XAPP253 (v1.0) January 12, 2001

Synthesizable 266 MBits/s DDR SDRAM Controller

Author: Jennifer Tran and Ratima Kataria

Summary

The DDR, DCM, and SelectI/O™ features in the Virtex™-II architecture make it the perfect choice for implementing a controller of a Double Data Rate (DDR) SDRAM. The Digital Clock Manager (DCM) provides the required Delay Locked Loop (DLL), Digital Phase Shift (DPS), and Digital Frequency Synthesis (DFS) functions. This application note describes a controller design for a 16-bit DDR SDRAM. The application note and reference design are enhanced versions of XAPP200 targeted to the Virtex-II series of FPGAs. At a clock rate of 133 MHz, 16-bit data changes at both clock edges. The reference design is fully synthesizable and achieves 133 MHz performance with automatic place and route tools.

Introduction

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

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 Joint Electron Device Engineering Council (JEDEC) 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-II 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 and outputs are SSTL_2 Class II, including clock signals.

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 CS and CAS Low while holding RAS and 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 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.The timing is shown in Figure 1.

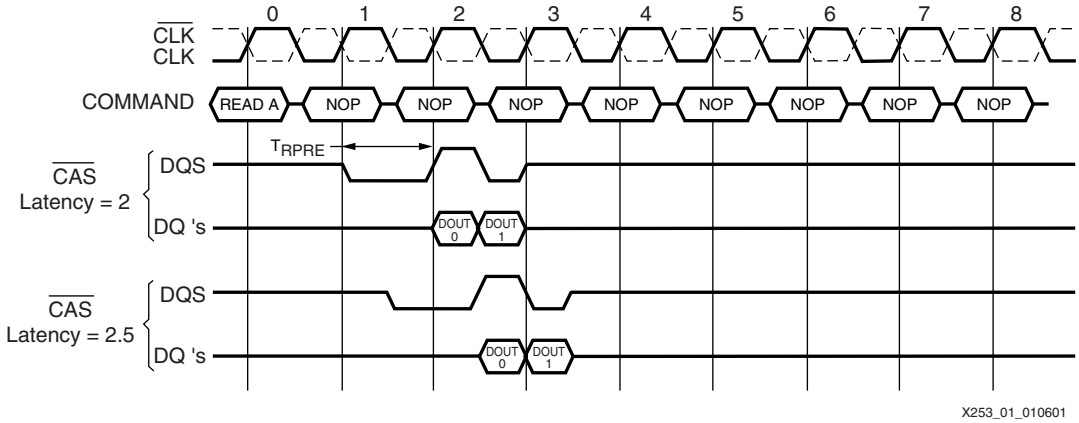

Figure 1: **Burst Read Operation Timing**

Burst Write Operation

The burst write command is issued by having CS, CAS and WE Low while holding 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.The timing is shown in Figure 2.

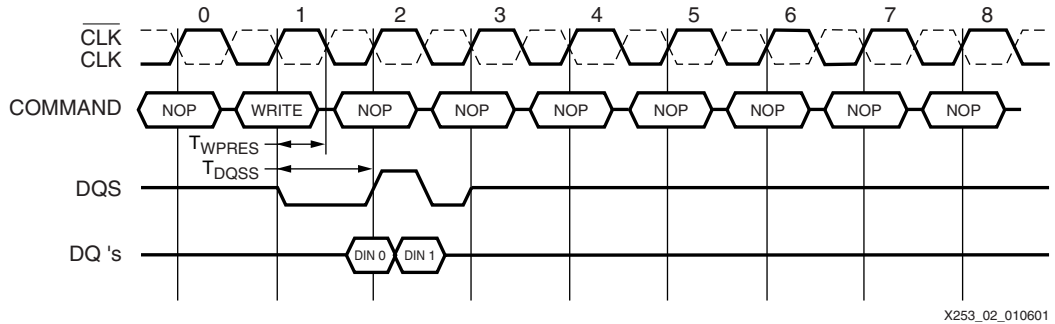

Figure 2: **Burst Write Operation Timing**

DDR SDRAM Controller

The DDR SDRAM controller design features:

- Programmable burst lengths of 2, 4, and 8
- Programmable CAS latency of 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 133 MHz, double data rate (266 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.

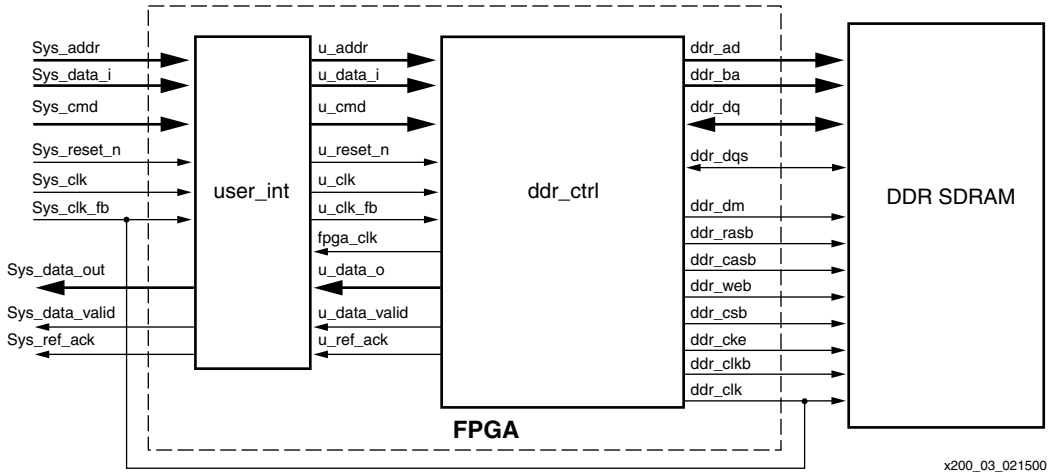

Figure 3: **Top Level Block Diagram**

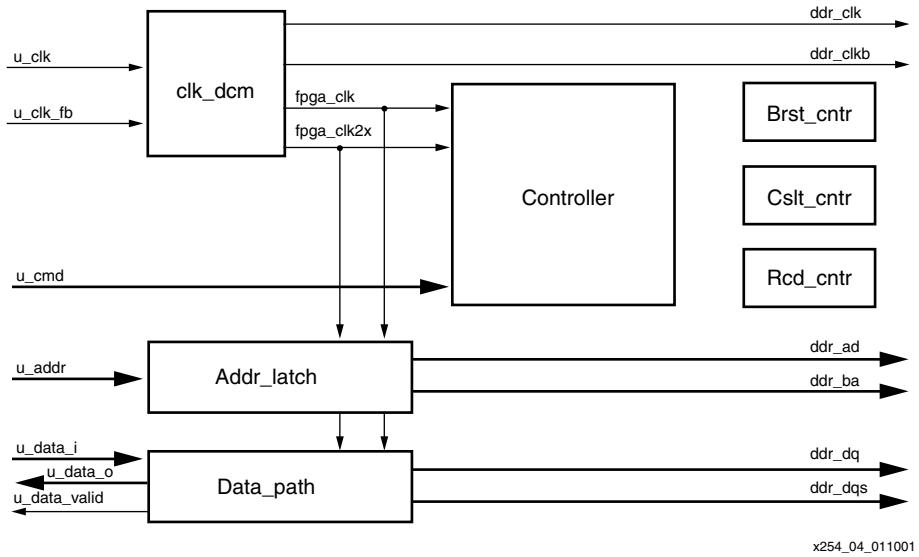

Figure 4: **ddr_ctrl Block Diagram**

Table 1: **ddr_ctrl Pin Description**

	Pin Name	Pin Direction	Width	Description
Interface to User Logic	u_data_i	In	32	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	32	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	16	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 $T_{REF}/(T_{CLK} \times$ 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 (TRCD + CAS latency + 6) clock cycles.

u_data_valid is high when u_data_o contains valid read data.

Clock DCMs

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 DCM is primarily used as a DLL in this design.

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.

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.

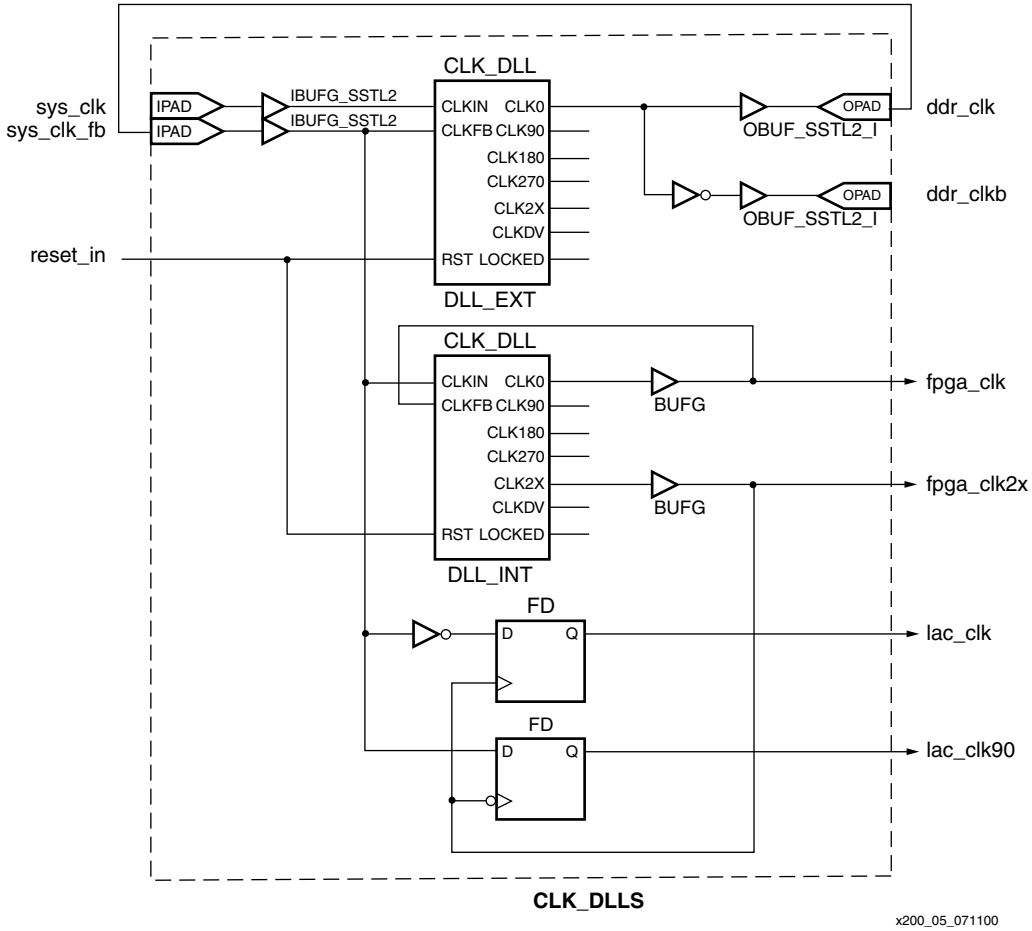

Figure 5: **DCMs Used as DLLs**

The bottom section of the CLK_DLLs diagram in Figure 5 shows the generation of logic accessible clocks. In the reference design, the clock signal needs to connect to the non-clock inputs of the CLBs via general routings, thus the delay can be significant. To guarantee minimal delays on the clock signal going to any non-clock inputs of the CLBs, create a logic accessible clock (LAC) signal by inverting the clock signal and clocking it with clk2x. The reference design duplicates LAC signals for each load to keep delays to a minimum (Figure 6).

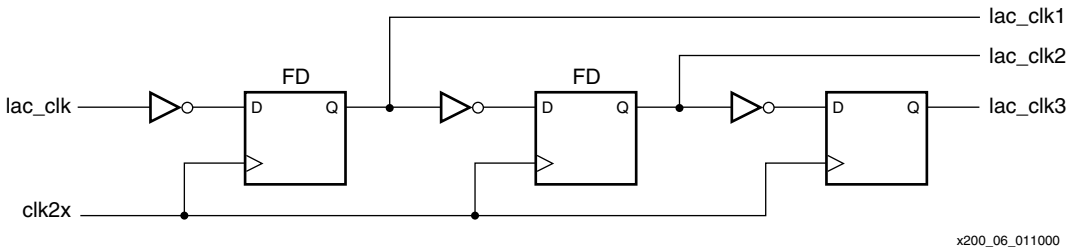

Figure 6: **Duplicating Logic Accessible Clock (LAC)**

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.

Data Path

Figure 7 shows the basic data flow for read and write. For a write cycle, the 32 bit u_data_i is delayed using Shift Registers (SRL16) and then the Double Data Rate registers (FDDRRSE) output 16 bits each on positive edge and negative edge of the clock. The 3-stated signal for ddr_dq is generated from ddr_write_en, which is mapped into the IOB.

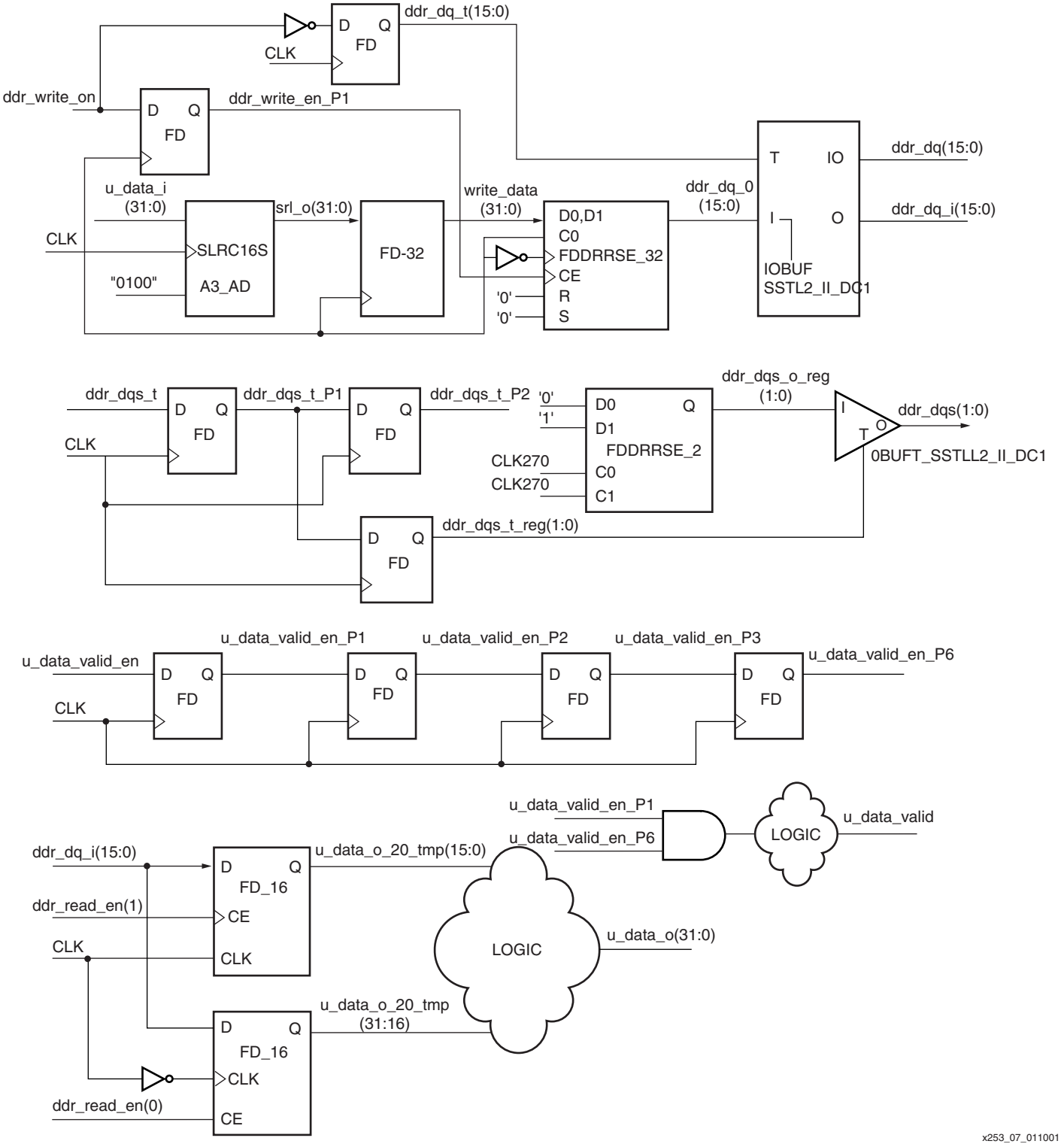

Figure 7: **Data Path**

The ddr_dqs signal is generated using the FDDRRSE. FDDRRSE is clocked by CLK270 to ensure that the ddr_dqs edges are at the center of ddr_dq.

For a read cycle, the 16-bit ddr_dq is clocked in and then assembled into 32-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.

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.

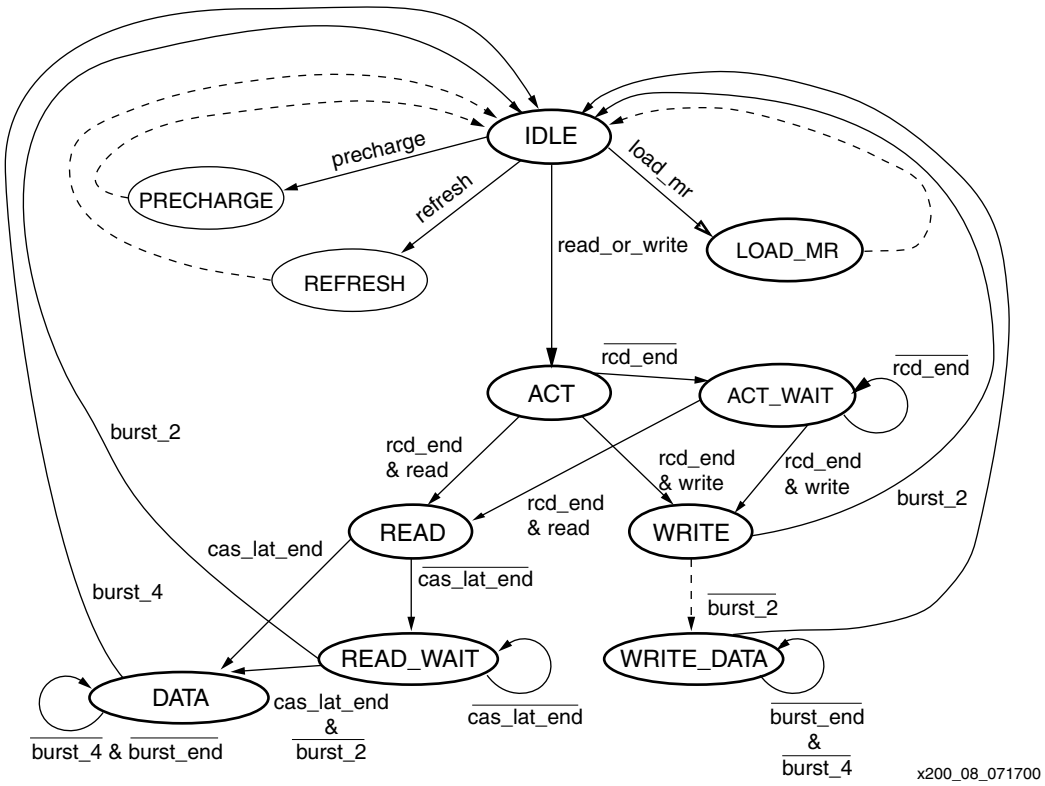

Figure 8: **State Machine Diagram**

Timing Diagrams

Write Cycle

Figure 9 shows the timing diagram for writing a burst of 32 bits of data 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 32-bit data is 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.

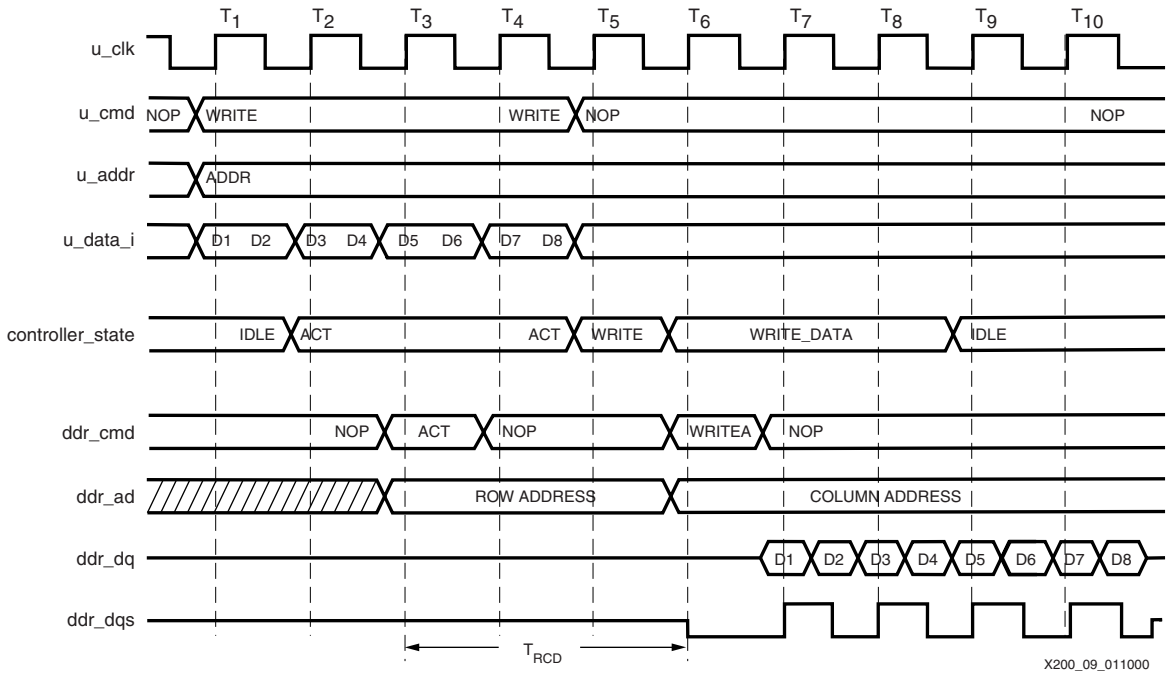

Figure 9: **Write Cycle Timing Diagram**

Read Cycle

Figure 10 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 32-bit words. The u_data_valid signal is asserted when read data is valid on u_data_o.

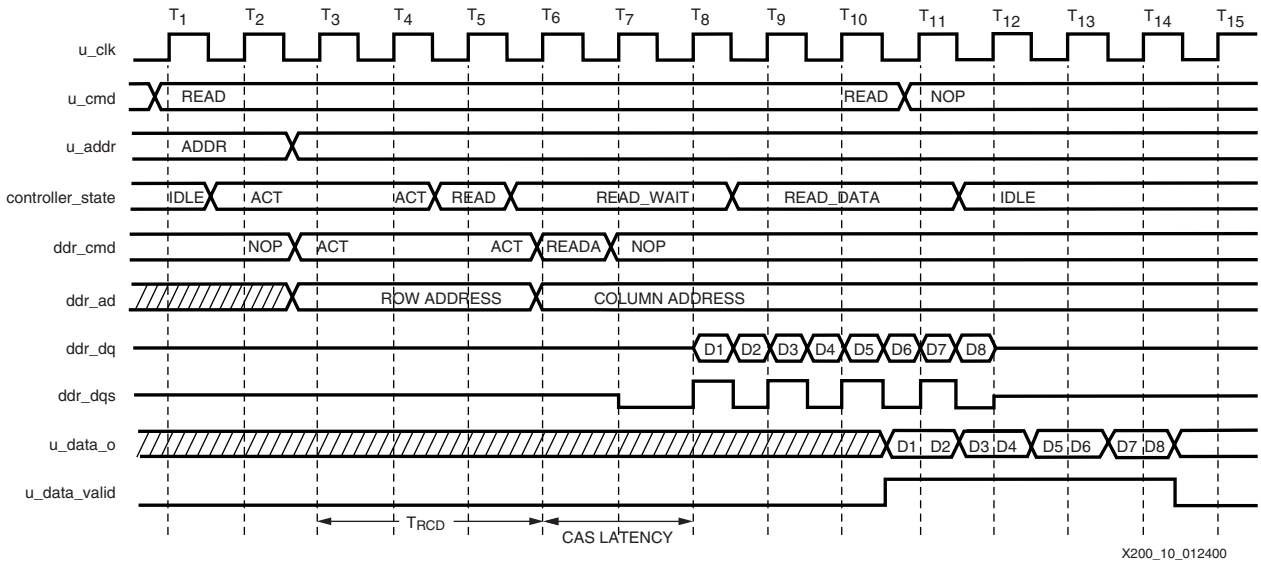

Figure 10: **Read Cycle Timing Diagram**

Introduction to Xilinx DCI

As FPGAs get bigger and system clock speeds get faster, PCB board design and manufacturing has become more difficult. With ever faster edge rates, maintaining signal integrity becomes a critical issue. Designers must make sure that most PC board traces are terminated properly to avoid reflections or ringing.

To terminate a trace, resistors are traditionally added to make the output and/or input match the impedance of the receiver or driver to the impedance of the trace. However, due to the increase in the device I/O counts, adding resistors close to the device pins increases the board area and component count and might even be physically impossible. To address these issues and to achieve better signal integrity, Xilinx developed a new I/O technology for the Virtex-II series, Digital Controlled Impedance (DCI).

DCI uses two multi-purpose reference pins in each bank to control the impedance of the driver or the parallel termination value for all of the I/Os of that bank. The N reference pin (VRN) must be pulled up to V_{CCO} by a reference resistor, and the P reference pin (VRP) must be pulled down to ground by another reference resistor. The value of each reference resistor should be equal to the characteristic impedance of the PC board traces, or should be twice that value (configuration option).

Please refer to "Chapter 2 - Design Considerations - Using DCI" for details. It is beyond the scope of this application note to provide details of DCI usage.

DCI in Virtex-II Hardware

DCI only works with certain single-ended I/O standards and does not work with any differential I/O standard. DCI supports the following Virtex-II standards:

LVDCI, LVDCI_DV2, GTL_DCI, GTLP_DCI, HSTL_I_DCI, HSTL_II_DCI, HSTL_III_DCI, HSTL_IV_DCI, SSTL2_I_DCI, SSTL2_II_DCI, SSTL3_I_DCI, and SSTL3_II_DCI.

To correctly use DCI in a Virtex-II device, users must follow the following rules:

1. V_{CCO} pins must be connected to the appropriate V_{CCO} voltage based on the IOSTANDARDs in that bank.
2. Correct DCI I/O buffers must be used in the software either by using IOSTANDARD attributes or instantiations in the HDL code.

3. External reference resistors must be connected to multi-purpose pins (VRN and VRP) in the bank cannot be used as regular I/Os. Refer to the Virtex-II pinouts for the specific pin locations. Pin VRN must be pulled up to V_{CCO} by its reference resistor. Pin VRP must be pulled down to ground by its reference resistor.
4. The value of the external reference resistors should be selected to give the desired output impedance. If you are using GTL_DCI, HSTL_DCI, or SSTL_DCI I/O standards, then they should be 50 Ω.
5. The values of the reference resistors must be within the supported range. Availability of this range is planned for the next release of the Virtex-II Data Sheet. (~30 to 100 W)
6. Follow the DCI I/O banking rules.

The DCI I/O banking rules are the following:

1. V_{REF} must be compatible for all of the inputs in the same bank.
2. V_{CCO} must be compatible for all of the inputs and outputs in the same bank.
3. No more than one DCI I/O standard using Single Termination type is allowed per bank.
4. No more than one DCI I/O standard using Split Termination type is allowed per bank.
5. Single Termination and Split Termination, Controlled Impedance Driver, and Controlled Impedance Driver with Half Impedance can co-exist in the same bank.

In this reference design, SSTL2_I_DCI and SSTL2_II_DCI standards are used. The input and output standards are specified in the "user constraint file" available with the reference design.

Design / Implementation

The following design techniques for controller design are used for operation at the desired speed.

- DDR registers are used for Double Data rate throughput
- 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 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
- "Logic accessible clock" is used to route to non-clock CLB pins

Place and Route

A constraint file (top.ucf) is included in the reference design. The constraint file contains the following:

- A clock period
- The first lac_clk path
- 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
- FAST attribute on all the outputs for better performance
- IOSTANDARD attribute for all I/Os

After running place and route, check the trace report for any timing violations.

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.
- Xilinx strongly recommends using a board design tool to analyze the traces and check for signal integrity.

Reference Design and Results

The Verilog reference design can be easily modified for a different system design requirement (**xapp253.zip**).

The following results where obtained using a Virtex-II X2v1000-5FG256.

Device Utilization:

Number of External IOBs	137 out of 172	79%
Number of SLICEs	159 out of 5120	3%
Number of BUFGMUXs	4 out of 16	25%
Number of DCMs	2 out of 8	25%

Performance: This reference design is implemented at 133 MHz

Conclusion

This reference design shows how to interface a Virtex-II series FPGA to a DDR SDRAM. The Xilinx DCM and flexible SelectI/O features eliminate the need for special clock generators, I/O drivers, and external termination on the board.

The Double Data Rate registers make Virtex-II devices an ideal choice for interfacing with a DDR SDRAM. This reference design is a 16-bit version of the controller. A 64-bit version can be implemented with a few changes in the control logic.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/12/01	1.0	Initial Xilinx release.