting the read address FIFO being non-empty.
2. The controller compares the input row and bank address to the active row and bank address to make sure there is no address conflict. If there is a conflict, the controller closes the bank by issuing a PRECHARGE command followed by an ACTIVE and FIRST_READ commands.
3. The controller can detect read requests from the IDLE or READ_WAIT state or in the WRITE_WAIT states.
4. The corresponding READ commands are issued to the memory by the controller.
5. Steps 1 through 4 are continued until the read address FIFO is empty.
6. After the read burst is completed, the controller stays in the READ_WAIT state for WAIT_CYCLES, which is the number of wait cycles after the last read command has been issued. This can be modified to a desirable number of cycles. After the WAIT_CYCLES number of cycles has passed with no new incoming read requests, the controller closes the bank and moves to the IDLE state. Any pending AUTO REFRESH requests detected by the AUTO_REF command will then be issued.

See Figure 6, Figure 7, and Figure 8.

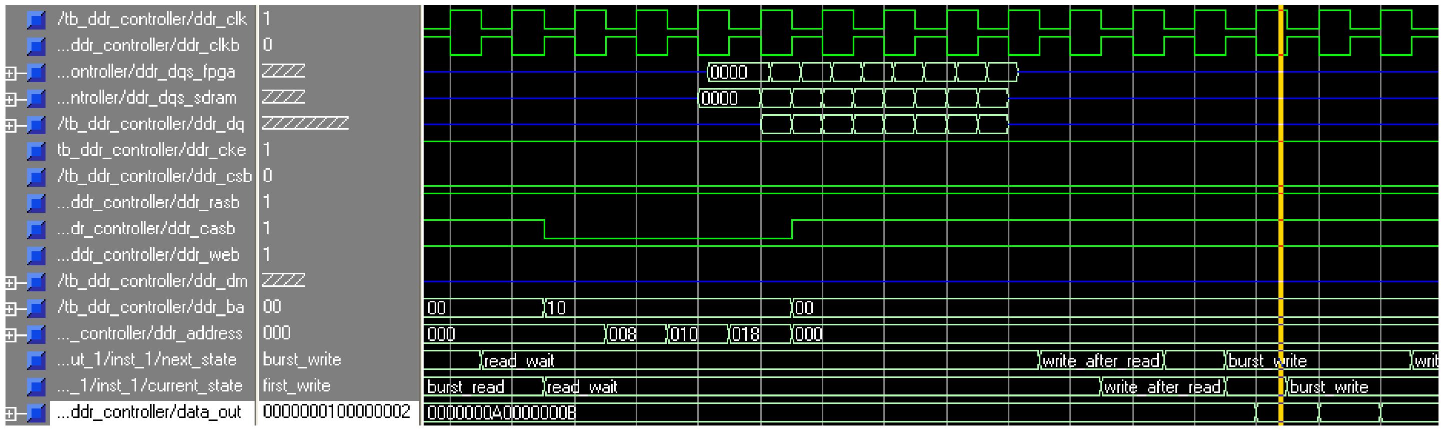
/tb_ddr_controller/ddr_clk 1
...ddr_controller/ddr_clkb 0
...ontroller/ddr_dqs_fpga ZZZZ
...ntroller/ddr_dqs_sdram ZZZZ
/tb_ddr_controller/ddr_dq ZZZZZZZZ
tb_ddr_controller/ddr_cke 1
/tb_ddr_controller/ddr_csb 0
...ddr_controller/ddr_rasb 1
...dr_controller/ddr_casb 1
...ddr_controller/ddr_web 1
/tb_ddr_controller/ddr_dm ZZZZ
/tb_ddr_controller/ddr_ba 00
..._controller/ddr_address 000
...ut_1/inst_1/next_state burst_write
..._1/inst_1/current_state first_write
...ddr_controller/data_out 0000000100000002
0000
00
10
008
010
018
read_wait
write_after_read
burst_write
burst_read
0000000A0000000B

Figure 6: **Reads of Burst Lengths 2**

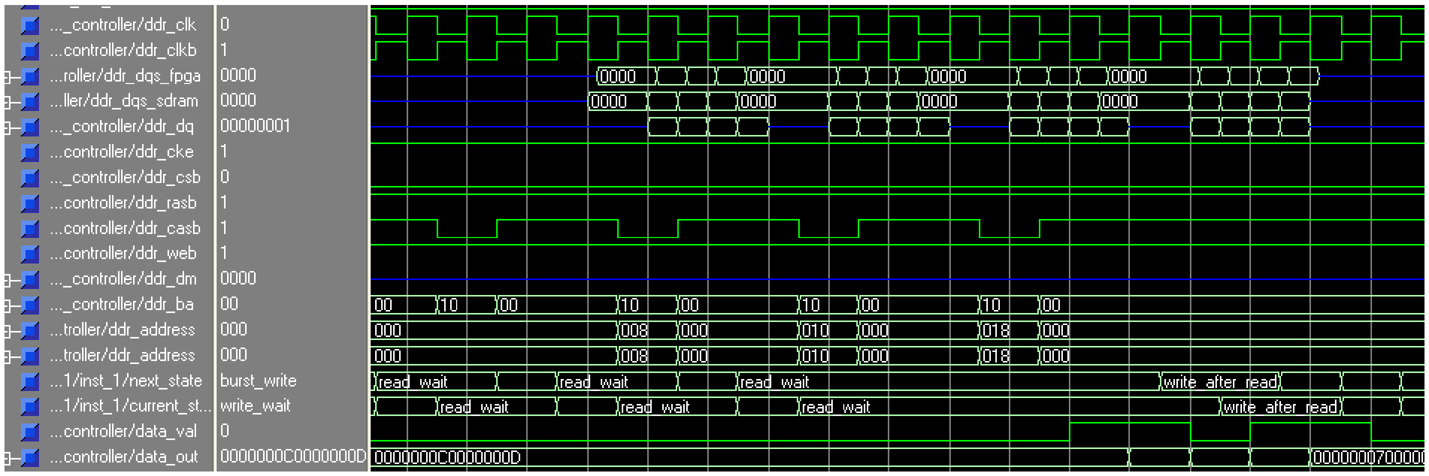

Figure 7: **Reads of Burst Lengths 4**

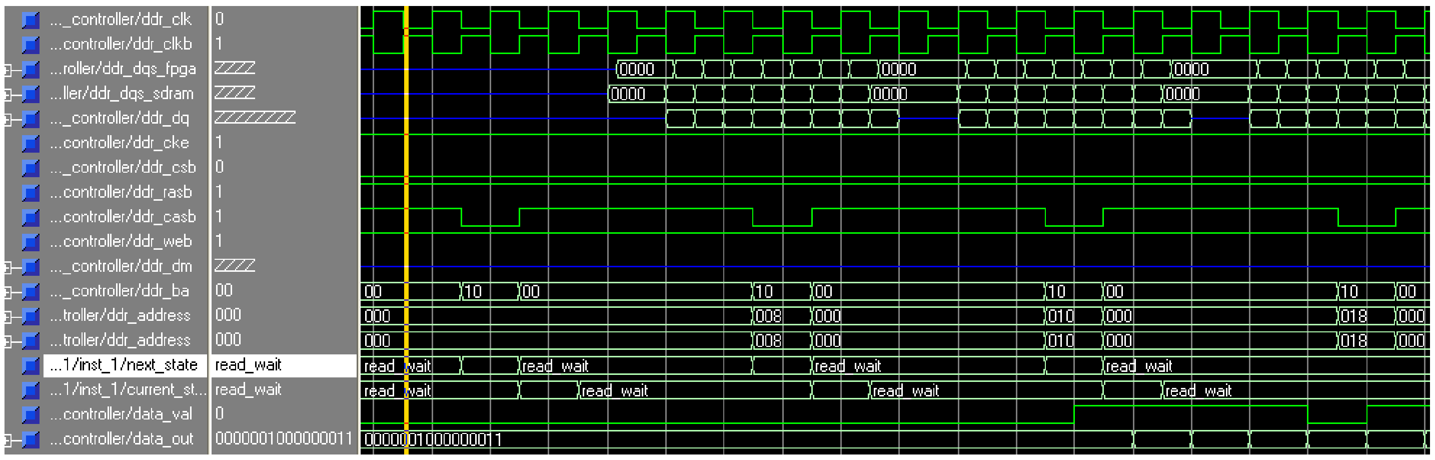

Figure 8: **Reads of Burst Lengths 8**

Priority Between Read and Write Requests

The controller implementation provided in this application note does not support providing priority between read and write requests. It is optimized for an implementation which provides continuous write bursts followed by read bursts or vice versa. Care has been taken to provide a generic state machine that is optimal for all the burst lengths. The read/write state machines can be modified for better performance for a known backend interface.

Clocking Methodology

This section describes the clocking methodology implemented in this design. The first DCM generates CLK0 and CLK90. CLK0 directly follows the user-supplied input clock. This DCM also supplies the CLKDV output, which is the input clock divided by 16 used for the AUTO REFRESH counter. Figure 9 is the DCM implementation.

The second DCM in the controller block (DCM2_RECAPTURE) generates a phase-shifted version of the user input clock. It is used to recapture data from the DQS clock domain during a memory Read. Data recaptured in the rclk domain is then transferred to the system clock domain. The phase-shift value is specific to the system and must be programmed accordingly.

When adequate DCM resources are available, a third DCM can be used for better timing margins.The third (optional) DCM is explained in further detail in I/O Timing Analysis. This DCM is used to generate WCLK, a phase shifted version of the system clock. WCLK is used to clock data at the DDR IOB registers during a Write.

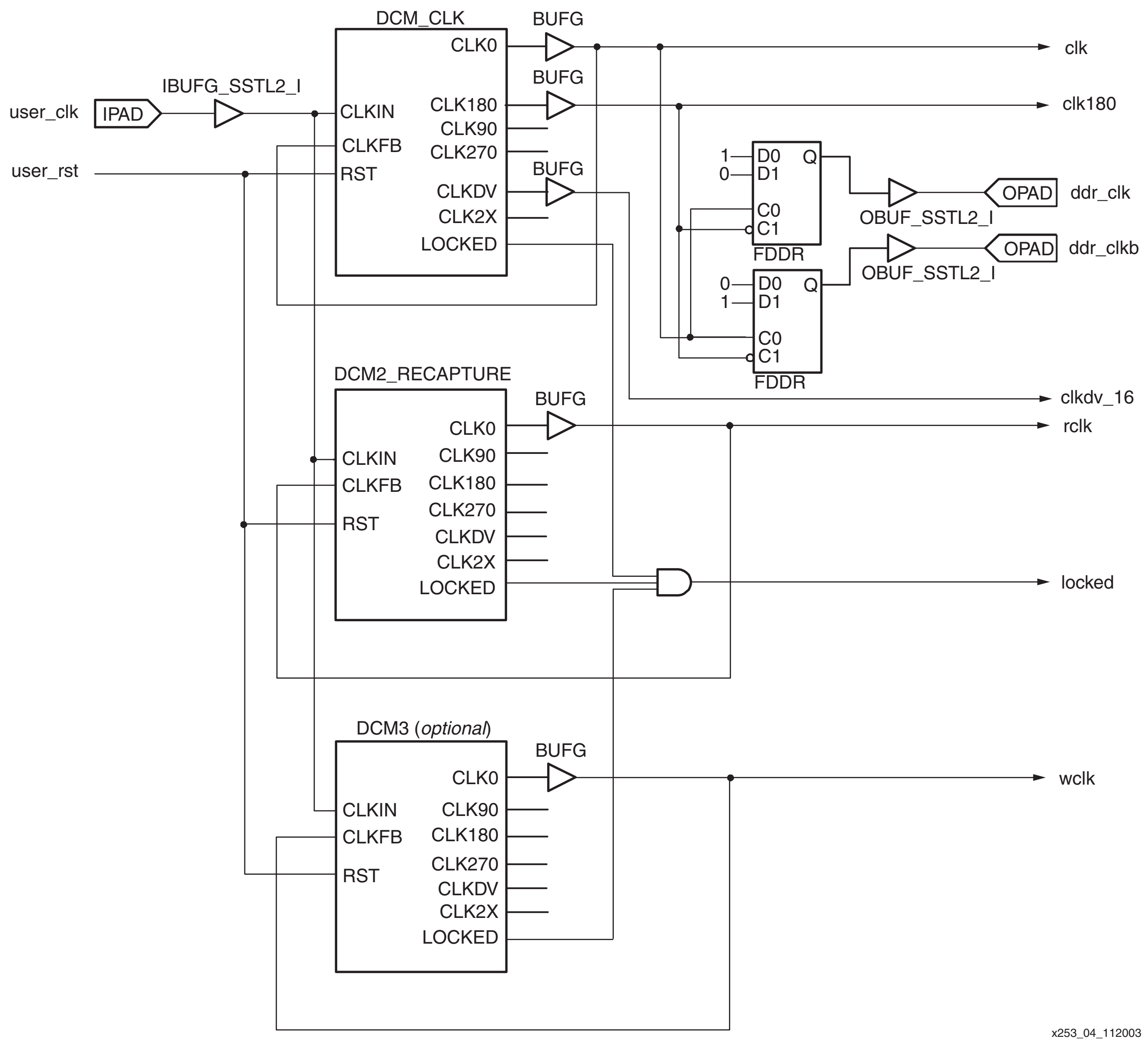

Figure 9: **DCM Implementation**

Data Path

Virtex-II devices have I/O blocks (IOBs) that can be used for DDR functions. To minimize clock-to-out delays, the reference design allows both inputs and outputs to the DDR SDRAM interface to be registered within the IOBs.

Figure 10 provides a representation of IOBs available in Virtex-II devices and shows how an IOB is used in case of a bidirectional data (DQ) signal, where the DDR IOB needs to handle both input and output signals.

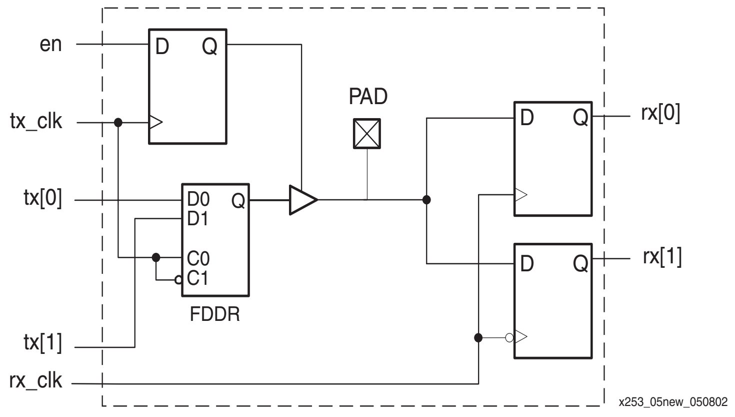

Figure 10: **DDR IOB Example Implementation**

Figure 11 illustrates the controller's Data path. For all input signals, an SRL label is used to indicate multiple pipeline stages. These pipeline stages are used to align the data and the DQS signal with the DDR SDRAM control signals. Figure 11 also clearly shows the portion of the Data path implemented in the IOBs.

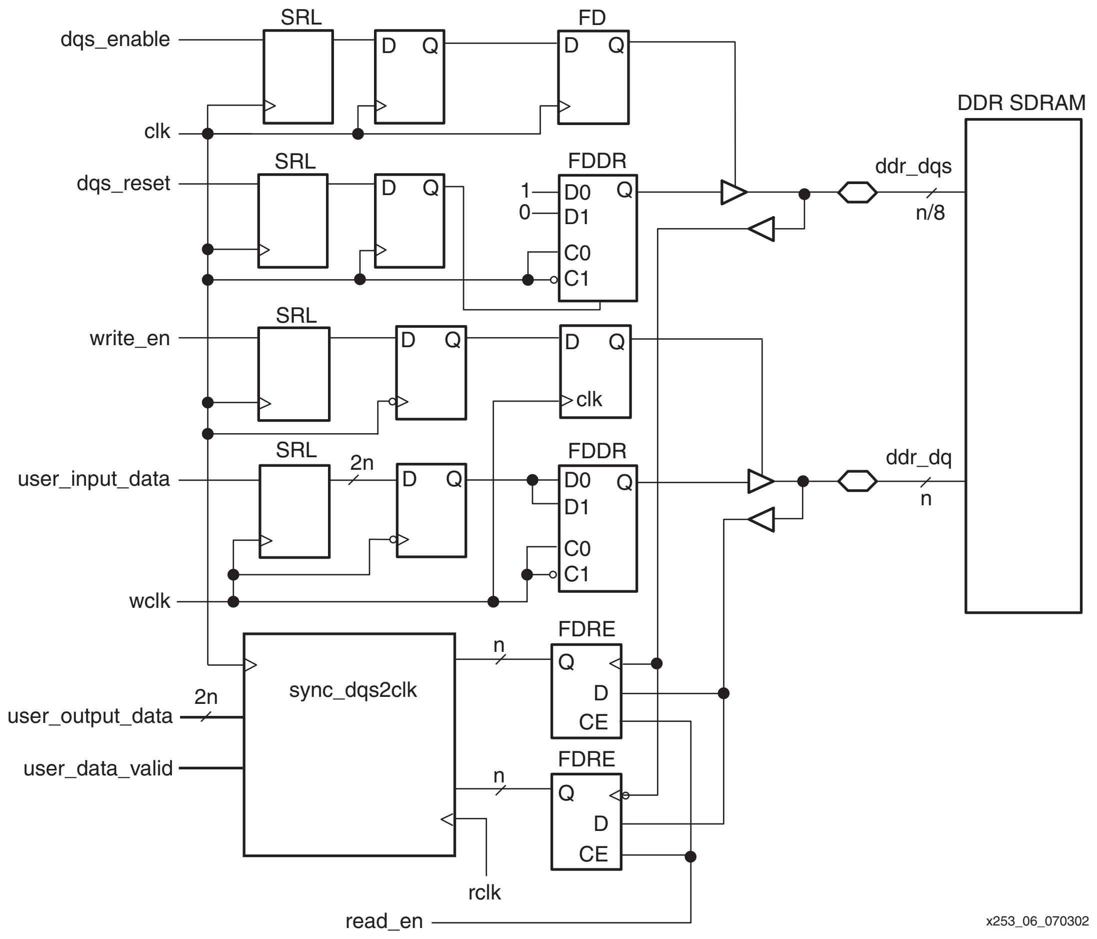

Figure 11: **DDR SDRAM Controller Data Path**

Write Data Path

During a memory Write, the controller must make sure the Data Strobe (DQS) is center aligned with the data at the pins of the DDR SDRAM device. The memory device requires the DQS and CLK to be approximately phase-aligned, although specification allows skew. In order to minimize this skew, both CLK and DQS are forwarded through the DDR IOB flip-flops.

The controller block generates the dqs_enable and the dqs_reset signal required at the IOB flip-flops for generating the DQS signal during a Write cycle. The dqs_enable signal controls the 3-state output while the dqs_reset holds the DQS flip flop in reset. The dqs_reset helps to meet the Write preamble timing required by the memory. The dqs_reset is released after a clock cycle and the DDR flip flops clocked by CLK generate the DQS to the memory.

The write_en signal generated by the controller block controls the 3-state output of the data path. The write timing analysis discusses both cases when data is generated at the DDR IOB registers by CLK (similar to DQS) and when data is generated by using WCLK to offer more margin. WCLK is a phase shifted version of CLK. The first case depends on the trace difference required between the DQS and DQ signals (from the Read timing analysis) to position the DQS inside the Data window at the memory. To improve available margin, the design looks at using an additional DCM to generate WCLK. WCLK is used to position DQS in the center of the data window at the memory.

Read Data Path

During Read cycles, the memory provides DQS edge aligned with the Data to the controller. This designs uses DQS to capture data provided by the memory. The DQS is distributed on local dedicated clocking resources to clock the Data in the DDR IOBs. Because DQS is strobing in nature, data captured on the DQS domain needs to be recaptured.

Data Recapture

In order to recapture data, a relationship between the DQS domain and the system clock domain must be found. The reference design uses a phase shifted version of the system clock (rclk) to recapture data from the DQS domain. This is explained in the Read Recapture Timing Analysis section.

Data in the DQS domain is written into an asynchronous FIFO built using LUT RAMs. The phase shifted clock, rclk is used to write the data into the FIFO. The data is then read using the system clock (CLK) to convert the data into the system clock domain. Since the recapture clock is asynchronous to the internal system clock, all transfers between clock domains are double registered to remove any setup, hold, or metastability issues.

Pinout Details for Using Local Clock Distribution

Virtex-II devices contain local clock distribution networks along the left and right edges of the device. These networks allow a signal to enter an IOB and connect to a low-skew routing resource that connects directly to a fixed number of IOBs.

As described in Figure 12, each I/O tile contains four IOBs that share a switch-matrix. IOB PAD3 is the top IOB in a I/O tile. In order for the DQS to access a local low-skew clock line, DQS must be placed on the top IOB, PAD3 in the I/O tile. If IOB PAD3 is not available in a tile, or is unbonded, then it cannot be used to place the DQS signal.

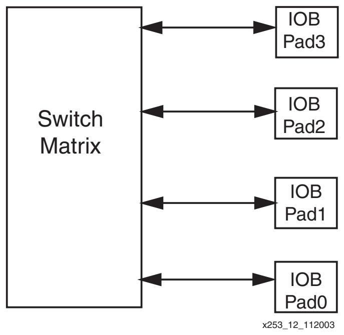

Figure 12: **Virtex-II Input/Output Tile**

Placing the DQS signal in PAD3 gives access to a local clock line which is a HEX line spanning five rows above the chosen I/O tile and six rows below the chosen I/O tile. The data (DQ) pads should be placed in the bonded IOBs available within these specific rows.

The Figure 13 shows a sample image from the FPGA editor showing the DQS pad driving the DQ pads located five rows above as well as six rows below.

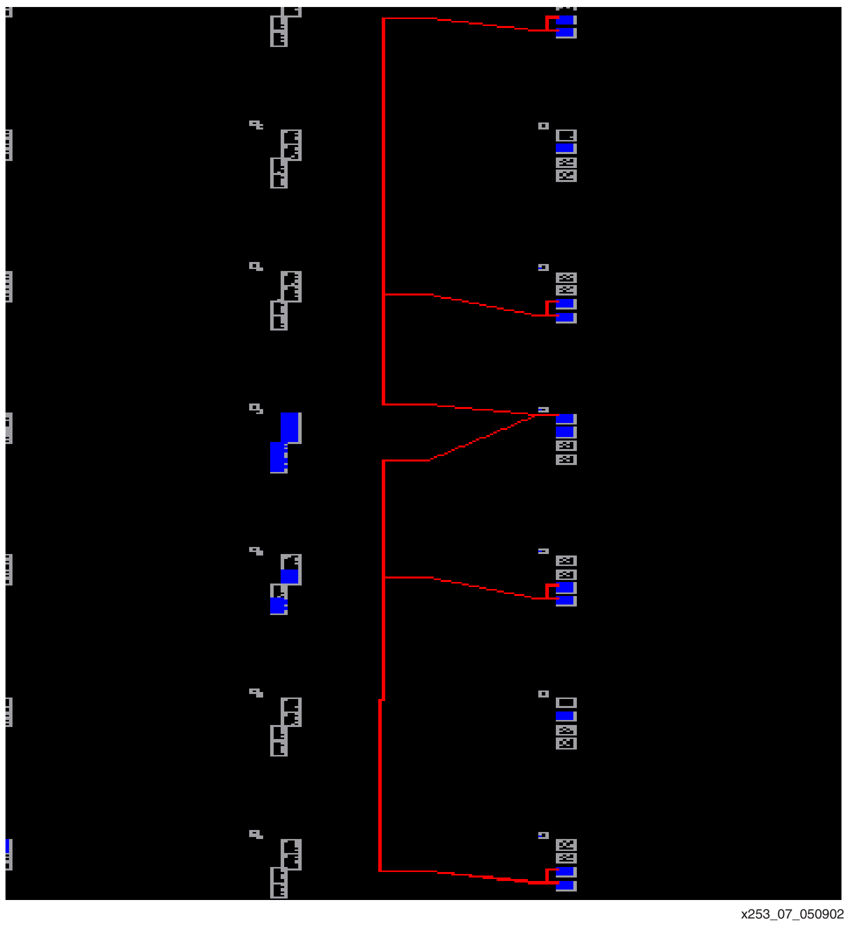

Figure 13: **Read Recapture**

I/O Timing Analysis

This section includes a detailed timing analysis of the reference design. The analysis uses a XC2V3000-5 device and a -5 speed grade Micron DDR SDRAM device. The parameters used for the analysis are shown in Table 4 and Table 5. The Virtex-II values are from the source synchronous section of the Virtex-II data sheet.

Table 4: **Timing Parameters for an XC2V3000-5 FF1152**

Description	Symbol	Min	Max	Units
Input setup time, no delay (SSTL2)	T_{IOPICK}	1.34		ns
Input hold time, no delay (SSTL2)	T_{IOICKP}	–0.81		ns
Global clock and off with DCM	$T_{ICKOFDCM}$		2.50	ns
Package Skew	$T_{PKGSKEW}$		115	ps
Clock tree skew	$T_{CLKSKEW}$		100	ps
Duty cycle precision (DLL outputs)	CLKOUT_DUTY_CYCLE_DLL		±150	ps
Clock synthesis period jitter	CLKOUT_PER_JITT_0		±100	ps
	CLKOUT_PER_JITT_90		±150	ps

Table 5: **Timing Parameters for a DDR SDRAM device (Micron MT46V4M32-5)**

Description	Symbol	Min	Max	Units
Clock cycle time	t_{CK}	5	10	ns
CK high-level width	t_{CH}	0.45	0.55	t_{CK}
CK low-level width	t_{CL}	0.45	0.55	t_{CK}
Data input setup time from DQS	t_{DS}	0.5		ns
Data input hold time from DQS	t_{DH}	0.5		ns
Data output skew from DQS	t_{DQSQ}		0.5	ns
Access window of DQS from CK/$\overline{CK}$	t_{DQSCK}	–0.75	+0.75	
Half clock period	t_{HP}	t_{CH},t_{CL}		ns
DQ-DQS hold	t_{QH}	t_{HP} – 0.45 ns		ns
Data valid window		$t_{QH} - t_{DQSQ}$		ns

Read Timing Analysis

Since during a Read cycle the DQS and the DQ are edge-aligned, additional delay needs to be added on DQS to meet setup and hold timing requirements at the IOBs. This delay is achieved using both trace delay and route delay on the DQS through the HEX lines. The Read timing analysis derives the difference in trace delay between DQS and DQ.

Figure 14 shows the timing relationships of DQ to DQS at the pins of the DDR SDRAM device, at the pins of the FPGA, and at the IOB flip-flops. The trace delay of the DQ lines is referenced as t_{DQ} and the trace delay of the DQS lines is referenced as t_{DQS}.

Once DQS arrives at the pins of the FPGA, it enters an IOB and is routed to eight data (DQ) IOBs. Therefore, the DQS internal FPGA delay is comprised of the delay through an SSTL2 pad plus routing delay to the DQ loads. This is represented in Figure 14 as $t_{DQS\_INT\_DELAY}$.

The internal delay from the DQS IOB to the closest DQ IOB is $t_{DQS\_INT\_DELAY\ (MIN)}$. The delay to the farthest DQ IOB is $t_{DQS\_INT\_DELAY\ (MAX)}$. Table 6 represents sample delays on the DQS internal route. The delay between the minimum and maximum route delay on DQS is called $t_{DQS\_INT\_SKEW}$. Because the data (DQ) signals are latched in the IOB, there is no route delay. Any internal data delay through an SSTL2 pad is taken care of in the setup/hold requirements at the DDR IOBs (t_{IOPICK}/t_{IOICKP}).

The worst case setup occurs when using the maximum data (DQ) delay and minimum clock (DQS) delay. Similarly, the worst case hold occurs when using the maximum clock (DQS) delay and minimum data (DQ) delay. As shown in Figure 14, the difference between t_{SU} and T_{IOPICK} is the slack on the setup.

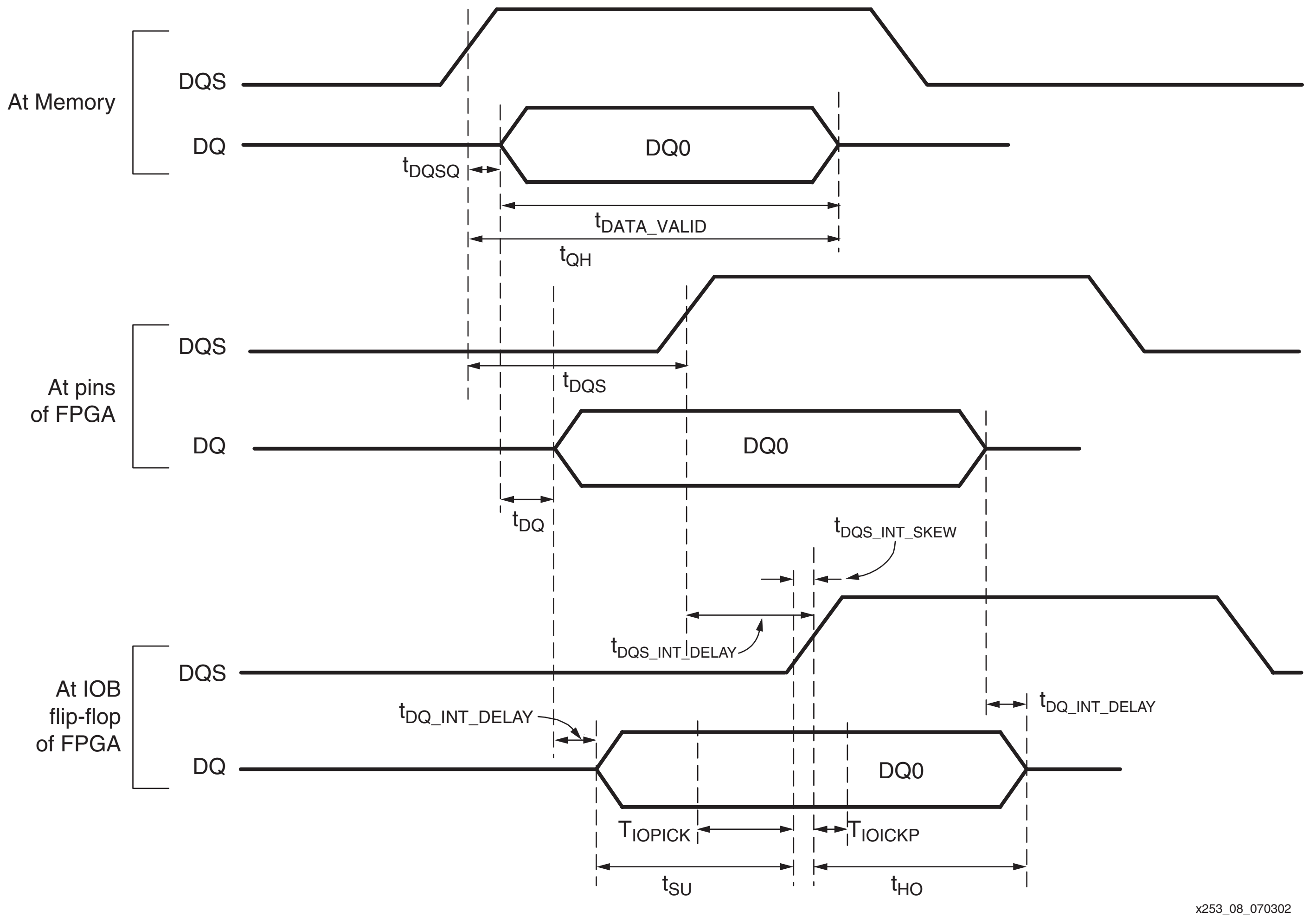

Figure 14: **DQS and DQ Signals**

Table 6: **DQ and DQS Internal Route Delay**

Description	Symbol	Min	Max	Units
Input Package Delay	$t_{DQ\_INT\_DELAY}$	0.00	0.00	ns
Input Clock Delay	t_{IOPI_SSTL2}		1.18	ns
DQS Internal Route Delay	$t_{DQS\_ROUTE\_DELAY}$[1]	363	376	ps
$t_{IOPI\_SSTL2} + t_{DQS\_ROUTE\_DELAY}$	$t_{DQS\_INT\_DELAY}$[1]	1.543	1.556	ns

Notes:

1. These values are based on implementation results and hence the actual trace numbers need to be substituted.

Similarly the difference between t_{HO} and T_{IOICKP} is the slack on the Hold.

$t_{SU} = (t_{DQS} + t_{DQSINTDELAYMIN}) - (t_{DQSQ} + t_{DQ} + t_{DQINTDELAY}) > T_{IOPICK}$

$T_{HO} = t_{DQ} + t_{DATAVALID} + t_{DQSQ} + t_{DQINT\_DELAYMIN}) - (t_{DQSQ} + t_{DQSINTDELAYMAX})$ where

$t_{DATAVALID} = t_{QH} - t_{DQSQ}$

The amount of delay needed between DQS and DQ traces is found by solving the two equations for $t_{DQS} - t_{DQ}$ where t_{DQS} is the trace delay on DQS and t_{DQ} is the trace delay on DQ.

$T_{IOPICK} - t_{DQS\_INT\_DELAYMIN} + (t_{DQSQ} + t_{DQINTDELAY})$
$< t_{DQS} - t_{DQ} < - T_{IOICKP} + t_{QH} + t_{DQINTDELAY} - t_{DQSINTDELAYMAX}$

Using the timing values in the Virtex-II data sheet:

$1.38 - (1.543) + (0.5 + 0.0) < t_{DQS} - t_{DQ}$

$< -(-0.81) + (0.45 \times 5 - 0.5) + 0.0 - 1.556$

$0.337 < t_{DQS} - t_{DQ} < 1.004$ ns $\rightarrow$ Equation 1

The minimum analysis uses the fact that minimum numbers are 40 percent of the maximum numbers while using local clocks. Equation 2:

$(T_{IOPICK} \times PF) - (t_{DQSINTDELAYMIN} \times PF) + t_{DQSQ} + (t_{DQINTDELAY} \times PF)$

$< t_{DQS} - t_{DQ} < (-T_{IOICKP} \times PF) + t_{QH} + (t_{DQINTDELAY} \times PF) - (t_{DQSINTDELAYMAX} \times PF)$

$1.38 \times 0.4 - (1.543)0.4 + (0.5 + (0.00 \times 0.4)) < t_{DQS} - t_{DQ} <$

$- (-0.81)0.4 + ((0.45 \times 5) - 0.5) + ((0.0 \times 0.4) - (0.4 \times 1.556))$

$0.435 < t_{DQS} - t_{DQ} < 1.45$ $\rightarrow$ Equation 2

Combining equation 1 and 2, the range for $t_{DQS} - t_{DQ}$ becomes

$0.435 < t_{DQS} - t_{DQ} < 1.004$

A typical midpoint of 0.75 ns is chosen as the trace difference between $t_{DQS} - t_{DQ}$.

A sufficient data window is necessary to meet the setup and hold requirements at the FPGA and also take care of the package skew inside the FPGA.

Setup and Hold requirements at the DDR registers, package skew, and DQS internal skew (DQS_INT_SKEW) affect the data margin inside the FPGA. Hence, the available margin at the FPGA = Data valid window at the memory – ((Setup + Hold at FPGA) + Worst case package skew + DQS_INT_SKEW).

Data valid window at the memory from the memory specification is equal to ($t_{QH} - t_{DQSQ}$).

t_{QH} is equal to $MIN_{(CH,CL)} - 0.5$ ns

$t_{QH} = (0.45 \times 5) - 0.5 = 2.25 - 0.5$ ns $= 1.75$ ns

Data valid window at the memory = 1.75 ns – 0.5 ns = 1.25 ns

A maximum of 100 ps is substituted for the internal DQS skew.

Available margin at the FPGA = 1.25 – (1.38 – 0.81 + 0.115 + 0.100) = 0.465 ns

Write Timing Analysis

Since the trace delay on the DQS signal is skewed by a typical value of 0.75 ns, the controller needs to make sure that during a Write cycle, the setup and hold requirements at the memory (shown in Figure 15) are met. Also, the available data margin for the Write operation must be calculated.

For an XC2V3000 -5 speed grade FF1152, the data window available at the pins of FPGA for one data period ($t_{CK}/2$) is calculated as:

$$DataValidWindow = \frac{t_{CK}}{2} - (T_{PKGSKEW} + T_{CLKTREESKEW} + T_{DUTYCYCLEDISTORTION} + T_{JITTER})$$

$$= 2.5 - (0.115 + 0.1 + 0.14 + 0.200) = 1.945ns$$

As shown in Figure 15, t_{DQS_SKEW} is the skew between the center of the data valid window and the additional trace delay added on the data strobe (DQS). The center of the data valid window is a quarter of the clock period. From the Read timing, a typical 0.75 ns trace difference is selected between t_{DQS} and t_{DQ}.

$$t_{DQSSKEW} = \frac{t_{CK}}{4} - (t_{DQS} - t_{DQ}) = (1.25 - 0.75)ns = 0.5ns$$

The data valid window at the memory = 1.945 – t_{DQS_skew} = 1.945 – 0.5 = 1.445 ns
Of this, 1 ns is needed to meet setup and hold requirements at the memory. Hence,

Available margin at the memory = 1.445 – 1.00 = 0.445 ns.

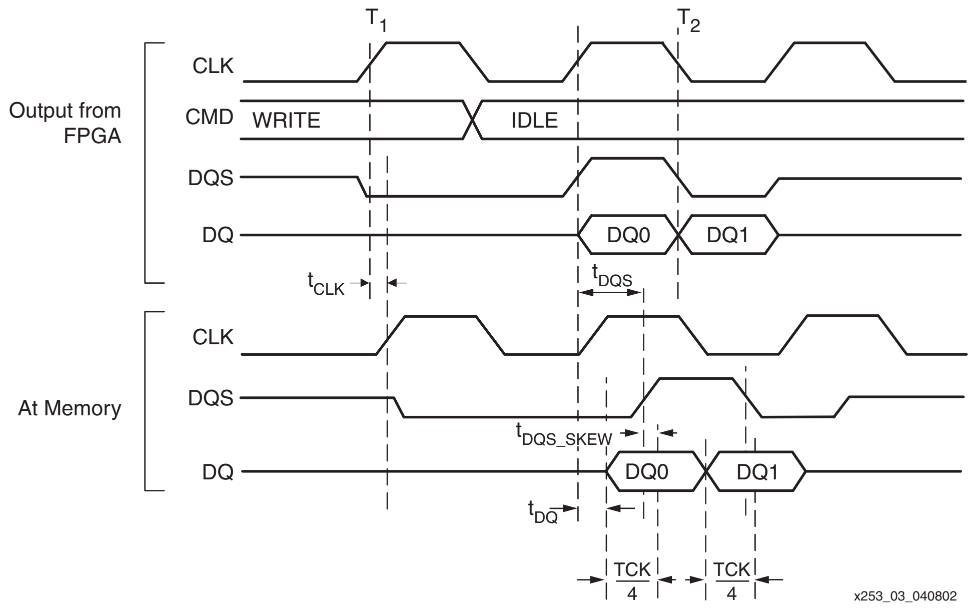

Figure 15: **Timing Diagram of Memory Setup And Hold Requirements**

Improving the Write Data Margin

Using a clock with a fixed phase shift for the data during a Write cycle, helps improve data margin. By eliminating the DQS skew, DQS is aligned with respect to the center of the data window and there is more margin. The required phase shift is calculated to compensate for the DQS skew.

For enough margin during a Write cycle, the data gets aligned with DQS by using a negative phase shifted clock for the data.

In this case, the available data margin is calculated. Since the phase shift compensates for the T_{DQS_SKEW}, the Data valid window at the memory is calculated as:

$$\begin{aligned}\text{Data valid window at memory} &= 1.945 - (t_{DQS\_SKEW} + \text{CLKIN\_CLKFB\_PHASE}) \\ &= 1.945 - (0 + 0.100)\text{ ns} \\ &= 1.845\text{ ns}\end{aligned}$$

The memory setup and Hold timing requirements account for 1 ns of this value.

The available margin at memory = 1.845 ns – 1.00 ns= 0.845 ns

Another timing requirement that must be met during the Write cycle is the relationship between CLK and DQS. This requirement is specified as the DQS Low to High setup time (t_{DQSS}) and is the time from the rising edge of the CLK to the rising edge of DQS. The DDR SDRAM parameters are shown in Table 5. Because DQS and CLK are both generated through DDR flip-flops from the same clock, there is originally one clock cycle (t_{CK}) worth of setup time. However, the trace lengths of CLK and DQS as well as the output standard adjustment will adjust this value.

The following equation highlights this relationship:

$$t_{DQSS(MIN)} < t_{CK} + t_{DQS} + T_{OSSTL2(II)} - t_{CLK} - T_{OSSTL2(1)} < t_{DQSS\ (MAX)}$$

Solving using the values in Table 4 and Table 5 results in the following allowable trace difference between DQS and CLK.

$$-0.64\text{ ns} < t_{DQS} - t_{CLK} < 1.86\text{ ns}$$

Read Recapture Timing Analysis

During a memory Read, data is captured by the IOB flip-flops with the DQS signal. Because DQS is strobing in nature and not a free running clock, there is no guarantee of a successive clock edge moving the data from the IOB into the second stage of the data path. Therefore, the data must be recaptured from DQS to another clock domain.

This reference design uses a phase-shifted version of the user clock to do this data recapture. In order to use this, the designer must calculate the required phase shift. This is outlined in Figure 16. The DDR SDRAM clock is generated by forwarding the internal user clock though the IOB DDR flip flops. This clock will travel from the FPGA to the DDR SDRAM (t_{CLK}).

The DDR SDRAM specification states that upon receiving the clock, the memory will output the DQS signal within $\pm t_{DQSCK}$. The DQS signal travels from the DDR SDRAM to the FPGA (t_{DQS}), where it is routed to the IOB flip-flops ($t_{DQS\_INT\_DELAY}$).

The phase shift feature of the DCM is used to align the recapture clock with the DQS signal inside the FPGA. Because data is transferred from the DQS domain to the recapture clock domain, the recapture clock should be positioned at the earliest possible arrival of DQS. This ensures the greatest time for the clock domain transfer. The following equation is for the recapture clock phase shift.

$$\text{Target phase shift} = (T_{ICKOFDCM} \times PF) + t_{CLK(MIN)} + t_{DQS(MIN)} + t_{DQSINTDELAY(MIN)}$$

The timing relationship between DQS and the recapture clock must be constrained. Under best case conditions, there will be one clock period to transfer from the DQS to the recapture clock domain. Under worst case conditions, the transfer from the DQS to the recapture clock domain will have one period minus the difference between the maximum and minimum path timing. Using the prorating factor of 0.25 gives the following equation:

$$\text{DQS to RCLK} = t_{CK} - \text{Phase\_shift}_{(MAX)} - \text{Phase\_shift}_{(MIN)}\text{ where,}$$

$$\text{Phase Shift}_{(MAX)} = T_{ICKOFDCM} + t_{CLK(MAX)} + t_{DQSCK(MAX)} + t_{DQS(MAX)} + t_{DQSINTDELAY(MAX)}$$

$$\text{and Phase Shift}_{(MIN)} = \text{Target Phase Shift.}$$

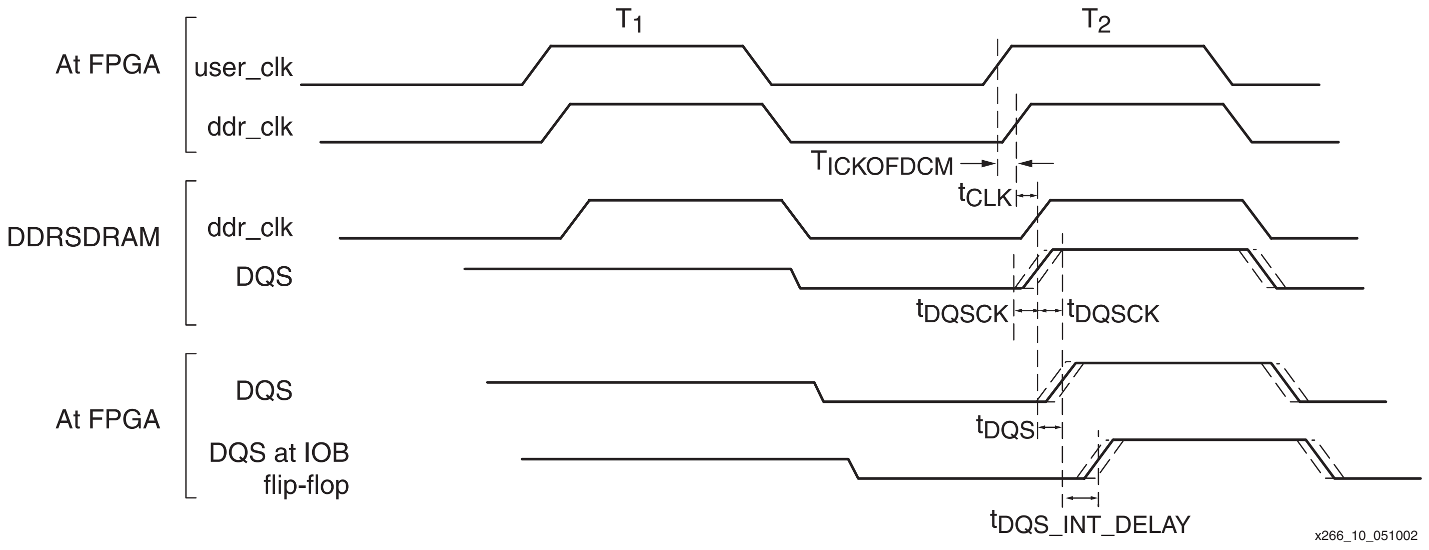

Figure 16: **Clock to DQS Output Delay Time**

Control Timing Analysis

All address and control signals are generated by the controller on the falling edge of CLK. This naturally creates a half-cycle setup and hold at the memory. These values will however be offset by the trace delay of the clock and the command and address signals (t_{CMD}). The command and address setup and hold time is listed in the table.

$$t_{SU} = t_{CK}/2 + t_{CLK} - t_{CMD} > t_{IS}$$

$$t_{HO} = t_{CK}/2 + t_{CMD} - t_{CLK} > t_{IH}$$

Solving these equations for ($t_{CLK} - t_{CMD}$) gives the following equation:

$$t_{IS} - t_{CK}/2 < t_{CLK} - t_{CMD} < t_{CK}/2 - t_{IH}$$

$$-1.5\text{ ns} < t_{CLK} - t_{CMD} < 1.5\text{ ns}$$

Board Considerations

The following system design analysis should be done before completing board layout.

1. An IBIS simulation or H-SPICE simulation should be performed to evaluate signal integrity and switching characteristics on the signals at both the FPGA and the memory device.
2. A careful timing analysis must be completed using the duty cycle seen in simulation.
3. Carefully analyze the trace differences required between DQS and DQ.
4. Place the DQS and DQ pins on the FPGA in accordance with the rules mentioned in Pinout Details for Using Local Clock Distribution.
5. The SSO guidelines must be followed when placing the pins.
6. Follow the power-up sequencing specified by the memory vendor. The targeted memory device in the reference design requires power to simultaneously be applied first to V_{DD} and V_{DDQ}, and then to V_{REF}.

Maintain an LVCMOS Low level on DDR_CKE during power-up and configuration.

Design Implementation

The design targets an XC2V3000 –5 FF1152 device. The design uses three DCMs, one for providing the clock and the DQS to the memory, one for phase shifting the write data to center align the data with DQS at the memory, and the third DCM to generate the phase shifted recapture clock to capture data. The following tools were used during design implementation:

- Synthesis – Synplify Pro. 7.3.3
- Xilinx FISE 6.1 SP2
- Placer Effort level – set to High

Resource utilization on the XC2V3000 –5 FF1152 is shown in Table 7:

Table 7: **Resource Utilization**

Resource	Number Used
DCMs	3
Slices	940
BUFGMUXes	5
Constrained IOBs	32 – DQ bits and 4 – DQS bits
Performance	200 MHz

Hardware Verification

This design has been tested on hardware for a performance of 200 MHz by externally delaying the DQS signal on the board. The tests were conducted on an XC2V1000 –5 FG456 device interfacing to a Micron MT46V32M8 device. The DQS, DQ, and the CLK signals used SSTL2_C2_DCI, while the address and control signals were configured for SSTL2_C1_DCI.

Reference Design Files

VHDL reference design files are available for downloading from the Xilinx web site: http://direct.xilinx.com/bvdocs/appnotes/xapp253.zip

Conclusion

The design uses external delay on the DQS signal to delay the Data Strobe with respect to Data (DQ) in order to capture the data in the IOBs. For interfaces where external delays are not preferred, the scheme to internally delay and calibrate the DQS signal as mentioned in XAPP678 and XAPP688 is recommended.

The Virtex-II DDR registers make an ideal choice for interfacing with DDR SDRAMs. The VHDL reference design is a 32-bit version of the controller.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/12/01	1.0	Initial Xilinx release by authors Jennifer Tran and Ratima Kataria.
07/16/02	2.0	Update of application note and reference design to achieve 400 Mb/s data transfers.
04/25/03	2.1	Minor edit to "Control Timing Analysis" section.
12/18/03	2.2	Updates throughout the document.
06/01/04	2.3	Added "Reference Design Files" section.