XAPP702 (v1.8) April 23, 2007

DDR2 Controller Using Virtex-4 Devices

Author: Tze Yi Yeoh

Summary

DDR2 SDRAM devices offer new features that surpass the DDR SDRAM specifications and enable a DDR2 device to operate at data rates of 400 Mb/s and above. High data rates demand higher performances from the controller and the I/Os in an FPGA. The controller must operate synchronously with the operating speed of the memory to achieve the desired bandwidth.

Introduction

This application note describes a DDR2 controller implementation in a Virtex™-4 device interfacing to a Micron DDR2 SDRAM device. This document provides a brief overview of the DDR2 SDRAM device features and a detailed explanation of the controller operation when interfacing to high-speed DDR2 memories. It also explains the backend user interface to the controller. The reference design for the DDR2 SDRAM controller is provided with the Memory Interface Generator (MIG) tool that is integrated with the Xilinx Core Generator tool.

This application note describes the DDR2 controller operations for the direct clocking physical layer. Two types of physical layer designs are available for the Virtex-4 family – direct clocking and SERDES. The direct clocking physical layer captures data directly with the FPGA clock (not using the data strobe) and is described in detail in XAPP701, *DDR2 SDRAM Physical Layer Using Direct-Clocking Technique*. The SERDES physical layer uses the data strobe to capture data and is described in XAPP721, *High-Performance DDR2 SDRAM Interface Data Capture Using ISERDES and OSERDES*. The reference design for the DDR2 SDRAM controller is provided with the Memory Interface Generator (MIG) tool that is integrated with the Xilinx Core Generator tool.

DDR2 SDRAM Device Overview

The DDR2 SDRAM device uses the SSTL 1.8V I/O standard and uses a DDR architecture to achieve high-speed operation. The memory operates using a differential clock provided by the controller. Commands are registered at every positive edge of the clock. A bidirectional data strobe (DQS) is transmitted along with the data for use in data capture at the receiver. DQS is transmitted by the DDR2 SDRAM device during reads, and the controller transmits DQS during writes. DQS is edge-aligned with data for reads and is center-aligned with data for writes.

Read and write accesses to the DDR2 SDRAM device are burst oriented. Accesses begin with the registration of an active command and are then followed by a Read or Write command. The address bits registered with the active command are used to select the bank and row to be accessed. The address bits registered with the Read or Write command are used to select the bank and the starting column location for the burst access.

The DDR2 controller design includes a user backend interface that generates the write address, write data, and the read addresses. This information is stored in four asynchronous FIFOs for address and data synchronization between the backend and controller modules. Based on the availability of data in the FIFOs and the commands issued by the command logic block, the controller issues the correct commands to the memory, taking into account the memory timing requirements.

DDR2 Commands Issued by the Controller

Table 1 lists the commands issued by the controller. The commands are detected by the memory using the following control signals:

- Row Address Select ($\overline{RAS}$)
- Column Address Select ($\overline{CAS}$)
- Write Enable ($\overline{WE}$) signals
- Clock Enable (CKE) held High throughout and after device configuration
- Chip Select ($\overline{CS}$) held Low throughout device operation

Table 1: **DDR2 Commands**

Selection	Function	Row Address Select	Column Address Select	Write Enable Signals
1	Load Mode	L	L	L
2	Auto-Refresh	L	L	H
3	Precharge(1)	L	H	L
4	Bank Activate	L	H	H
5	Write	H	L	L
6	Read	H	L	H
7	No Operation/IDLE	H	H	H

Notes:

1. Address signal A10 is held High during PRECHARGE ALL BANKs and is held Low during single bank precharge.

Mode Register

The Mode register (MR) is used to define the specific mode of operation of the DDR2 SDRAM. This includes the selection of burst length, burst type, CAS latency, and operating mode. Figure 1 shows the features of the mode register used by the controller.

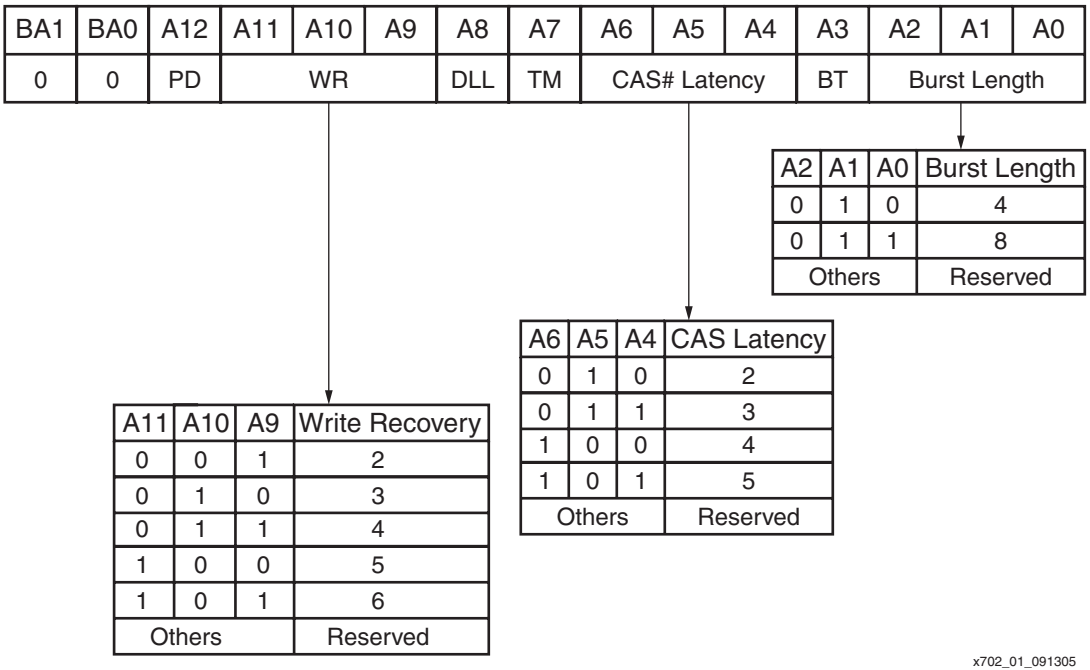

Figure 1: **Mode Register**

Bank addresses BA1 and BA0 select the Mode registers. Table 2 lists the configuration of the bank address bits.

Table 2: **Bank Address Bit Configuration**

BA1	BA0	Mode Register
0	0	MR
0	1	EMR1
1	0	EMR2
1	1	EMR3

Extended Mode Register

The Extended Mode register (EMR) controls functions beyond those controlled by the MR. As listed in Table 3, these additional functions are DLL enable/disable, output drive strength, on-die termination (ODT), posted CAS additive latency (AL), off-chip driver impedance calibration (OCD), $\overline{\text{DQS}}$ enable/disable, RDQS/$\overline{\text{RDQS}}$ enable/disable, and output disable/enable.

Table 3: **Extended Mode Register**

BA1	BA0	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	1	Out	RDQS	$\overline{\text{DQS}}$	OCD program			R_{TT}	Posted CAS			R_{TT}	ODS	DLL

The DDR2 SDRAM reference design uses an AL of 0. The reference design does not use OCD.

Extended Mode Register 2 (EMR2)

Bank addresses are set to `10`, where BA1 is High and BA0 is Low. All address bits are set Low.

Extended Mode Register 3 (EMR3)

Bank address bits are set to `11`, where BA1 and BA0 are High. All address bits are set Low.

Initialization Sequence

The initialization sequence used in the controller state machine follows the DDR2 SDRAM specifications. The memories' voltage requirements must be met by the interface. For a minimum of 200 µs after stable power and valid clock (CK, CK#), NOP or DESELECT commands are applied and then clock enable (CKE) is asserted HIGH. Following is the sequence of commands issued for initialization:

1. A minimum of 400 ns after CKE is asserted HIGH, a PRECHARGE ALL command is issued.
2. An EMR(2) command is issued next by providing a LOW on BA0, a LOW on BA2, and a HIGH on BA1.
3. An EMR(3) command is issued by providing a HIGH on BA0, a HIGH on BA1, and a LOW on BA2.
4. An EMR command is issued to enable the memory DLL by providing a LOW on A0, BA1, and BA2. A HIGH is provided on BA0.
5. A Mode register set (MRS) command is issued to reset the memory DLL by providing a HIGH on A8 and LOW on BA2, BA1, and BA0.
6. A Precharge ALL command is issued.
7. Two auto-refresh commands are issued.
8. Finally, an MRS command is issued by providing a LOW on A8. This command is required to initialize the device with operating parameters (without DLL reset).

9. At least 200 clocks after step 8, OCD Calibration (Off Chip Driver impedance adjustment) is executed. If OCD calibration is not used, then an EMR with default OCD command (A9=A8=A7=1) is issued, followed by an EMR with OCD.

The DDR2 SDRAM device is now ready for calibration.

The calibration procedure used in the direct clocking physical layer is done on a per-bit basis. Every DQ channel is routed in the FPGA through a primitive called an IDELAY which comprises a selectable tap delay line. This tap delay line allows the user to incrementally add or subtract a discrete delay amount (78 ps) to the DQ line. This process allows the user to optimally position the DQ line such that it is centered with respect to the FPGA clock. The algorithm and circuit diagram of the calibration procedure is given in XAPP701.

After the DQ calibration, the controller issues a pattern write command in order to write a known pattern to the memory device. This is followed by a read command to the same location that the know pattern was written to. The data being read back helps determine the delay in clock cycles between the read command and the received data. This count is registered and is used for read enable generation during normal read operations.

This read enable calibration is implemented in the data_write and rd_data modules. During pattern write, the controller issues a write command to the memory, and known write data pattern is generated in the data_write module. During pattern read, the controller issues a read command to the same location and generates a ctrl_rden signal that is delayed to align with the received read data from the memory. This delay logic is implemented in the pattern_compare8 module. The delayed ctrl_rden is used during normal read operation. The pattern_compare8 module asserts the COMP_DONE signal, indicating to the controller the completion of read enable calibration. The controller module is now ready for normal read/write operation.

The Memory Interface Generator (MIG) tool (version 1.7) generates one read_enable logic (pattern_compare8 module) per bank. If the bank has four data bits, the MIG tool generates a pattern_compare4 module.

Precharge Command

The precharge command is used to deactivate the open row in a particular bank. The bank is available for a subsequent row activation for a specified time (t_{RP}) after the precharge command is issued. Input A10 determines whether one or all banks are to be precharged.

Auto-Refresh Command

DDR2 devices are required to be refreshed once every 7.8 µs. The circuit to request auto-refresh counters is built inside the controller. The controller uses the DCM CLKDV output to provide the low frequency clock required for the auto-refresh counter. To save on the BUFG that is used by the CLKDV output of the DCM, designers can use the high frequency CLK0 output of the DCM or the CLK/4 output of the DCM (used by the IDELAY Circuit) to clock the refresh counter. If the clock to the auto-refresh circuitry is changed, the max_ref_count in the mem_interface_top_parameters_0.v file needs to be changed accordingly.

The auto_ref signal flags the need for a pending auto-refresh command. This signal is held High until the controller issues an auto-refresh command. The controller completes the transactions in a current open bank before issuing the auto-refresh command.

Active Command

Before any Read or Write commands can be issued to a bank within the DDR2 SDRAM, a row in the bank needs to be activated using an active command. After a row has been opened, Read or Write commands can be issued to the row according to the t_{RCD} specification. DDR2 SDRAM devices also support posted CAS AL. These devices allow a Read or a Write command to be issued prior to the t_{RCD} specification by delaying the actual registration of the Read or Write command to the internal device by AL clock cycles. The DDR2 controller supports the CAS ALs while issuing Read or Write commands following an active command.

When the controller detects an incoming address referring to a row in a bank other than the currently opened row, the controller issues an address conflict signal. The controller then issues a precharge command to deactivate the open row. It also issues another active command to the new row.

Read Command

The Read command is used to initiate a burst read access to an active row. The value on BA0 and BA1 selects the bank address, and the address inputs provided on $A_0 – A_1$ select the starting column location. After the read burst is over, the row is still available for subsequent access until it is precharged.

Figure 2 shows the case of a read command with an AL of 0. Hence, the read latency in this case is the same as the CAS latency (CL = 3).

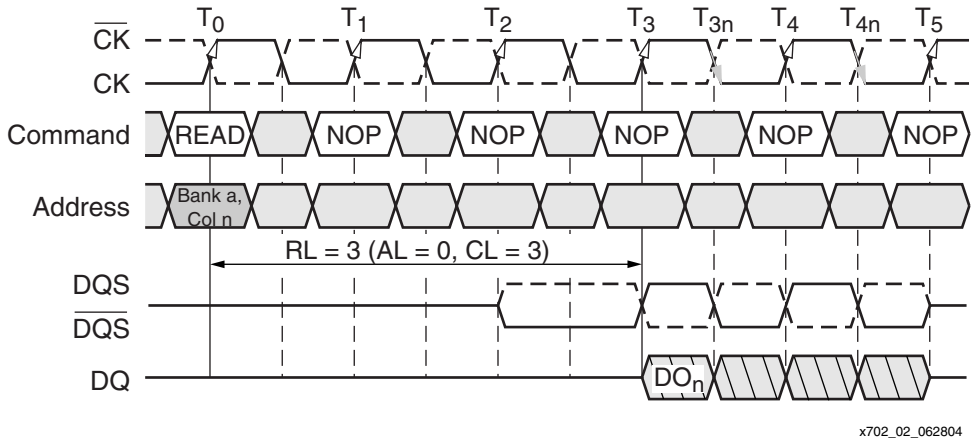

Figure 2: **Read Command Example**

Write Command

The write command is used to initiate a burst access to an active row. The value on BA0 and BA1 selects the bank address, while the value on address inputs $A_0 – A_1$ selects the starting column location in the active row. DDR2 SDRAMs use a write latency (WL) equal to the read latency minus one clock cycle.

Write Latency = Read Latency – 1 = (Additive Latency + CAS Latency) – 1.

Figure 3 shows the case of a write burst with a write latency of two. The time between the write command and the first rising edge of the DQS signal is determined by the write latency.

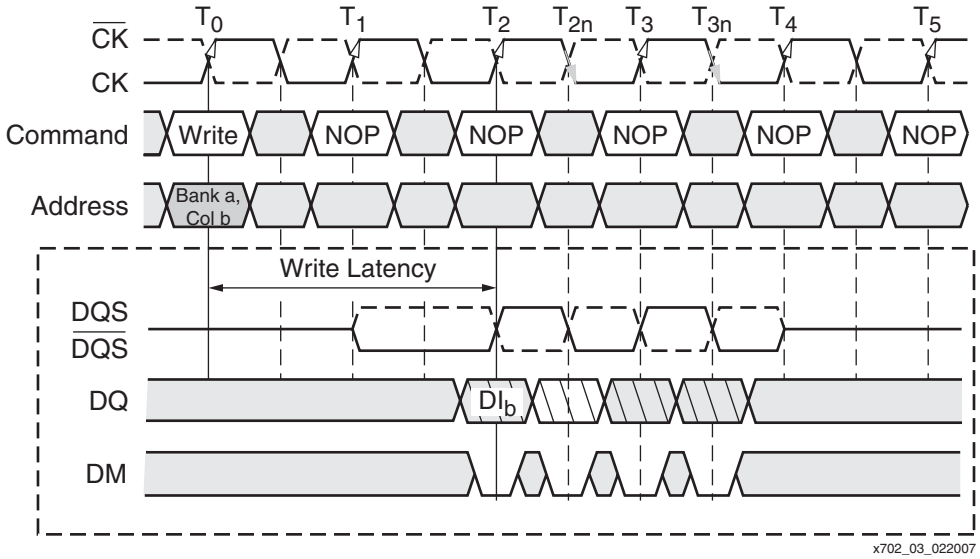

Figure 3: **Write Command Example**

DDR2 Interface Implementation

The DDR2 SDRAM interface includes the blocks shown in Figure 4.

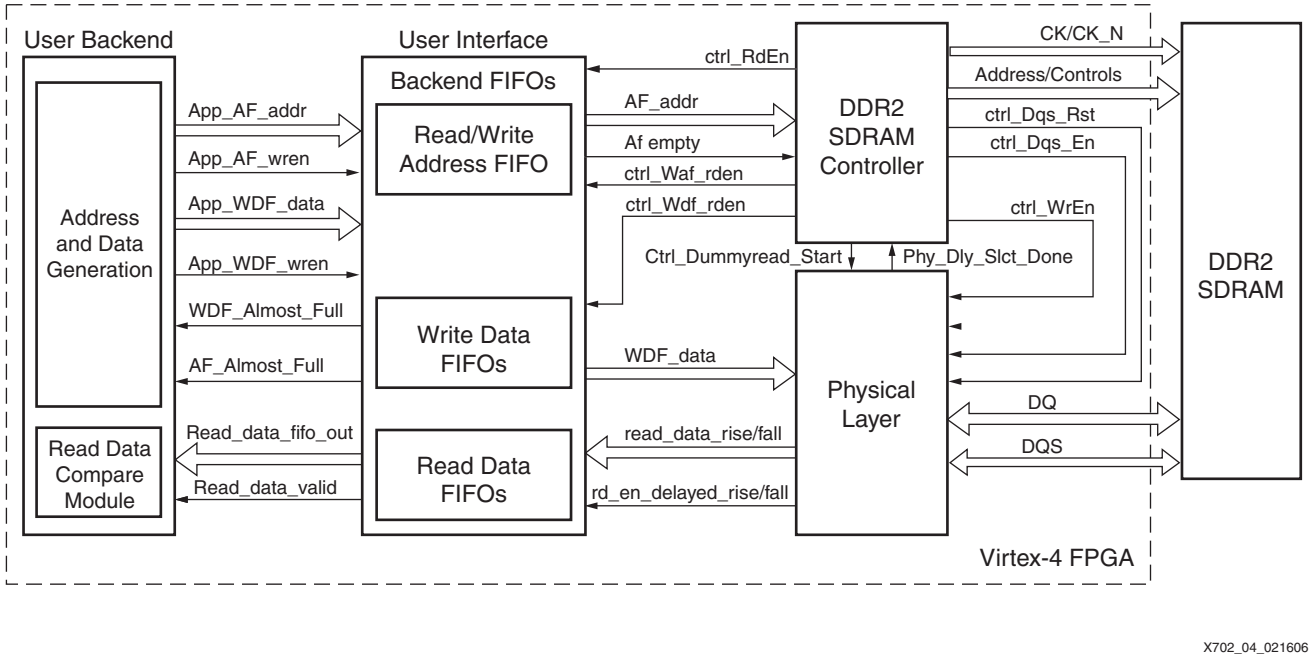

Figure 4: **DDR2 SDRAM Interface Block Diagram**

User Backend

The backend of the reference design provides address and data patterns to test all the design aspects of a DDR2 controller. The user backend includes the following blocks: backend state machine, read data comparator, and an address and data generator module.

The address and data generation module generates the various address and data patterns written to the memory. The address values and dynamic command requests are pre-stored in a block RAM used as a ROM. The address and dynamic command values stored are selected to test accesses to different rows and banks in the DDR2 SDRAM device. The data pattern generator includes a state machine issuing patterns of rising edge data. Falling edge data is the inverse of rising edge data. The backend state machine emulates the user backend application by issuing read enable signals to address the FIFO.

User Interface

The backend user interface includes three FIFOs: the write-data FIFO, the read and write address FIFO, and the read data FIFO. The first two FIFOs are accessed by the user backend modules. The read data FIFO is accessed by the datapath module to store the captured read data.

User to Controller Interface

Table 4 lists the signals between the user interface and the controller.

Table 4: **List of Signals Between User Interface and Controller**

Port Name	Port Width	Port Description	Notes
Af_addr	36	Output of the Address FIFO in the user interface. Mapping of these address bits: • Memory Address (CS, Bank, Row, Column): [31:0] • Reserved: [35] • Dynamic Command Request: [34:32]	Monitor FIFO-full status flag to write address into the address FIFO
Af_empty	1	The user interface Address FIFO empty status flag output. The user application can write to the Address FIFO when this signal is asserted until the write data FIFO-full status flag is asserted.	FIFO16 Almost Empty Flag
ctrl_af_RdEn	1	Read Enable input to Address FIFO in the user interface.	This signal is asserted for one clock cycle when the controller state is Write, Read, Load Mode register, Precharge All, Auto-Refresh, or Active resulting from dynamic command requests. Figure 5 shows the timing waveform for burst length of 8 with four back-to-back Writes followed by four back-to-back Reads.
ctrl_wdf_RdEn	1	Read Enable input to Write Data FIFO in the user interface.	The controller asserts this signal two clock cycles after the first write state. This signal remains asserted for 2 clock cycles for a burst length of 4 and four clock cycles for a burst length of 8. Figure 5 shows the timing waveform. Sufficient data must be available in Write Data FIFO associated with a write address for the required burst length before issuing a write command. For example, for a 64-bit data bus and a burst length of 4, the user should input two 128-bit data words in the Write Data FIFO for every write address before issuing the write command.

Table 5 lists the memory address (Waf_addr), which includes the column address, row address, bank address, and chip-select width for deep memory interfaces.

Table 5: **Waf_addr Memory Address**

Address	Description
Column Address	[col_ap_width – 1:0]
Row Address	[col_ap_width + row_address – 1:col_ap_width]
Bank Address	[col_ap_width + row_address + bank_address – 1:col_ap_width + row_address]
Chip Select	[col_ap_width + row_address + bank_address + chip_address – 1:col_ap_width + row_address + bank_address]

The address space spanned by Af_addr is discontinuous. Specifically, bit Af_addr [10] in the user interface address bus is ignored by the controller logic. In the case in which the memory controller is interfacing to DDR2 devices with only nine column bits, Af_addr[9] is also ignored. The column address width parameter col_ap_width includes an auto-precharge bit (A10) and a column address parameter. The column address parameter indicates the number of column address bits of the selected memory component.

- For a 9-bit column address, col_ap_width is defined as 11. The lower-order nine bits carry the column address, bit A9 is not used, and bit A10 is tied Low during normal reads and writes. As a result, the Auto-Precharge function is not supported. The col_ap_width parameter is used internally for prepending the A10 bit during the Precharge command.
- For a 10-bit column address, col_ap_width is defined as 11.
- For an 11-bit column address, col_ap_width is defined as 12.

Dynamic Command Request

Table 6 lists commands not required for normal operation of the controller. The user has the option of requesting these commands if the commands are required by their application.

Table 6: **Optional Commands**

Command	Description
000	Load Mode Register
001	Auto-Refresh
010	Precharge All
011	Active
100	Write
101	Read

Figure 5 shows four consecutive Writes followed by four consecutive Reads with a burst length of 8.

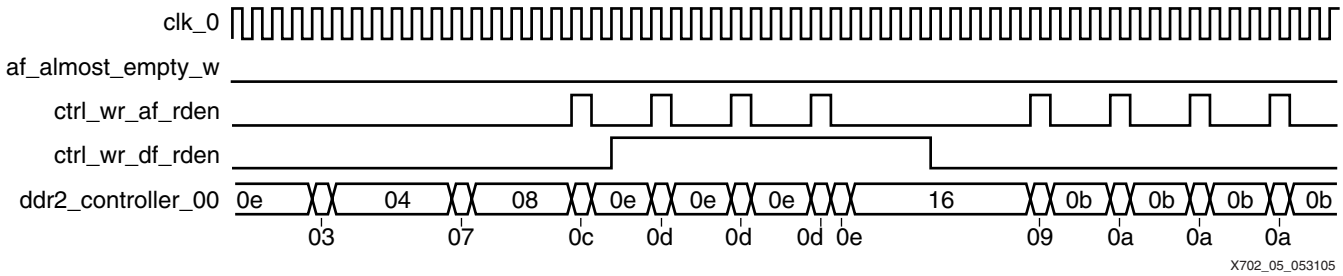

Figure 5: **Consecutive Writes Followed by Consecutive Reads with Burst Length 8**

Table 7 lists the state signal values for Figure 5.

Table 7: **State Signal Values for Figure 5**

State Signal Value	Description
0c	First write
0d	Burst write
09	First read
0a	Burst read
0e	Write wait
03	Precharge
04	Precharge wait
07	Active
08	Active wait
16	Write to read
0b	Read wait

Controller to Physical Layer Interface

Table 8 lists the signals between the controller and the physical layer.

Table 8: **Signals Between the Controller and Physical Layer**

Port Name	Port Width	Port Description	Notes
ctrl_Dummyread_Start	1	Output from the controller to the physical layer. When asserted, the physical layer begins strobe and data calibration after memory initialization.	This signal is asserted after read strobe begins to toggle in the dummy read state. This signal is deasserted when the phy_Dly_Slct_Done signal is asserted.
phy_Dly_Slct_Done	1	Output from the physical layer to the controller, indicating calibration is complete.	This signal is asserted after data bits have been delayed to center align with respect to the FPGA global clock. The ctrl_Dummyread_Start signal is de-asserted when phy_Dly_Slct_Done signal is asserted. Normal operation begins after this signal is asserted.
ctrl_Dqs_Rst	1	Output from the controller to the physical layer for the write strobe preamble.	This signal is asserted for one clock cycle during a write. The CAS Latency and AL values determine how many clock cycles after the first write state this signal is asserted. Figure 6 shows the timing waveform for this signal with CAS Latency of 3 and AL of 0 for four back-to-back writes with a burst length of 8.
ctrl_Dqs_En	1	Output from the controller to the physical layer for a write strobe.	This signal is asserted for three clock cycles during a write with a burst length of four and five clock cycles with a burst length of 8. The CAS Latency and AL values determine how many clock cycles after the first write or burst write state this signal is asserted. Figure 6 shows the timing waveform for this signal with CAS Latency of 3 and AL of 0 for four back-to-back writes with a burst length of 8.
ctrl_WrEn	1	Output from the controller to the physical layer for write data three-state control.	This signal is asserted for two clock cycles during a write with a burst length of 4 and for four clock cycles with a burst length of 8. The CAS Latency and AL values determine how many clock cycles after the first write or burst write state this signal is asserted. Figure 6 shows the timing waveform for this signal with CAS Latency of 3 and AL of 0 for four back-to-back writes with a burst length of 8.

Figure 6 shows the timing waveform for control signals from the controller to the physical layer.

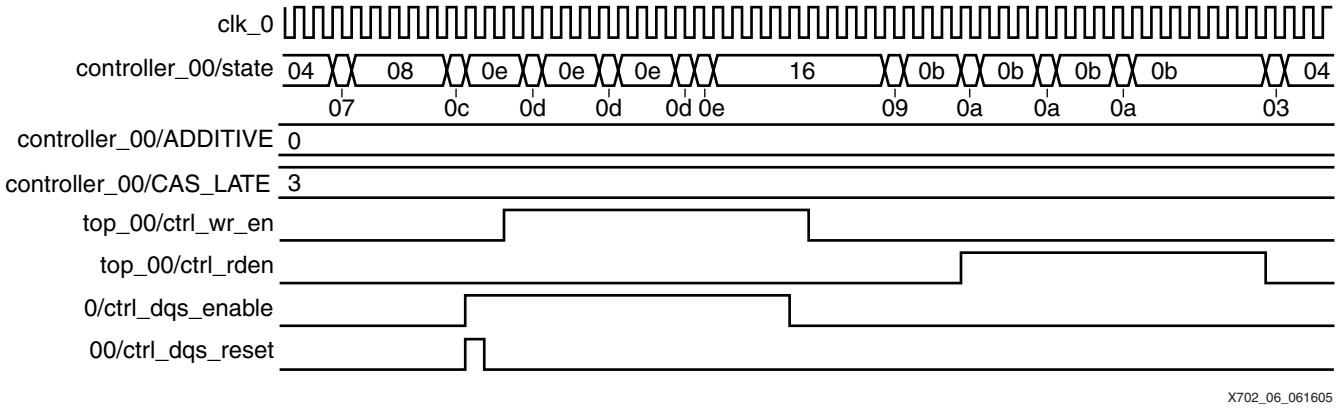

Figure 6: **Timing Waveform for Control Signals from the Controller to the Physical Layer**

Table 9 lists the state signal values for Figure 6.

Table 9: **State Signal Values for Figure 6**

State Signal Value	Description
0c	First write
0d	Burst write
09	First read
0a	Burst read
0e	Write wait
03	Precharge
04	Precharge wait
07	Active
08	Active wait
16	Write to read
0b	Read wait

Controller Implementation

The controller state machine issues the commands in the correct order while considering the timing requirements of the memory. Figure 7 shows an outline of the various stages in the controller state machine.

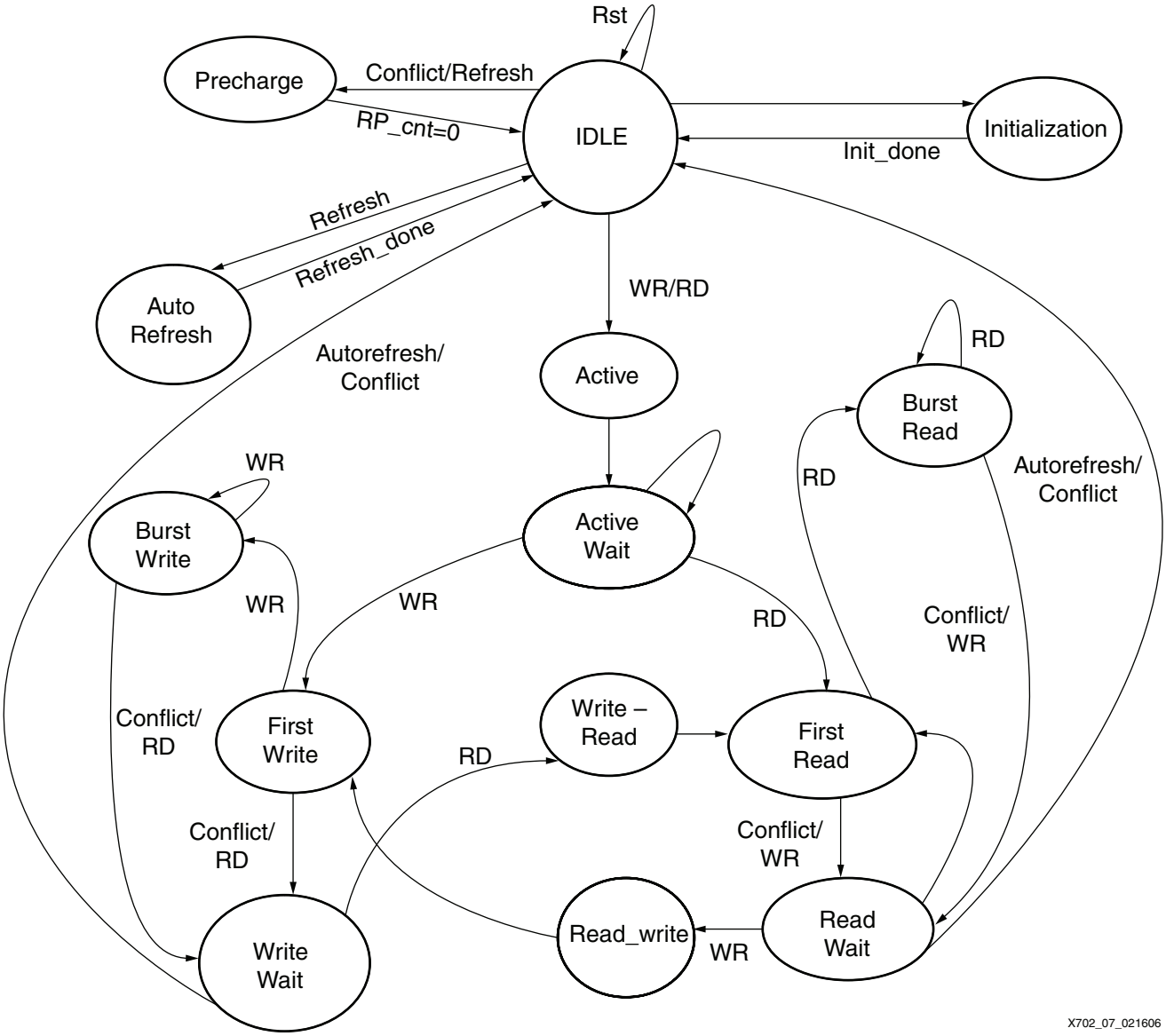

Figure 7: **DDR2 Controller State Machine**

Before the controller issues commands to the memory:

1. The controller issues a read-enable signal to the read/write address FIFO.
2. The controller activates a row in the corresponding bank if all banks have been precharged, or it compares the bank and row addresses to the already open row and bank address. If there is a conflict, the controller precharges the open bank and then issues an active command before moving to the read/write states.
3. In the write state, if the controller detects a Read command, the controller waits for the write_to_read time before issuing the Read command. Similarly, in the read state, when the controller detects a Write command from the command logic block, the controller waits for the read_to_write time before issuing the Write command.
4. A dynamic command request from the backend user application for a Precharge, Auto-Refresh, Active, or Load Mode register results in the controller issuing a Precharge command.
5. The commands are pipelined to synchronize with the address signals before being issued to the DDR2 memory.

Design Hierarchy

Figure 8 shows the design hierarchy beginning with a top-level module called ddr2.

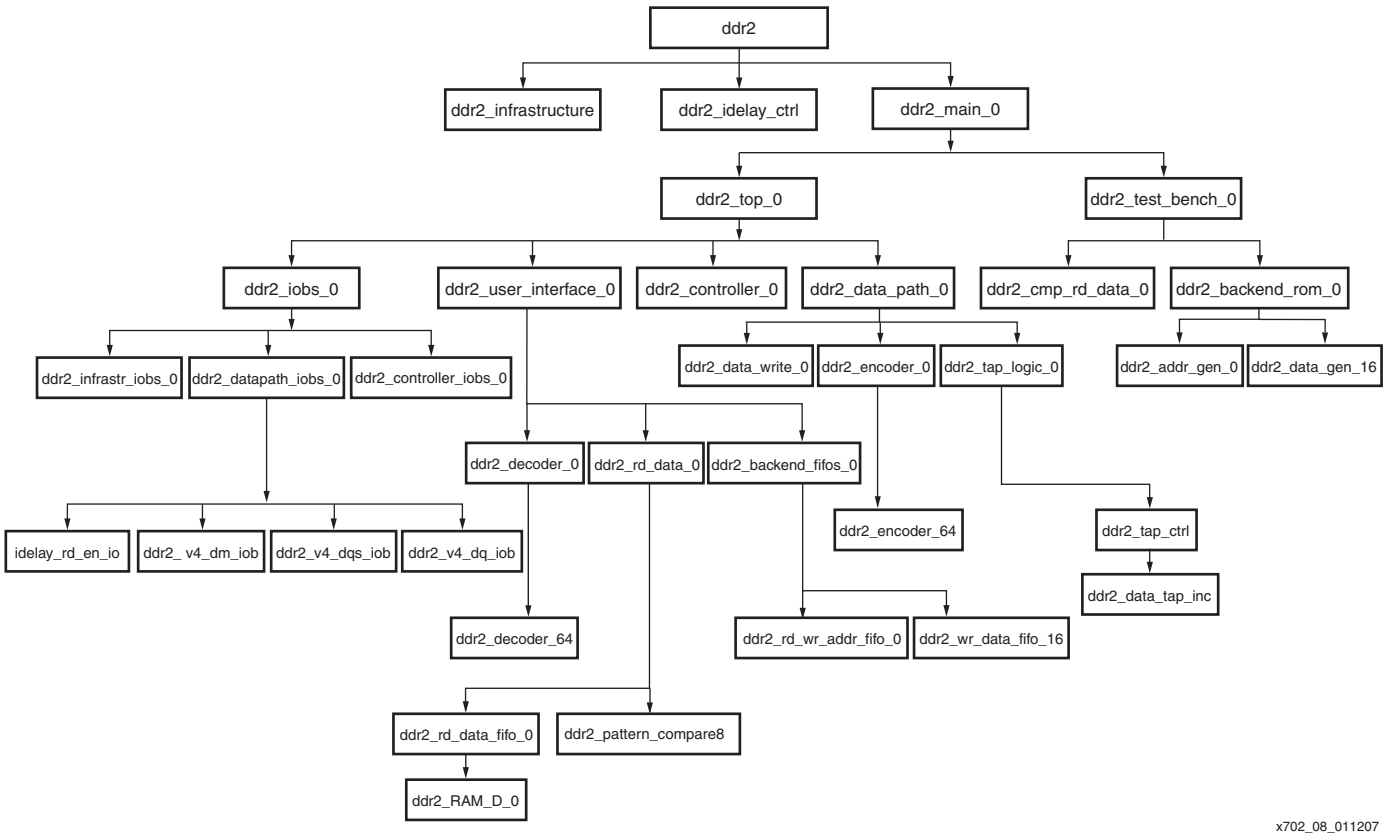

Figure 8: **Design Hierarchy**

Reference Design

The reference design for the DDR2 SDRAM memory controller using the Direct Clocking Data Capture Technique is integrated with the Memory Interface Generator (MIG) tool. This tool has been integrated with the Xilinx CORE Generator™ software. For the latest version of the design, download the IP Update on the Xilinx website at:
http://www.xilinx.com/xlnx/xil_sw_updates_home.jsp

Conclusion

In this reference design, the utilization of the Virtex-4 DCM, IOB, and differential clock tree enable the DDR2 controller to operate at the required speed.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/10/04	1.0	Initial Xilinx release.
06/13/05	1.1	Revised “DDR2 Interface Implementation” section.
07/15/05	1.2	Revised port names in Table 4, changed the text in “User to Controller Interface”, and revised Figure 4 and Figure 8.
07/29/05	1.3	Revised Figure 4 and updated reference design files.
09/14/05	1.4	Revised Figure 1 and updated the “Reference Design” section.
11/18/05	1.5	Revised “Auto-Refresh Command” section.
02/22/06	1.6	Updated Figure 4, Figure 7, and Figure 8. Also updated Table 4, Table 7, Table 8, and Table 9.
03/01/07	1.7	• Revised “Introduction” section. • Within the “Extended Mode Register” section, revised text within the “Initialization Sequence”. • Revised Figure 8. • Revised “Reference Design” section.
04/23/07	1.8	• Restored missing Figure 3, corrected to show proper timing relationships. • Corrected caption under Figure 5.