XAPP200 (v2.3) March 21, 2000

Synthesizable 1.6 GBytes/s DDR SDRAM Controller

Author: Jennifer Tran

Summary

The DLLs and the SelectI/O™ features in the Virtex™ architecture and Spartan™-II family make it the perfect choice for implementing a controller of a Double Data Rate (DDR) SDRAM. This application note describes the reference controller design for a 64-bit DDR SDRAM. At a clock rate of 100 MHz, and 64-bit data changing at both clock edges, a peak bandwidth of 1.6 GBytes/s is obtained. The reference design is synthesizable and achieves 100 MHz performance with auto place and route tools.

Introduction

With microprocessors getting faster every year, memory architectures must also improve to enhance the overall system performance. DDR SDRAM and RDRAM (Direct RAMBUS DRAM) are the top two contenders for this next generation of SDRAM.

DDR SDRAM has a peak bandwidth of 1.6 GBytes/s with 64-bit data lines and a 100 MHz clock frequency. While RDRAM, also with a peak bandwidth of 1.6 GBytes/s, but with 16-bit lines and a 400 MHz clock frequency is a revolutionary change, DDR SDRAM is a natural evolution from the existing SDRAM architecture.

The main advantage of DDR SDRAMs over RDRAMs is the use of the basic system infrastructure developed for PC-100. This eliminates the numerous design changes required by different "packet" protocols. DDR SDRAM was approved as a JEDEC (Joint Electron Device Engineering Council) standard in February, 1998, and is currently supported by most major SDRAM vendors.

This application note describes the DDR SDRAM controller design implemented in a Virtex series device. The first section briefly reviews the DDR SDRAM basics. The following section describes the SDRAM controller in detail. The final section summarizes the results.

DDR SDRAM Review

The DDR SDRAM is an enhancement to the traditional Synchronous DRAM. It supports data transfers on both edges of each clock cycle, effectively doubling the data throughput of the memory device.

The DDR SDRAM operates with a differential clock: CLK and $\overline{\text{CLK}}$ (the crossing of CLK going High and $\overline{\text{CLK}}$ going Low will be considered as the positive edge of the CLK). Commands (address and control signals) are registered at every CLK positive edge. Input data is registered on both edges of the DQS (data strobe), and output data is referenced to both edges of DQS, as well as to both edges of CLK.

A bidirectional data strobe is transmitted by the DDR SDRAM during reads and by the memory controller during writes. DQS is edge-aligned with data for reads, and center-aligned with data for writes.

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 ACTIVE command are used to select the bank and row to be accessed. 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.

The DDR SDRAM provides for programmable read or write burst lengths of 2, 4, or 8 locations. An AUTO PRECHARGE function may be enabled to provide a self-timed row precharge that is initiated at the end of the burst access.

As with standard SDRAMs, the pipelined, multibank architecture of DDR SDRAMs allows for concurrent operation, thereby providing high effective bandwidth by hiding row precharge and activation time.

All inputs are SSTL_2, including clock signals. All outputs are also SSTL_2, Class II.

Burst Read Operation

The burst read operation in DDR SDRAM is done in the same manner as the current SDRAM. The burst read command is issued by asserting $\overline{CS}$ and $\overline{CAS}$ Low while holding $\overline{RAS}$ and $\overline{WE}$ High at the rising edge of the clock (CLK) after T_{RCD} from the bank activation. The address inputs determine the starting address for the burst. The mode register sets the type of burst (sequential or interleave) and the burst length (2, 4, 8). The first output data is available after the $\overline{CAS}$ latency from the read command, and the consecutive data are presented on the falling and rising edge of DQS until the burst length is completed.

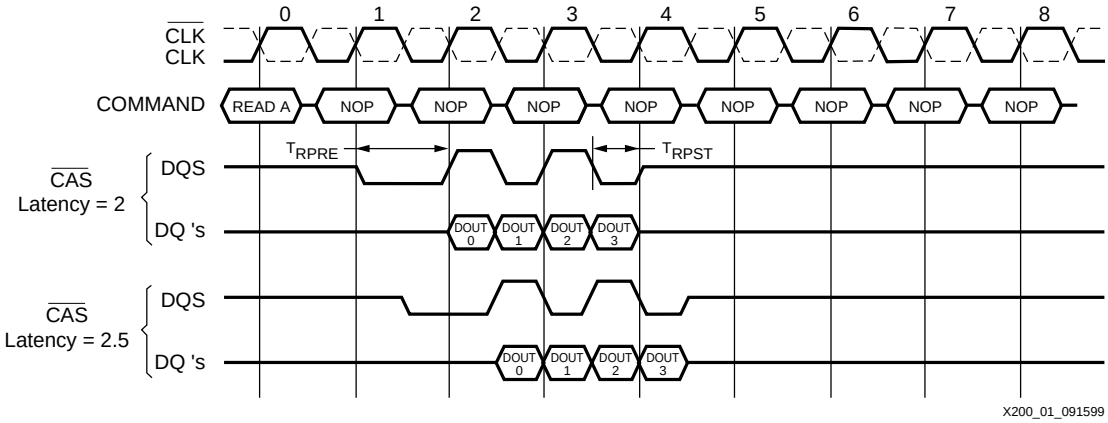

Figure 1: **Burst Read Operation Timing**

Burst Write Operation

The burst write command is issued by having $\overline{CS}$, $\overline{CAS}$ and $\overline{WE}$ Low while holding $\overline{RAS}$ High at the rising edge of the clock (CLK). The address inputs determine the starting column address. There is no write latency relative to DQS required for burst write cycle. The first data of a burst write cycle must be applied on the DQ pins T_{DS} (data-in setup time) prior to data strobe edge. The data strobe signal is enabled after T_{DQSS} from the rising edge of CLK issued by the WRITE command. The remaining data inputs must be supplied on each subsequent falling and rising edge of Data Strobe until the burst length is completed. When the burst has been finished, any additional data supplied to the DQ pins will be ignored.

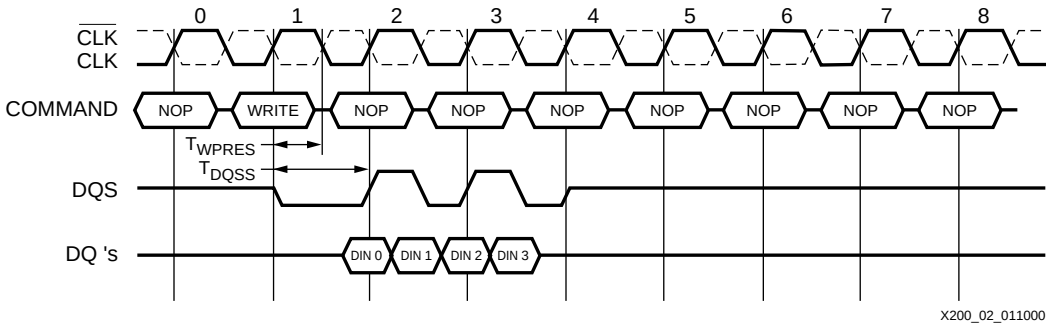

Figure 2: **Burst Write Operation Timing**

DDR SDRAM Controller

The DDR SDRAM controller design features:

- Programmable burst lengths of 2, 4, and 8
- Programmable CAS latency of 1.5, 2, 2.5 and 3
- Burst length applies to both read and write
- Supports the following DDR SDRAM commands: LOAD_MR, AUTO_REFRESH, PRECHARGE, ACT ROW, READA, WRITEA, BURST_STOP, and NOP
- Interfaces with DDR SDRAM at 100 MHz, double data rate (200 Mbits/s/pin)
- Uses CLK instead of DQS to receive data from the DDR
- Uses clock DLLs to provide zero clock skew between user, FPGA, and DDR SDRAM clocks
- Interfaces with DDR SDRAM using SSTL2 I/Os

Figure 3 shows the different blocks in the top level reference design. The user_int module just contains the I/O registers to latch system signals coming into the FPGA. The ddr_ctrl module contains the DDR SDRAM controller, including the I/Os to interface with the DDR SDRAM.

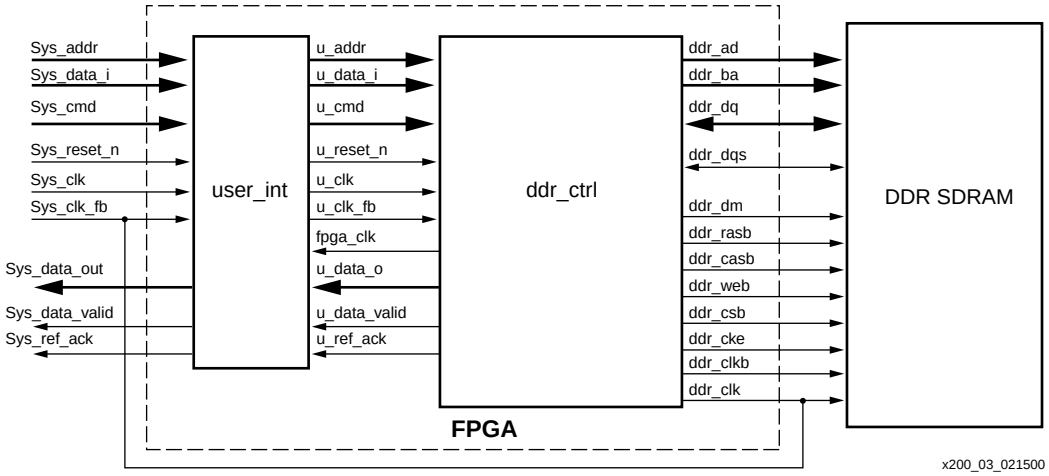

Figure 3: **Top Level Block Diagram**

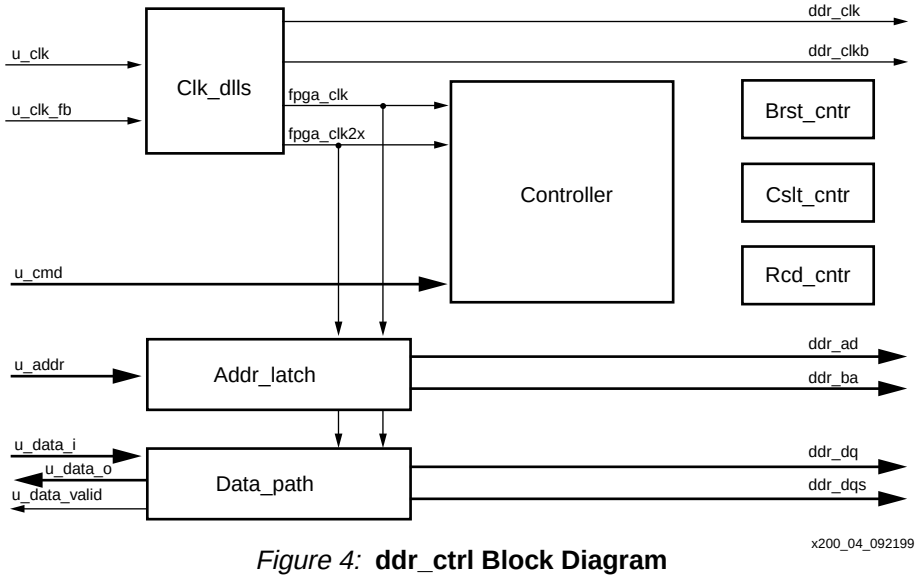

Figure 4: **ddr_ctrl Block Diagram**

Table 1: **ddr_ctrl Pin Description**

	Pin Name	Pin Direction	Width	Description
Interface to User Logic	u_data_i	In	128	Write data
	u_addr	In	22	Address
	u_cmd	In	7	Command for controller
	u_reset_n	In	1	Reset, active Low
	u_clk	In	1	Input clock
	u_clk_fb	In	1	Feedback clock, needs to connect to ddr_clk
	u_data_o	Out	128	Read data
	u_data_valid	Out	1	When data on u_data_o is valid, u_data_valid = 1.
	u_ref_ack	Out	1	Refresh acknowledge
Interface to DDR SDRAM	ddr_ad	Out	12	Address
	ddr_dq	In/Out	64	Data
	ddr_dqs	In/Out	2	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	2	Data mask

Controller Operation

This section describes how the user would issue different DDR SDRAM commands through the controller.

NOP

Set u_cmd = 0000001

Extended Mode Register Set (EMRS)

Set u_cmd = 0000010

Set u_addr[20] = 1 (BA0 = 1)

Mode Register Set (MRS)

Set u_cmd = 0000010

Set u_addr[2:0] = burst length

Set u_addr[3] = burst type

Set u_addr[6:4] = CAS latency

Set u_addr[7] = TM (0: normal mode, 1: test mode)

Set u_addr[8] = DLL (0: no DLL reset, 1: DLL reset)

Set u_addr[20] = 0 (BA0 = 0)

Auto Refresh

Set u_cmd = 01000000

A refresh counter is not included in the reference design. The user should have a refresh counter which periodically issues a refresh command to the DDR SDRAM. The time between refresh should be $T_{REF}/(T_{CLK}$ x number of rows)

Precharge all banks

Set u_cmd = 0010000

The ddr_cntrl will automatically set A_{10} to 1

Burst stop:

Set u_cmd = 1000000

Write with autoprecharge:

Set u_cmd = 0001000

Set u_addr = {ba, row_addr, col_addr}

Put first 128-bit data on u_data_i

Put subsequent data on u_data_i on every clock cycle until the end of burst

Read with autoprecharge:

Set u_cmd = 0000100

Set u_addr = {ba, row_addr, col_addr}

u_data_o should be available after
(T_{RCD} + CAS latency + 6) clock cycles

u_data_valid is high when u_data_o contains valid read data.

Clock DLLs

The top portion of the diagram in Figure 5 shows two DLLs used to deskew the clocks for the FPGA and the DDR SDRAM.

The first DLL, DLL_EXT, provides the ddr_clk and ddr_clkb signals to the DDR SDRAM and also receives the clock feedback from the ddr_clk.

The DDR SDRAM operates on the differential clock (CLK and CLKB). It is important to match the delay of ddr_clk and ddr_clkb both inside the FPGA and on the board. In the reference design, ddr_clk and ddr_clkb are assigned to pins C30 and E28 of the V300BG432 where the routings from the DLL to these two pins are identical.

The second DLL, DLL_INT, provides both fpga_clk and fpga_clk2x internal to the FPGA, its feedback is from fpga_clk. Both fpga_clk and fpga_clk2x use the global clock networks (denoted by BUFG) to provide low skew clocks to the entire FPGA.

Because the DDR SDRAM clock goes through an OBUF, one DLL can not be used to provide both FPGA and DDR SDRAM clocks.

Using two DLLs with the same clock input, but separate feedback signals, achieves zero-delay between input clock, FPGA clock, and DDR SDRAM clock.

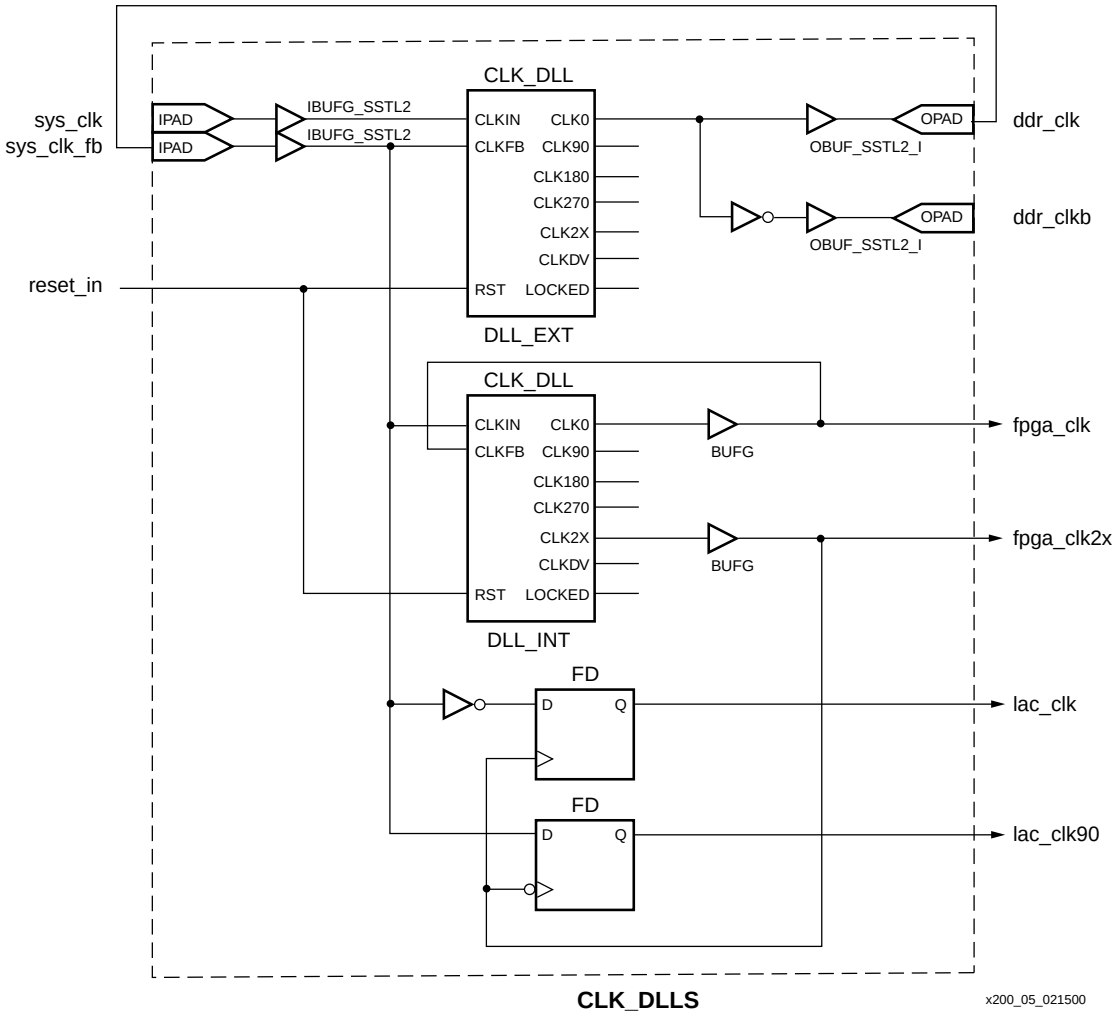

Figure 5: **DLLs in the Reference Design**

The bottom section of the CLK_DLLs diagram in Figure 5 shows the generation of logic accessible clocks. In Virtex and Spartan-II devices, the clock networks are dedicated for high-fanout, low-skew clock signals. As a result, the output of BUFGs only drive clock inputs. In this design, the clock signal is used to select between the upper and lower bits of the data bus. This signal needs to connect to the non-clock inputs of the CLB. A "logic accessible clock" (LAC) can be generated by inverting the input clock signal and clocking it with clk2x. LACs can be duplicated by adding more flip-flops, each receiving the inverted LAC. Since the LAC does not use the global clock tree, the net delay can be large if it has many loads.To minimize loads on the logic accessible clocks, simply duplicate the LAC for each load as illustrated in Figure 6.

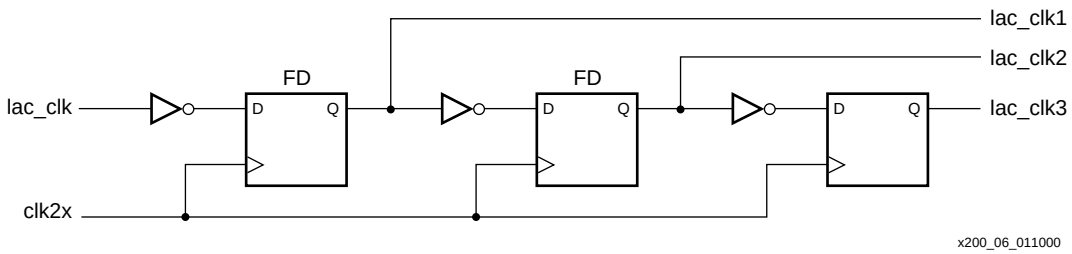

Figure 6: **Duplicating Logic Accessible Clock (LAC)**

Address Mux

The addr_latch module gets its control signals from the controller, and generates row, column, and bank addresses for the DDR SDRAM. The addr_latch also generates burst_max, cas_lat_max values for the burst counter (brst_cntr), and cas-latency counter (cslt_cntr). The controller generates address and control lines on the negative edge of clk to guarantee the hold time on the DDR SDRAM. See the "Board Layout Analysis" on page 11 section for more details of I/O timing.

Data Path

Figure 7 shows the basic data flow for read and write.

For a write cycle, the 128-bit u_data_i is multiplexed into a 64-bit data bus with clk2x, with lac_clk (logic accessible clock) selecting between the higher and lower bits. To minimize I/O delays, the data is registered once more in the IOB before going off-chip through SSTL2 IO buffers. The 3-stated signal for ddr_dq is generated from ddr_write_en, which is mapped into the IOB.

The ddr_dqs signal is generated by a toggle flip-flop which is clocked with the negative edge of clk2x. This ensures the ddr_dqs edges are at the center of ddr_dq. The 3-stated signal for ddr_dqs is also generated from the negative edge of clk2x.

For a read cycle, the 64-bit ddr_dq is clocked at the IOB input flip-flops with clk2x. It is then assembled into 128-bit data with the higher bits registered by the positive edge of the clock, and the lower bits registered with the negative edge of the clock.

All inputs and outputs to the DDR SDRAM are registered in the IOB. This guarantees minimum clock-to-out delay.

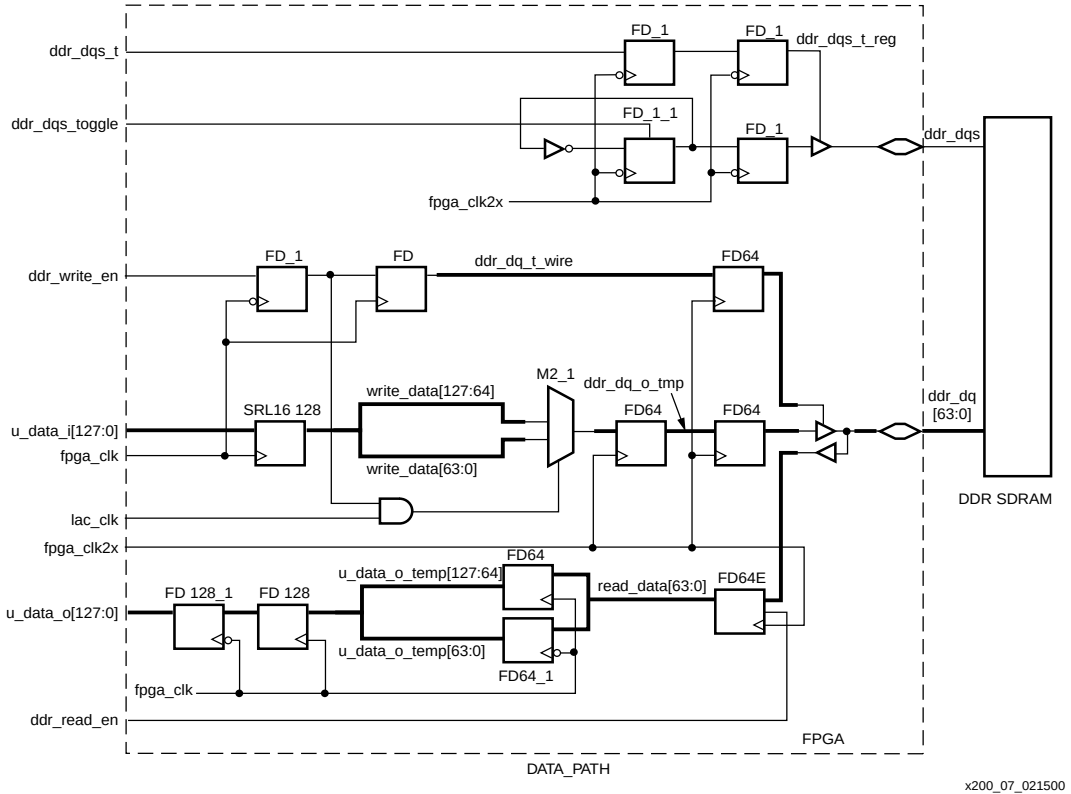

Figure 7: **Data Path**

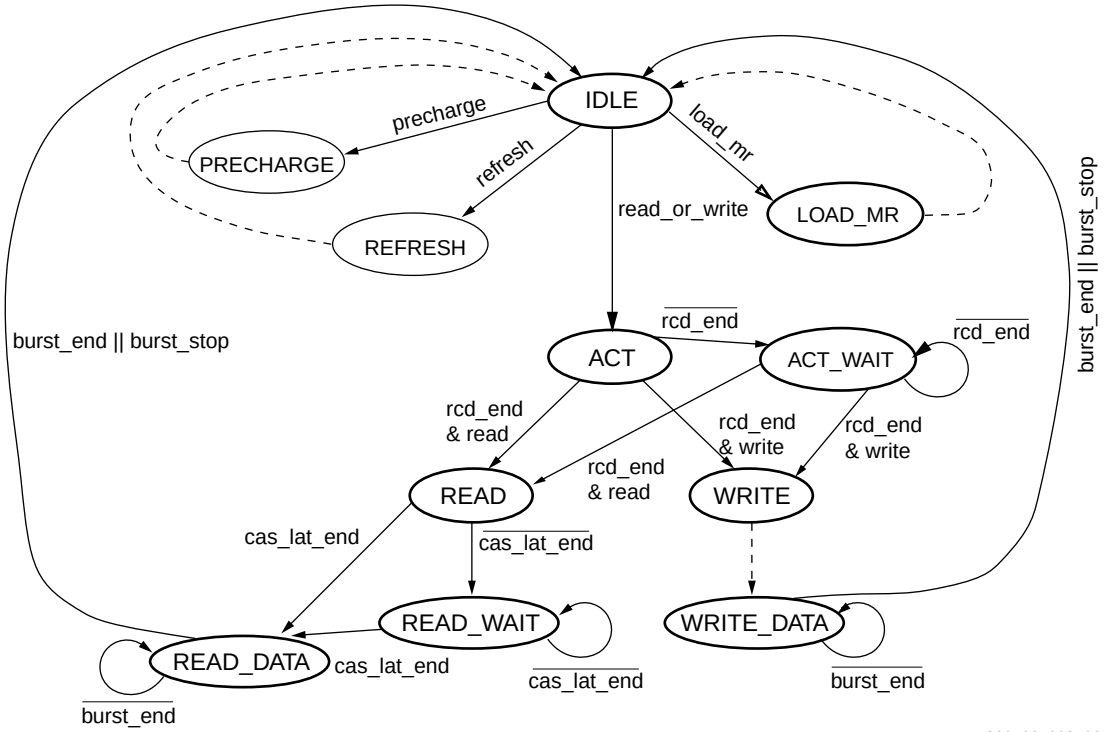

Figure 8: **State Machine Diagram**

Controller State Machine

An overview of the State Machine is shown in Figure 8. The dashed lines indicate an automatic sequence.

The controller is initially in the IDLE state. The next state of the controller could be PRECHARGE, LOAD_MR, REFRESH, or ACT, depending on the required command. The dashed lines in the state machine diagram show an automatic sequence. For read and write, the controller first goes into the ACT state (active row).

During write, the controller goes from ACT to WRITE, then to the WRITE_DATA state. The controller stays in WRITE_DATA until a BURST_STOP command or until the number of data in the burst is written. In this design, the ras-to-cas delay is three clock cycles. If a different ras-to-cas delay is needed, the state machine needs to be changed. Also, the pipeline registers in the data_path module need to be adjusted accordingly.

During read, the controller first goes to the READ state, issuing a read command to the DDR SDRAM. Then the controller goes into READ_WAIT to wait for CAS latency. The CAS latency is programmable by the user by doing a LOAD_MR command. At the end of CAS latency, the controller goes into READ_DATA state where it issues control signals for the data_path module to accept data. At the end of a burst or at BURST STOP command, the controller goes back into IDLE state.

The controller generates rasb, casb, and web for the DDR SDRAM. It also generates all control signals for data_path, addr_latch, and various counters in the design.

All control signals are generated by the clock positive edge, except for ddr_dqs_toggle, and ddr_dqs_t. The ddr_dqs_toggle and the ddr_dqs_t are control signals that generate DQS for the DDR SDRAM. They are generated on the negative edge of clk2x since data strobes are clocked by the negative edge of clk2x.

Timing Diagrams

Write Cycle

Figure 9 shows the timing diagram for writing a burst of eight data words to the DDR SDRAM.

The first four signals are inputs to the DDR controller. At T_1, the write command, DDR SDRAM address, and the first 128-bit data are placed on u_cmd, u_addr, and u_data_i respectively.

The fifth waveform shows the controller's state. The controller goes from IDLE into ACT state at T_2.

The last four waveforms show the signal interfaces to the DDR SDRAM. At T_3, the controller issues the ACT command and the row address to the DDR SDRAM. After T_{RCD} delay (three clock cycles), the controller issues the WRITEA command and column address. The signals ddr_dq and ddr_dqs, are then issued at a double data rate.

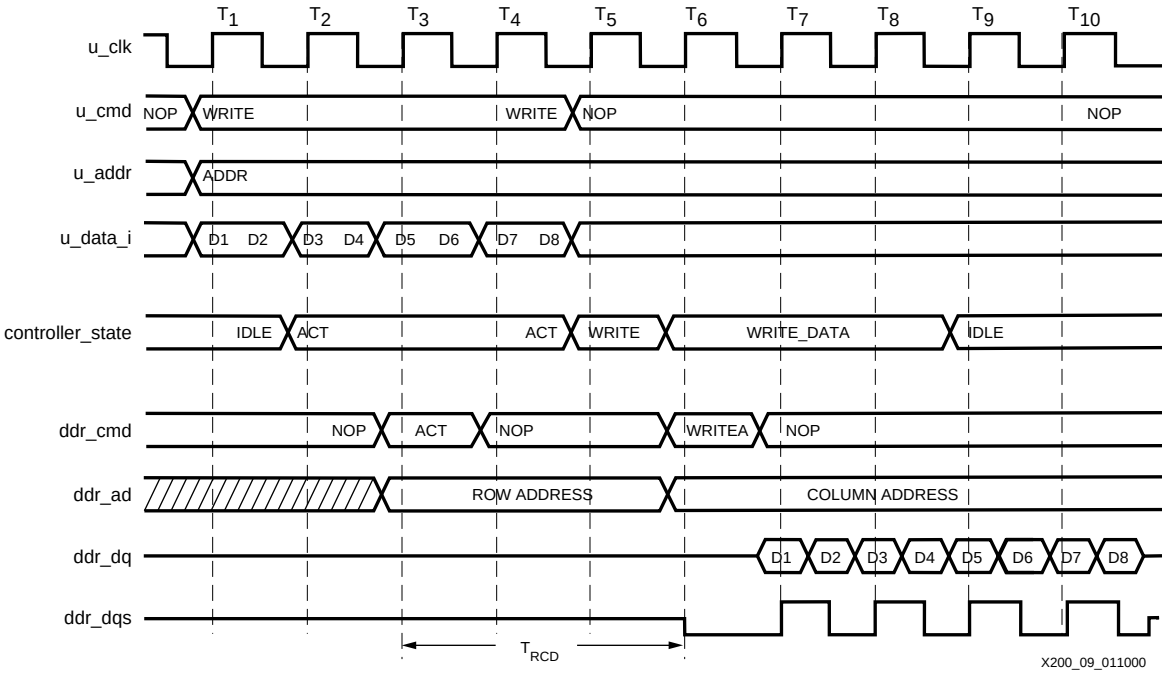

Figure 9: **Write Cycle Timing Diagram**

Read Cycle

Figure 6 shows the timing diagram for reading a burst of eight data words from the DDR SDRAM.

At T_1, the read command and the SDRAM address are placed on u_cmd and u_addr.

At T_2, the controller goes from IDLE to ACT state.

At T_3, the controller issues an ACT command and row address to the DDR SDRAM.

After T_{RCD} delay (three clk cycles), the controller issues a READA command and column address.

After CAS latency (two clock cycles), the DDR SDRAM presents the data and data strobe at every clock edge until the burst is completed.

The controller receives the data and assembles it back into 128-bit words. The u_data_valid signal is asserted when read data is valid on u_data_o.

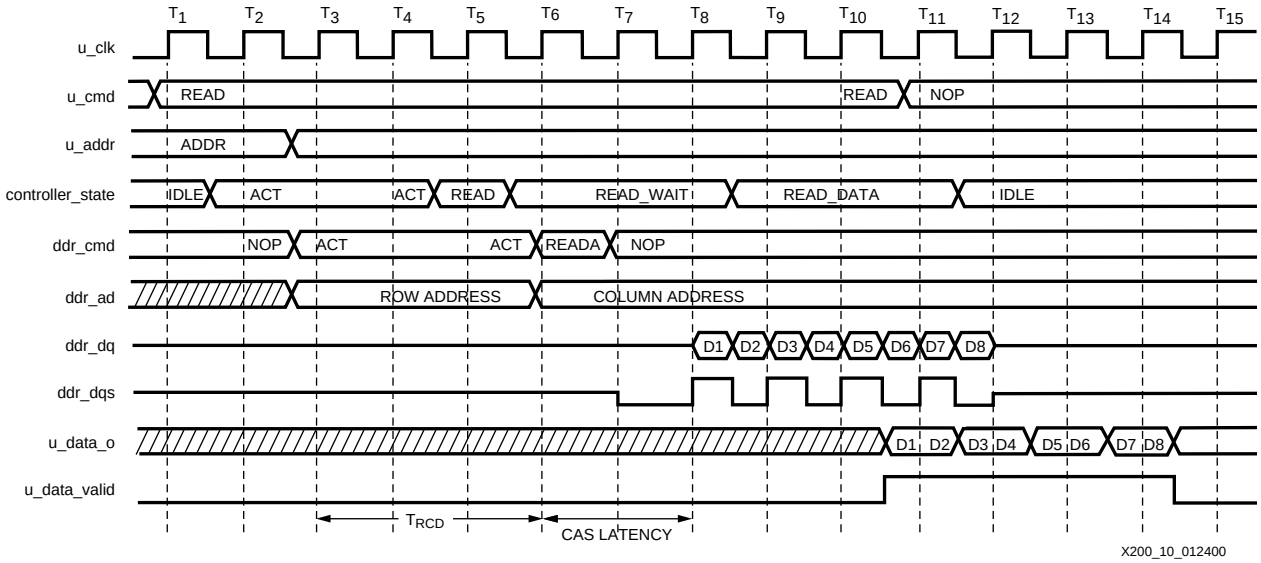

Figure 10: **Read Cycle Timing Diagram**

Design/ Implementation

Having a data path operate at 200 MHz is the design challenge. The following design techniques for controller design are used for operation at the desired speed.

- CLKDLL is used to remove clock skew between the FPGA and the DDR SDRAM.
- BUFG (global clock buffer) is used to drive clock nets in the FPGA.
- Non-DELAY input buffers are used because they are approximately 500 ps faster than the default DELAY buffers.
- All signals interfaced to the DDR SDRAM are registered in the IOB. The 3-stated signals for the DDR SDRAM data are also registered in the IOB.
- To minimize routing delays all critical signals are duplicated to have fanouts of four or less.
- For faster performance, some counters are implemented by using SRL16 (brst_cntr).
- "Logic accessible clock" is used to route to non-clock CLB pins.

Place and Route

The reference design includes a run_par script to show how to implement the design and generate a bitstream. run_par also generates a back-annotated netlist for post-route simulation. A constraint file (top.ucf) is also included in the reference design. The constraint file contains the following:

- A clock period.
- The first lac_clk path.
- All signals interfacing to the DDR SDRAM need to be registered in the I/O. This can be accomplished by having the IOB=TRUE attribute in the constraint file (or using the –pr b option when running map.)
- To get a minimum setup time, all inputs to the FPGA should have a NODELAY attribute.
- To get zero routing skew inside the FPGA, ddr_clk and ddr_clkb must be constrained to adjacent IOBs in the same tile.
- Some versions of the software require all SSLT2 I/Os to be manually placed.

After running place and route, check the trace report (top_par.twr) for any timing violations.

BitGen

By default, the DLL will remove all skew between CLKIN and CLKFB signals, assuming CLKIN is LVTTL. Since CLKIN for DLL_INT is SSTL2, fpga_clk actually lags ddr_clk by approximately 600 ps.

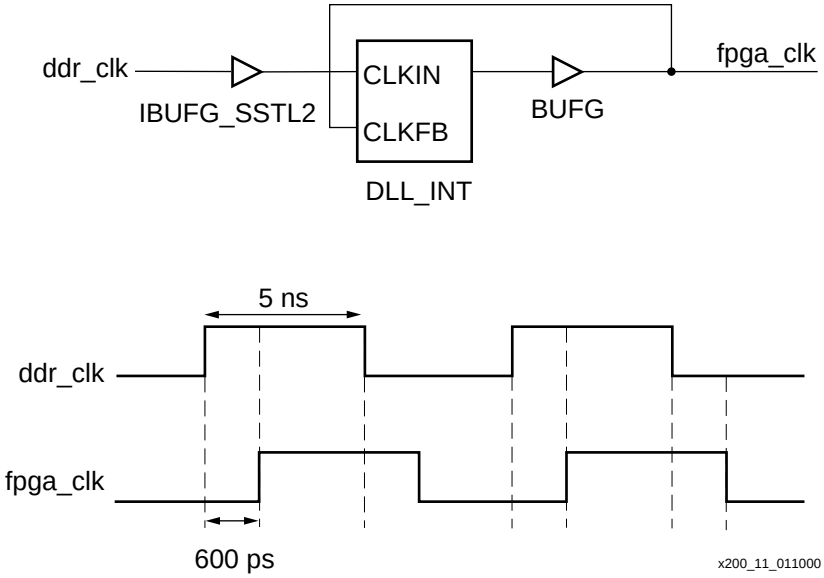

Figure 11: **dll_int Waveforms**

To remove this skew, some delay is added to the DLL_INT feedback path by the following BitGen option: `bitgen -g Ilvl1_mux_20:0 -g Ilvl1_mux_22:1`

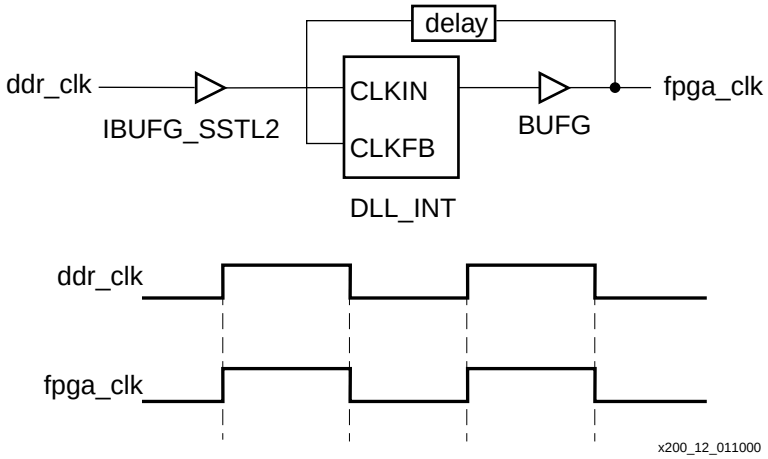

Figure 12: **dll_int Waveforms Adjusted by BitGen Option**

Results

The DDR SDRAM controller design uses 335 CLB slices (11% of an XCV300), two DLLs, two global clock buffers and 89 IOBs. The design is verified with back-annotated simulation at 100 MHz.

Board Layout Analysis

To ensure all setup and hold times are met before planning board layout, the designer must calculate the I/O timing between the FPGA and the DDR SDRAM. Additionally, proper termination must be planned to ensure good signal integrity since all DDR signals are SSTL2. It is strongly recommended that a board design tool be used to simulate the signals. Xilinx

provides IBIS models for all Virtex series and Spartan-II I/Os. SDRAM IBIS models are also available from memory vendors.

The following is a sample I/O analysis using a Virtex -6 speed grade device and a -8 SDRAM to achieve 100 MHz operation.

Read Operation

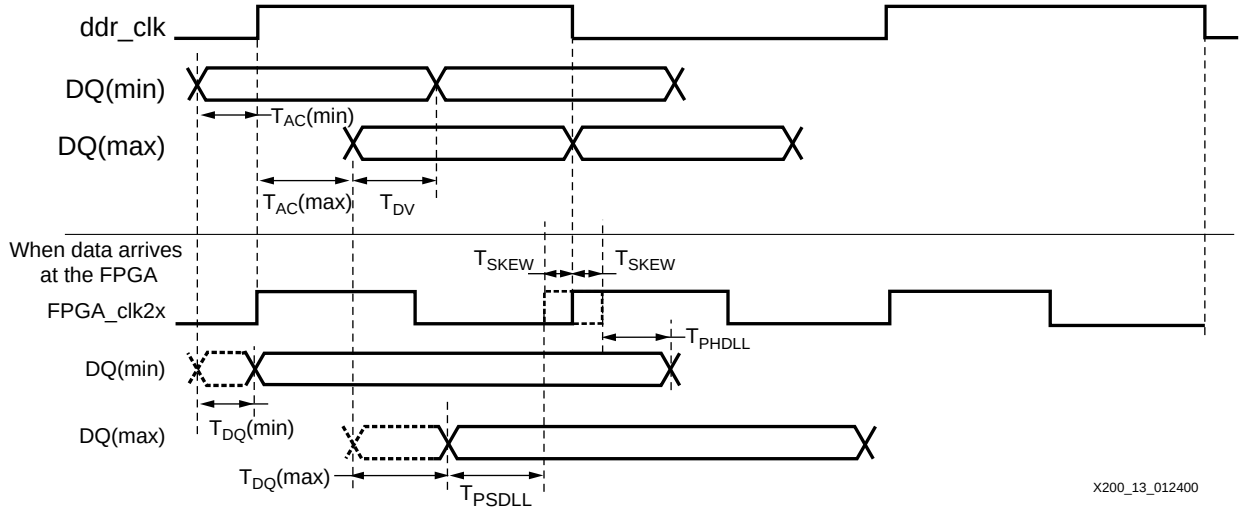

Figure 13: **Read Timing Diagram**

The FPGA uses a clock to receive data in the READ cycle. To calculate T_{DQ} (PC board delay for DQ), the designer must ensureT_{AC} (minimum and maximum) plus T_{DQ} meets both the FPGAs setup and hold requirements. Adding clock skew between ddr_clk and fpga_clk (T_{SKEW}) produces the following equation:

$$T_{SKEW} + T_{PHDLL} - T_{AC}(min) < T_{DQ} < (1/2 \times T_{CK}) - T_{SKEW} - T_{PHDLL} - T_{AC}(max) \quad (1)$$

Example:

Designing with a Virtex -6 speed grade device and a Micron -8 speed grade device, the parameters are outlined in Tables 2 and 3.

Table 2: **Parameters for -8 Speed Grade Micron Device**

Symbol	Parameter	Min	Max	Units
T_{AC}	Data access time from CLK	– 0.8	0.8	ns
T_{DV}	Output data valid	3.5		ns
T_{CK}	Clock cycle time	10		ns

Table 3: **Parameters for -6 Speed Grade Virtex Device**

Symbol	Parameter	Min	Max	Units
T_{PSDLL}	Input setup with DLL (SSTL2)	1.7		ns
T_{PHDLL}	Input hold with DLL (SSTL2)	– 0.4		ns

T_{SKEW} is the clock skew between the ddr_clk and the fpga_clk. The DLL_INT in the FPGA removes all clock delays between ddr_clk at the feedback pin and the internal fpga_clk. DLL output jitter and any clock skew are already included in the FPGA input setup and hold times. Any DDR clock jitter and skew on the board is assumed to be a maximum of 200 ps. Calculating with Equation 1:

$$T_{SKEW} + T_{PHDLL} - T_{AC}(min) < T_{DQ} < (1/2 \times T_{CK}) - T_{SKEW} - T_{PHDLL} - T_{AC}\ (max)$$

$$0.2 - 0.4 + 0.8 < T_{DQ} < 1/2 \times (10) - 0.2 - 1.7 - 0.8$$

$$0.6\ ns < T_{DQ} < 2.3\ ns$$

The result is that T_{DQ} needs to be between 0.6 ns and 2.3 ns. By delaying the ddr_clk by 500 ps on the board, the T_{DQ} can be from 0.1 ns to 1.8 ns.

Write Operation

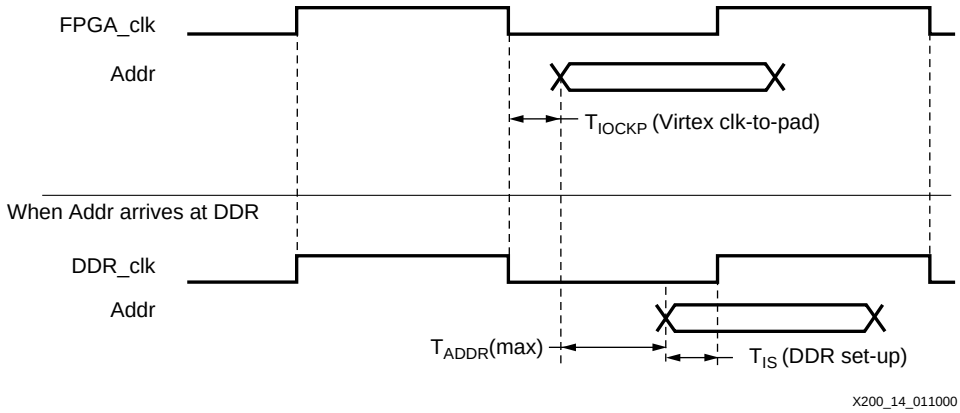

Figure 14: **Write Operation Timing Diagram**

Data Setup and Hold Times

The Data setup and hold is with respect to DQS (instead of clock). From the FPGA, the DQS signals are generated from the negative edge of clk2x and the DQ signals are generated on the positive edge of clk2x. DQ naturally has 1/4 x T_{CK} setup and hold times around DQS edges. T_{DS} and T_{DH} are usually 0.075 x T_{CK}. For a maximum setup and hold margin, keep DQS trace lines the same as DQ trace lines.

Address and Control Setup and Hold Times

In the reference design, to guarantee DDR hold time the address and control signals are generated by the negative edge of the fpga_clk. For the setup time, the FPGA clock-to-pad (maximum) and address/control setup time should be less than T_{CK}. Adding the clock skew produces the following equation:

$$T_{ADDR(MAX)} < 1/2 \times T_{CK} - T_{IOCKP(MAX)} - T_{IS} - T_{SKEW} \qquad (2)$$

Example:

Table 4: **Parameters for -8 Speed Grade Micron (200 MHz) Device**

Symbol	Parameter	Min	Max	Units
T_{CK}	Clock cycle time	10		ns
T_{DS}	DQ to DQS setup	0.075 x T_{CK} = 0.75		ns
T_{DH}	DQ to DQS hold	0.075 x T_{CK} = 0.75		ns
T_{IS}	Address and Control setup	0.15 x T_{CK} = 1.5		ns
T_{IH}	Address and Control hold	0.15 x T_{CK} = 1.5		ns

Table 5: **Parameters for -6 Speed Grade Virtex Device**

Symbol	Parameter	Min	Max	Units
T_{IOCKP}	Clock to pad (SSTL2_I)	0.9	2.4	ns
T_{OJITCC}	DLL output jitter		± 0.06	ns

$$T_{ADDR(MAX)} < 1/2 \times T_{CK} - T_{IOCKP(MAX)} - T_{IS} - T_{SKEW}$$
$$T_{ADDR(MAX)} < 5 - 2.4 - 1.5 - 0.2$$
$$T_{ADDR(MAX)} < 0.9 \text{ ns}$$

If the DDR clock is delayed by 500 ps, then $T_{ADDR(MAX)}$ can be up to 1.4 ns.

Board Layout Summary

- The DQ trace lines can be calculated with the following equation to ensure proper timing for the READ cycle:
  $T_{SKEW} + T_{PHDLL} - T_{AC}(min) < T_{DQ} < (1/2 \times T_{CK}) - T_{SKEW} - T_{PHDLL} - T_{AC}$ (max) (1)
- DQS is centered around DQ during a WRITE operation. DQS and DQ trace lines should be the same length to get maximum margin on setup and hold for the DDR SDRAM.
- Address and control lines are generated from the negative edge of the fpga_clk to guarantee the DDR hold time.
- The maximum trace length for address and control lines can be calculated from the following equation:
  $T_{ADDR(MAX)} < 1/2 \times T_{CK} - T_{IOCKP(MAX)} - T_{IS} - T_{SKEW}$ (2)
- ddr_clk can be delayed to reduce the minimum delay on T_{DQ} and increase the maximum delay on T_{ADDR}.
- ddr_clk and ddr_clkb trace lines should match each other to ensure the same delays. To match the feedback load on ddr_clk, the ddr_clkb can be routed back to an adjacent, unused IOB.
- All signals to the DDR SDRAM are SSTL2. Refer to the Xilinx application notes: **XAPP133** "Using the Virtex SelectI/O Resource" or **XAPP179** "Using SelectI/O Interfaces in Spartan-II FPGAs", and the DDR SDRAM datasheet for more guidelines on termination techniques and simultaneous switching guidelines.
- Xilinx strongly recommends using a board design tool to analyze the traces and check for signal integrity.

Conclusion

This reference design shows how to interface a Virtex series or Spartan-II family FPGA to a DDR SDRAM. The Xilinx DLLs and flexible SelectI/O features eliminate the need for special clock generators and I/O drivers on the board. The ability of Xilinx devices to adjust the skew

between the DDR and the FPGA clock using a bitgen option provides the designer with the added flexibility to tune the interface after the board is built.

The reference design is developed for both Verilog and VHDL and can be easily modified for a different system design requirement. The reference design can be downloaded from the Xilinx web site at: ftp://ftp.xilinx.com/pub/applications/xapp/

64-bit version: xapp200.zip or xapp200.tar.gz

16-bit version: xapp200_16.zip or xapp200_16.tar.gz

References

1. Xilinx Inc., Virtex 2.5V Field Programmable Gate Arrays, Datasheet, 1999
2. Xilinx Inc., XAPP133, Using the Virtex SelectI/O Resource, Application note, 1999
3. Micron Technology Inc., Double Data Rate (DDR) SDRAM, Datasheet, 1999\
4. Xilinx Inc., Spartan-II 2.5V Field Programmable Gate Arrays, Datasheet, 2000
5. Xilinx Inc., XAPP179, Using SelectI/O Interfaces in Spartan-II FPGAs, Application note, 2000

Revision History

The following table shows the revision history for this document.

Date	Version #	Revision
9/28/99	1.0	Initial release.
10/01/99	1.1	Revised Table 1, u_data_o
10/12/99	1.2	Revised Figure 5
1/10/00	2.0	Reformatted and major changes to the FPGA implementation of the Reference design. Added Spartan-II family support.
1/24/00	2.1	Updated Figure 10 & 13, corrected error on last sentence on page 9, and updated reference design locations.
2/18/00	2.2	Modified Figures 3, 5, & 7, and minor edits to text throughout.
3/21/00	2.3	Updated to include Virtex-E Extended Memory devices.