XAPP858 (v2.2) September 14, 2010

High-Performance DDR2 SDRAM Interface in Virtex-5 Devices

Authors: Karthi Palanisamy and Rich Chiu

Summary

This application note describes a 667 Mb/s DDR2 SDRAM interface implemented in a Virtex®-5 device. A customized version of this reference design can be generated using the Xilinx® Memory Interface Generator (MIG) tool.

Introduction

DDR2 SDRAM uses a source synchronous interface for transmission and reception of data. On a read, the data and strobe are transmitted edge aligned by the memory. To capture this transmitted data using Virtex-5 FPGAs, either the strobe and/or data can be delayed. In this design, the read data is captured in the delayed strobe domain and recaptured in the FPGA clock domain using a combination of the Input Double Data Rate (IDDR) and Configurable Logic Block (CLB) flip-flops.

DDR2 SDRAM Overview

DDR2 SDRAM devices are the next generation devices in the DDR SDRAM family. DDR2 SDRAM devices use the SSTL 1.8V I/O standard. The following section explains the features available in the DDR2 SDRAM devices and the key differences between DDR SDRAM and DDR2 SDRAM devices.

Table 1 provides a summary of the DDR2 reference design capabilities and features described in this application note.

Table 1: **DDR2 Reference Design Summary**

Parameters for Specification Details	Specification Details
Maximum Frequency, by speed grade (over ±3% voltage range)	**Speed Grade / Performance**
	-1 / 267 MHz
	-2 / 300 MHz
	-3 / 333 MHz
Device Utilization	**Design Details**
Internal FPGA Resource Utilization	See "Resource Utilization," page 45
Feature Support	**Configurations Supported**
Burst Lengths	4, 8
CAS Latency (CL)	3, 4, 5
Additive Latency	0, 1, 2, 3, 4
Bank Management	Up to 4 DDR2 banks open simultaneously
ECC	72-bit, 144-bit data bus widths
Component/Module	Component and DIMM (registered and unbuffered)
Other Features	On-Die Termination (ODT), 2T Address/Control Timing
HDL Language	Verilog, VHDL
Synthesis Tools	Xilinx Synthesis Technology (XST), Synplify Pro
Design Verification	**Configuration**
Device for Module Verification	Micron MT9HTF6472Y-667 DDR2 SDRAM RDIMM [Ref 1]
Device for Component Verification	Micron MT47H32M16CC-3 DDR2 SDRAM [Ref 2]

DDR2 SDRAM devices use 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 a strobe transmitted by the DDR2 SDRAM device during reads and by the controller during writes. DQS is edge aligned with data for reads and 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, which is 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 SDRAM controller reference design includes a user backend interface to generate the write address, write data, and read addresses. This information is stored in three backend FIFOs for address and data synchronization between the backend and controller modules. Based on the availability of addresses in the address FIFO, the controller issues the correct commands to the memory, taking into account the timing requirements of the memory.

DDR2 SDRAM Commands Issued by the Controller

Table 2 explains the commands issued by the controller. The commands are detected by the memory using these control signals: Row Address Select ($\overline{RAS}$), Column Address Select ($\overline{CAS}$), and Write Enable ($\overline{WE}$) signals. Clock Enable (CKE) is held High after device configuration, and Chip Select ($\overline{CS}$) is held Low throughout device operation. The DDR2 command functions supported in the controller are described in "Mode Register Definition," page 2.

Table 2: **DDR2 Commands**

Step	Function	$\overline{RAS}$	$\overline{CAS}$	$\overline{WE}$
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 Definition

The Mode register 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 Mode register features used by this controller. Bank Addresses BA1 and BA0 select the Mode registers.

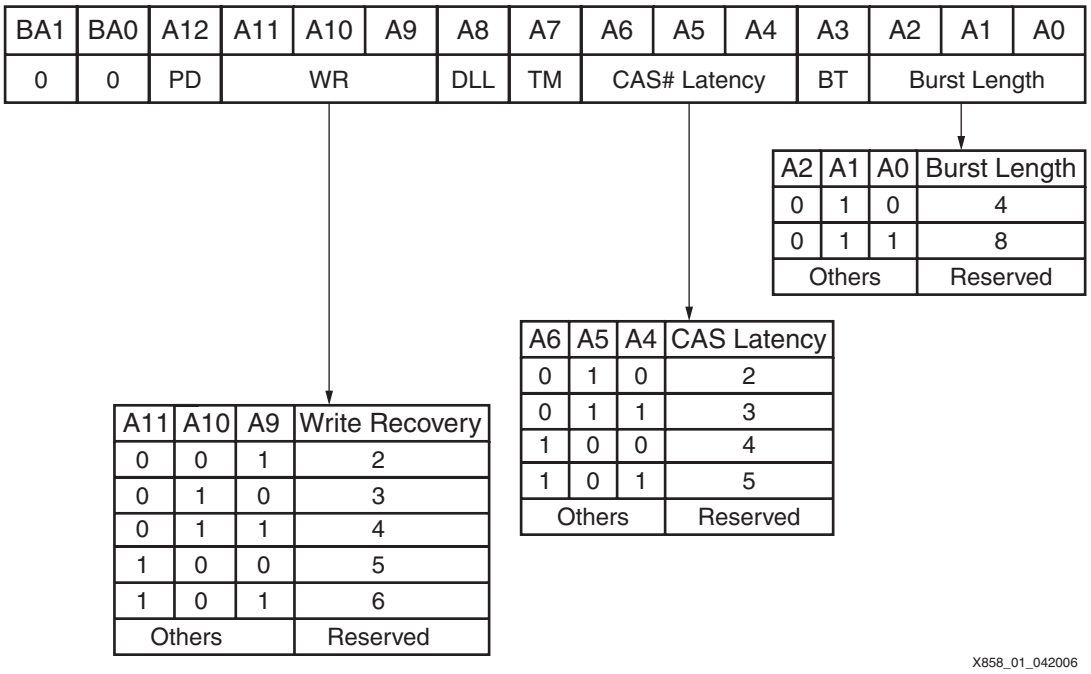


Figure 1: **Mode Register**

Table 3 shows the Bank Address bit configuration.

Table 3: **Bank Address Bit Configuration**

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

Extended Mode Register Definition

In addition to the functions controlled by the Mode register, the Extended Mode register (Table 4) controls these functions: DLL enable/disable; output drive strength (ODS); on-die termination; posted CAS additive latency (AL); off-chip driver impedance calibration (OCD); differential DQS enable/disable; RDQS enable/disable; and output disable/enable. OCD is not used in this reference design. The reference design uses a differential DQS signal, rather than a single-ended one. The on-die termination value (R_{TT}) is configurable in this reference design.

Table 4: **Extended Mode Register**

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

Extended Mode Register 2 (EMR2)

Bank Address bits are set to `10` (BA1 is set High, and BA0 is set Low). The address bits are all set to Low.

Extended Mode Register 3 (EMR3)

Bank Address bits are set to `11` (BA1 and BA0 are set High). Address bits are all set to Low, as in EMR2.

DDR2 Memory Commands

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 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 SDRAM devices need to be refreshed every 7.8 µs. The circuit to flag the Auto Refresh commands is built into the controller. The controller uses a system clock, divided by 16, to drive the refresh counter. When asserted, the auto_ref signal flags the need for Auto Refresh commands. The auto_ref signal is held High 7.8 µs after the previous Auto Refresh command. The controller then issues the Auto Refresh command after it has completed its current burst. Auto Refresh commands are given the highest priority in the design of this controller.

Active Command

Before any read or write commands can be issued to a bank within the DDR2 SDRAM memory, a row in the bank must be activated using an active command. After a row is opened, read or write commands can be issued to the row subject to the t_{RCD} specification. DDR2 SDRAM devices also support posted CAS additive latencies; these allow a read or 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 using additive latency clock cycles.

A conflict occurs when an incoming address refers to a row in a given bank where another row is currently open. In this case, the controller must first issue a Precharge command to deactivate the open row, and then issue an Active command to open the new row. The controller can optionally implement a bank management scheme to reduce overhead associated with opening and closing of banks and rows. This is described in "Bank Management," page 32.

Read Command

The Read command is used to initiate a burst read access to an active row. The values on BA0 and BA1 select the bank address. The address inputs provided on A_0 – A_i 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 an example of a read command with an additive latency of zero. Hence, in this example, the read latency is three, which is the same as the CAS latency.

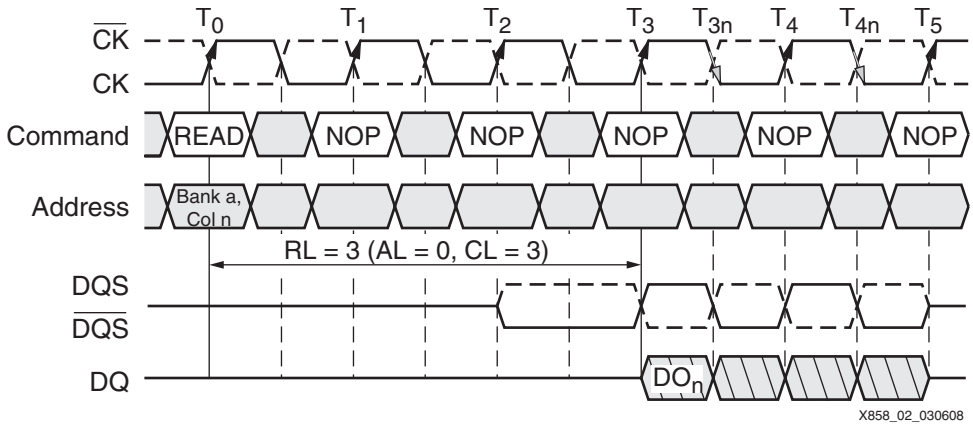


Figure 2: **Read Command Example**

Write Command

The Write command is used to initiate a burst access to an active row. The values on BA0 and BA1 select the bank address while the value on address inputs $A_0 – A_i$ select the starting column location in the active row. DDR2 SDRAMs use a Write Latency (WL) equal to Read Latency (RL) 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 WL of 2. The time between the Write command and the first rising edge of the DQS signal is determined by the WL.

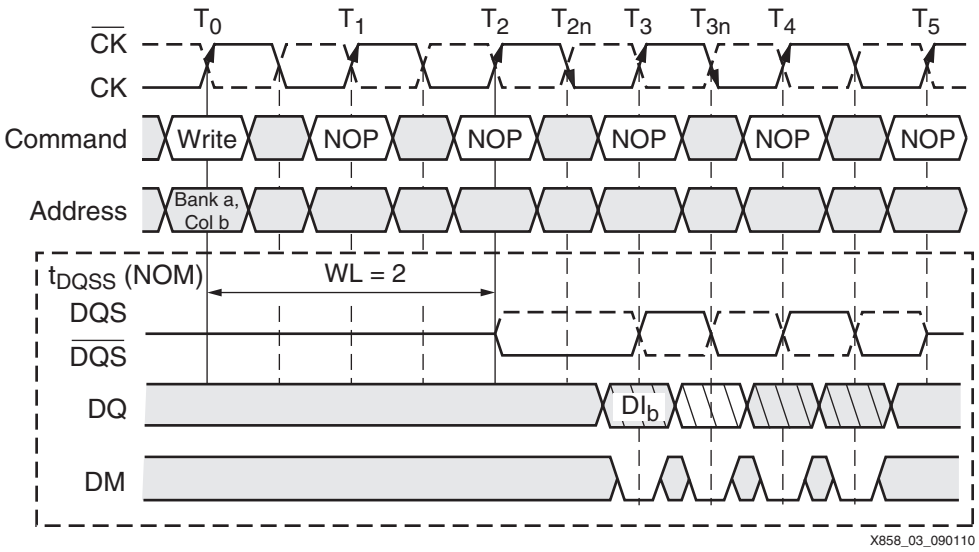


Figure 3: **Write Command Example**

Design Overview

The DDR2 SDRAM interface design block diagram is shown in Figure 4.

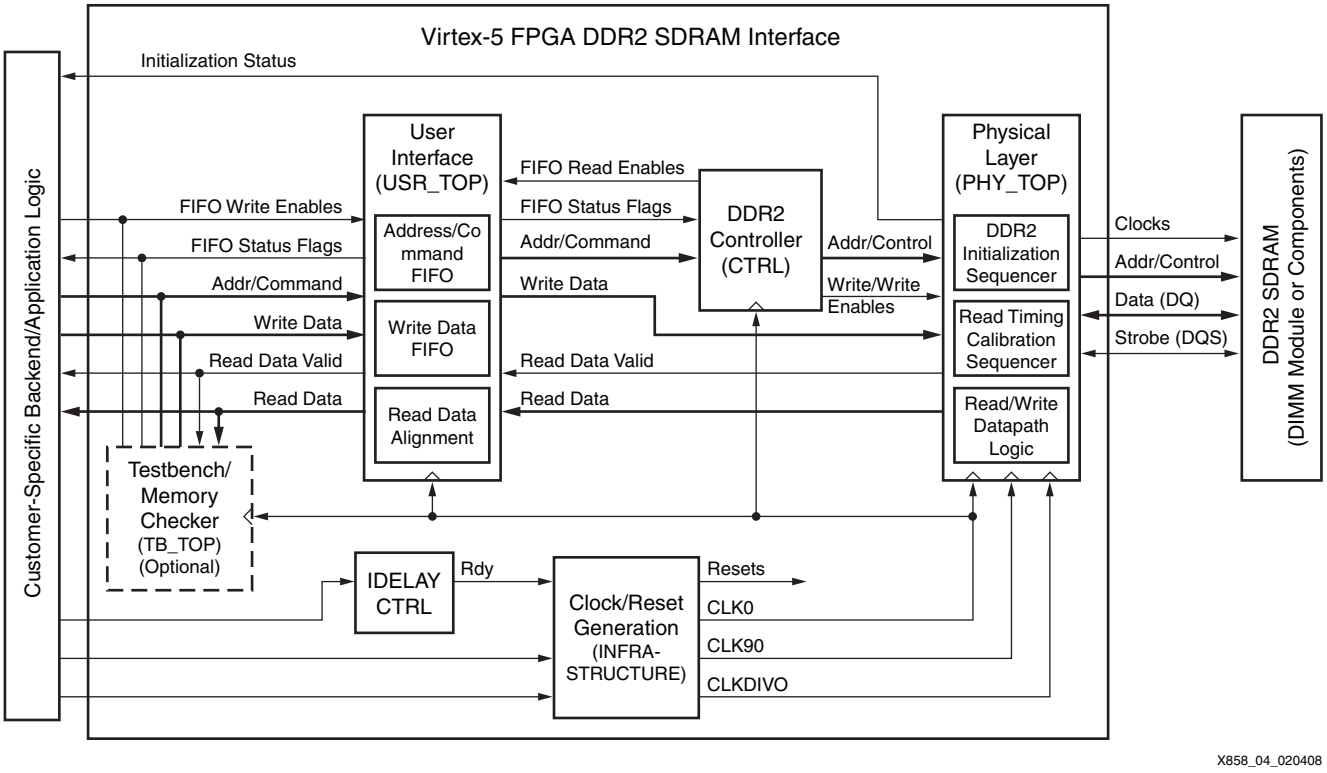


Figure 4: **DDR2 SDRAM Interface Block Diagram**

Table 5 provides a short description of each major block in the design.

Table 5: **DDR2 Memory Interface Design Major Block Descriptions**

Block	Description
Physical (PHY) Layer	The direct interface to the external DDR2 memory bus. Instantiates logic resources to generate the memory clock, control/address signals, and data/data strobes to/from the memory. Executes the DDR2 power-up and initialization sequence after system reset. Performs read data capture timing training calibration after system reset, and adjusts read data timing using ChipSync™ technology (IDELAY) elements.
Controller	Generates memory commands (e.g., Read, Write, Precharge, Refresh) based on commands from the User Interface block. Optionally, can implement a bank management scheme to reduce overhead with opening and closing of bank/rows. The controller logic takes over the DDR2 address/control bus after successful completion of DDR2 memory initialization and read timing calibration by the PHY layer.
User Interface	Provides a FIFO-like interface for the user-specific application to issue commands and write data to the DDR2 memory interface, and to receive read data from the DDR2 memory interface.
Clocking / Reset Logic	Generates clocks using Digital Clock Manager (DCM) module. Synchronizes resets to the various clock domains used in rest of design.
Synthesizable Testbench / Memory Checker	Optional module that can be used in place of actual user-specific backend application. Executes a repeating write/read test to check memory.

Clocking Scheme

Figure 5 shows the clocking scheme for this design. Global and local clock resources are used. The global clock resources consist of a digital clock manager (DCM) and several global clock buffers (BUFG). The local clock resources consist of regional I/O clock networks (BUFIO). The global clock architecture is discussed in this section.

The MIG tool allows the user to customize the design such that the DCM is not included. In this case, CLK0, CLK90, and CLKDIV0 must be supplied by the user.

The DCM can be replaced with a PLL if desired.

Global Clock Architecture

The user must supply two input clocks to the design:

- A system clock running at the target frequency for the memory. This clock is used by the DCM to generate the various clocks used by the memory interface logic.
- A 200 MHz clock for the IDELAYCTRL blocks, which in turn are used for the IDELAY IOB delay blocks for aligning read capture data.

The DCM generates three separate synchronous clocks for use in the design. This is shown in Table 6 and Figure 5.

Table 6: **DDR2 Interface Design Clocks**

Clock	Description	Logic Domain
CLK0	Skew-compensated replica of the input system clock	The clock for the controller and user interface logic, most of the DDR2 bus-related I/O flip-flops (e.g., memory clock, control/address, output DQS strobe, and DQ input capture)
CLK90	90° phase-shifted version of CLK0	Used in the write datapath section of physical layer. Clocks write path control logic, DDR2 side of the Write Data FIFO, and output flip-flops for DQ.
CLKDIV0	Divided-by-2 and edge-aligned version of CLK0	Clocks the memory initialization and read capture timing calibration state machines in the PHY layer.

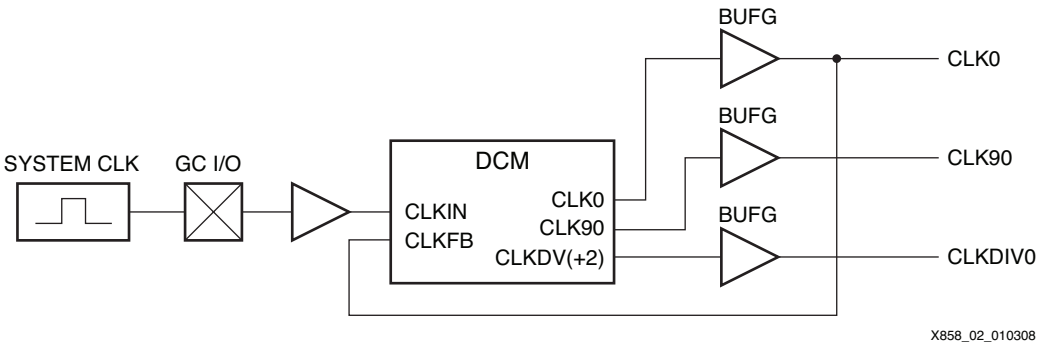


Figure 5: **Clocking Scheme for DDR2 Interface Logic**

Physical Layer

The physical layer (PHY) contains the clock/address/control generation logic, write datapath, read datapath, the state machine for read data capture timing calibration, and the memory initialization state machine.

After deassertion of system reset, the PHY logic performs the required power-on initialization sequence for the memory, followed by read data timing calibration. After calibration is complete, the PHY indicates that initialization is finished, and the controller can begin issuing commands to the memory. See Figure 6.

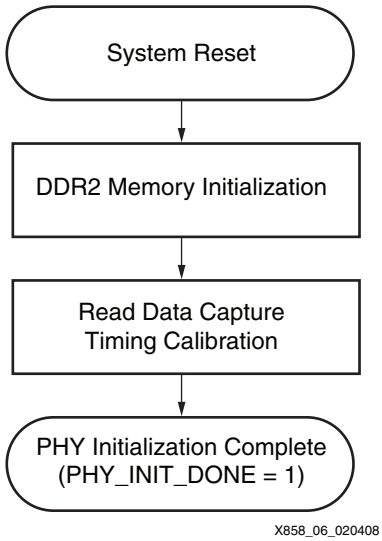


Figure 6: **PHY Overall Initialization Sequence**

Memory Initialization

The PHY layer executes a JEDEC-compliant initialization sequence for memory following deassertion of system reset. Certain bit values programmed to the DDR2 Mode register (MR) and Extended Mode register (EMRS), such as for the burst length, CAS latency, and additive latency, are configurable in the design and determined by the value of top-level HDL parameters. These parameters are discussed later in this document.

Figure 7 shows the memory initialization sequence executed for the PHY logic. Immediately after each load Mode Register, Precharge, or Auto Refresh command, a wait (idle) time of greater than 64 clock cycles is inserted before the next step. This is not shown in Figure 7.

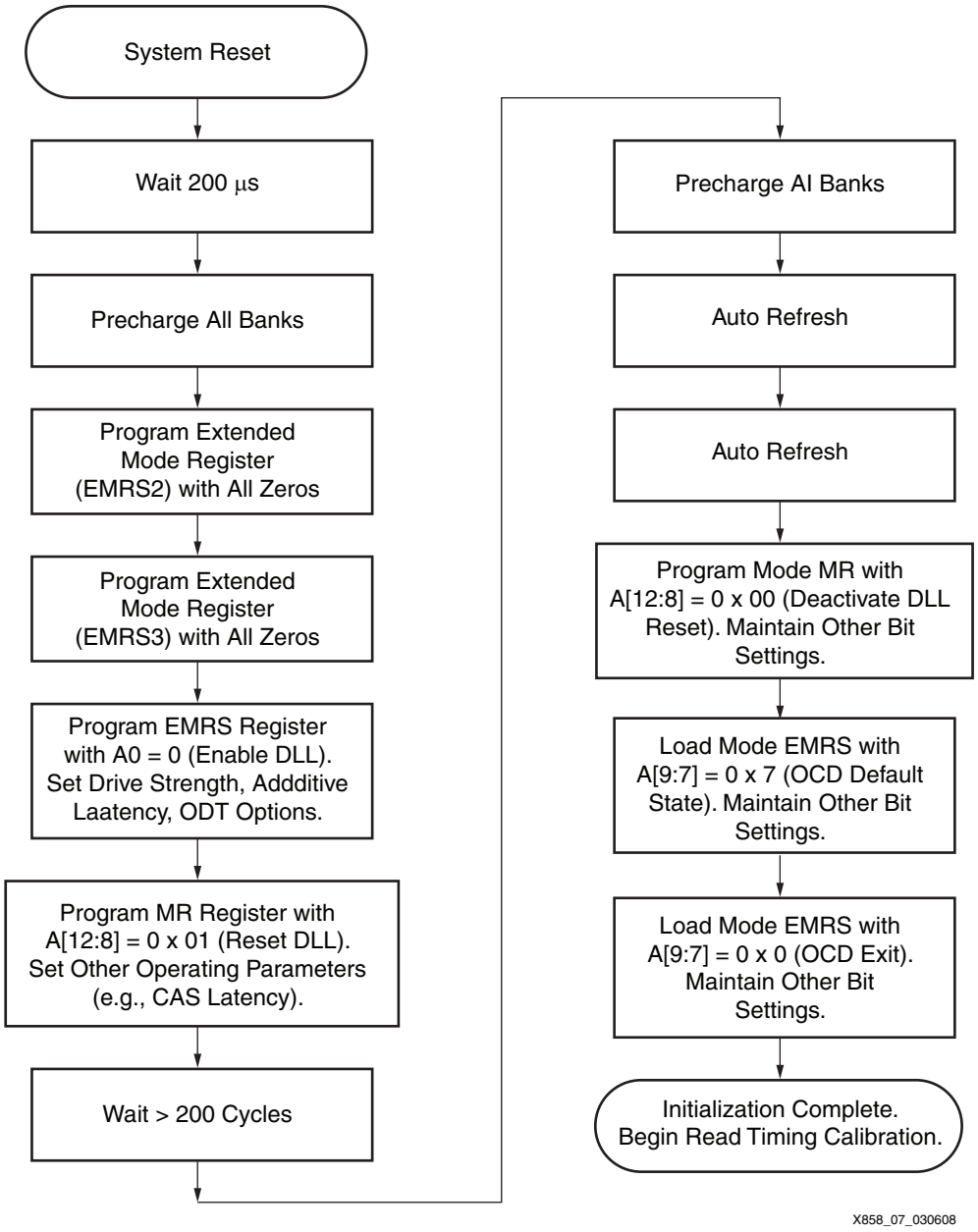


Figure 7: **Memory Initialization Sequence**

Clock and Address/Control Output Path

The DDR2 memory interface uses the internal system clock (CLK0) to generate a forwarded differential clock to the memory using the ODDR double-data rate output flip-flop. The ODDR is used to generate an inverted version of CLK0. CLK0 is also used to clock the ODDR flip-flops that drive the control (RAS_N, CAS_N, etc.) and address signals to the memory on a command. Because the forwarded clock to the memory is an inverted version of CLK0, the address and control signals arrive at the memory nominally 180° out of phase with the rising edge of the memory clock. Figure 8 shows the output timing for the forwarded clock and control/address signals to the memory.

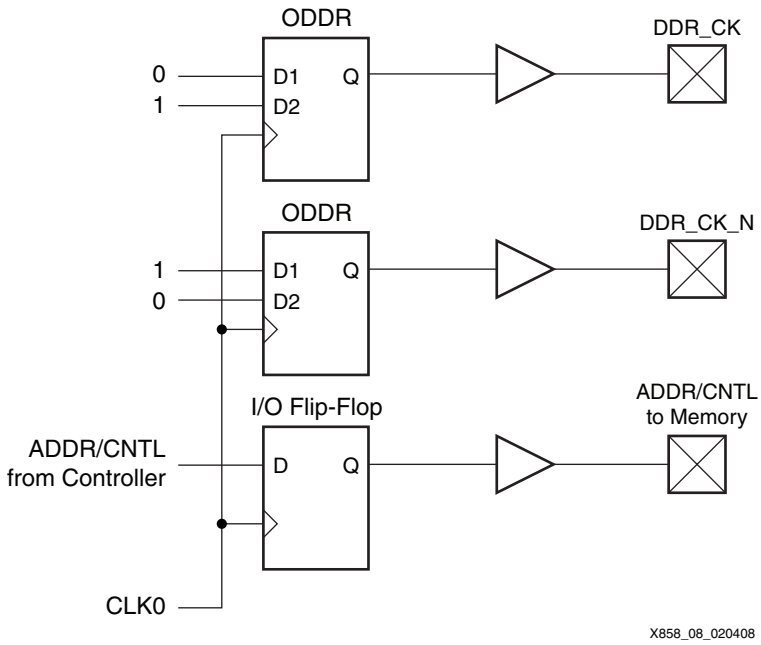


Figure 8: **Clock and Address/Control Path**

Write Datapath

The write datapath generates the data and strobe signals transmitted during a write command. The ODDR is used to transmit the data (DQ) and strobe (DQS) signals.

The strobe (DQS) forwarded to the memory is 180° out of phase with CLK0, and in-phase with the clock (CK, CK_N) forwarded to the memory. This is required to help meet the t_{DQSS} specification requirement for DDR2 memory (i.e., DQS and CK at the memory must be within ±0.25*(cycle time) of each other).

The write data transmitted using ODDR is clocked using CLK90 as shown in Figure 9. This center aligns DQ with respect to DQS on a write to meet the input data setup and hold time requirements of the DDR2 memory.

In addition to the ODDR used to register output data in each I/O, a second ODDR is in the IOB to control the 3-state enable for the I/O. This ODDR is used to enable the write data output one-half clock cycle before the first data word, and disable the write data one-half clock cycle after the last data word.

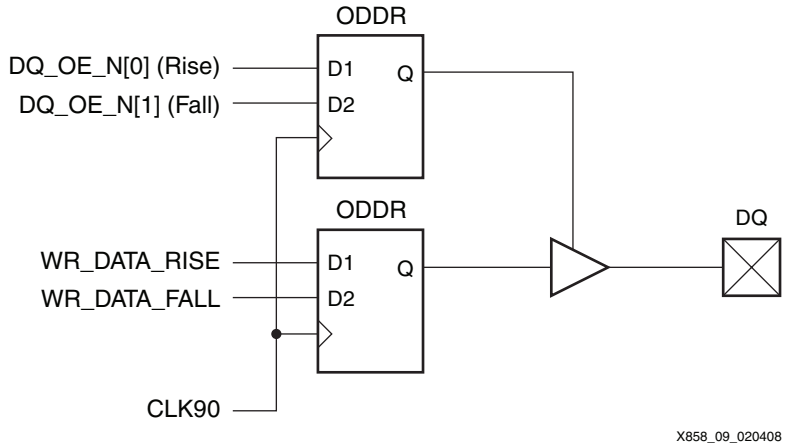


Figure 9: **Write Data Transmitted using ODDR**

The timing diagram for a write transaction is shown in Figure 10.

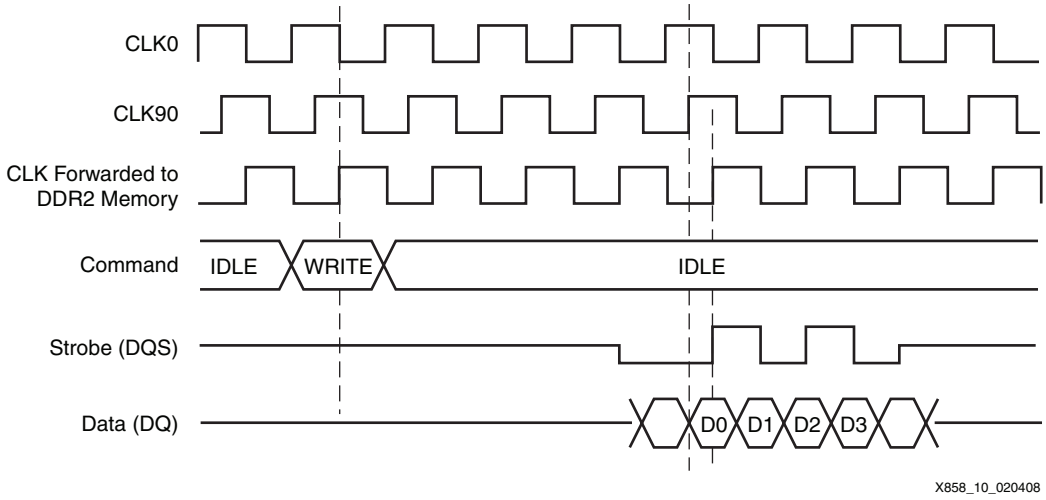


Figure 10: **Control/Address, Write Strobe (DQS) and Data (DQ) Timing for Write Latency = 4**

Write Timing Analysis

A sample write timing analysis for a DDR2 interface at 333 MHz for a -3 speed grade Virtex-5 device is shown in Table 7. Inter-symbol interference (ISI), crosstalk, user input clock jitter, and contributors to dynamic skew are not considered in this analysis.

Table 7: **Write Timing Analysis at 333 MHz**

Parameter	Value (ps)	Uncertainties before DQS	Uncertainties after DQS	Description
Clock Frequency		333.00		
T_{CLOCK}		3003.00		
$T_{DUTY\_CYC\_DLL}$		150.00	150.00	Duty cycle distortion from DCM
T_{BIT_PERIOD}		675.75	675.75	(Half clock period minus DCD) divided by 2

Table 7: **Write Timing Analysis at 333 MHz *(Cont'd)***

Parameter	Value (ps)	Uncertainties before DQS	Uncertainties after DQS	Description
T_{SETUP}	300.00	300.00	0.00	-3E DDR2 memory
T_{HOLD}	300.00	0.00	300.00	-3E DDR2 memory
$T_{PACKAGE\_SKEW}$		50.00	50.00	ESTIMATE. Design dependent.(1)
T_{JITTER}		0.00	0.00	Same DCM used to generate CLK0 and CLK90.
$T_{CLOCK\_TREE\_SKEW}$		100.00	100.00	Small value considered for skew on global clock line because DQS and associated DQ are placed close to each other. ESTIMATE. Design dependent.(2)
T_{OUT_OFFSET}		140.00	140.00	Phase alignment between different DCM outputs
$T_{PCB\_LAYOUT\_SKEW}$		50.00	50.00	PCB skew between DQS/DQ
Total Uncertainties		640.00	640.00	
Timing Margin		35.75	35.75	

Notes:

1. Appendix A of the *Memory Interface Generator (MIG) User Guide* [Ref 3] documents how to calculate package skew.
2. Clock skew between the DQ and DQS I/Os can be determined using the Timing Reporter and Circuit Evaluator (TRCE) tool.

Read Datapath

The read datapath comprises several flip-flop stages (*ranks*) over which read data from the DDR2 memory is transferred from the DQS strobe domain to the internal FPGA (CLK0) clock domain. It also includes the local I/O clock network for distributing the DQS for the initial data capture, and circuitry to ensure robust data capture on the last transfer of a read burst:

- Rank 1 Capture: Initial capture of DQ with DQS using the IOB IDDR flip-flop and IDELAY elements. The differential DQS pair must be placed on a clock-capable I/O pair and is distributed through the local BUFIO network to each of the corresponding DQ IDDR capture flip-flops. The IDDR generates two single-data-rate signals at its Q1 and Q2 outputs. Both Q1 and Q2 outputs are clocked by the falling edge of DQS because:
  - The IDDR is configured in SAME_EDGE mode
  - The DQS is inverted before clocking the IDDR
- Rank 2 Capture: The IDDR captures the data from the DDR2 memory, which is in the double-data-rate domain, and generates two single-data-rate streams at its Q1 and Q2 outputs.The outputs of the IDDR are transferred to flip-flops that are clocked by the rising or falling edge of CLK0. These flip-flops are located in the CLBs. There are two possible sets of flip-flops that each IDDR output can be transferred to. This allows synchronization of the IDDR outputs to either edge of CLK0 and reduces the maximum number of IDELAY taps required to perform timing alignment.

- Rank 3 Capture: The output of Rank 2 flip-flops clocked by the falling edge of CLK0 are transferred to flip-flops clocked by the rising edge of CLK0. In addition, the multiplexer for each DQ capture circuit is set to choose the appropriate synchronization path used.
- DQS Gate Circuit: This circuit disables the clock enable for each DQ IDDR at the end of a read burst to prevent any glitches associated with the DQS strobe being 3-stated by the memory from clocking the IDDR flip-flop. There is one such circuit per DQS group.

The read datapath is shown in Figure 11.

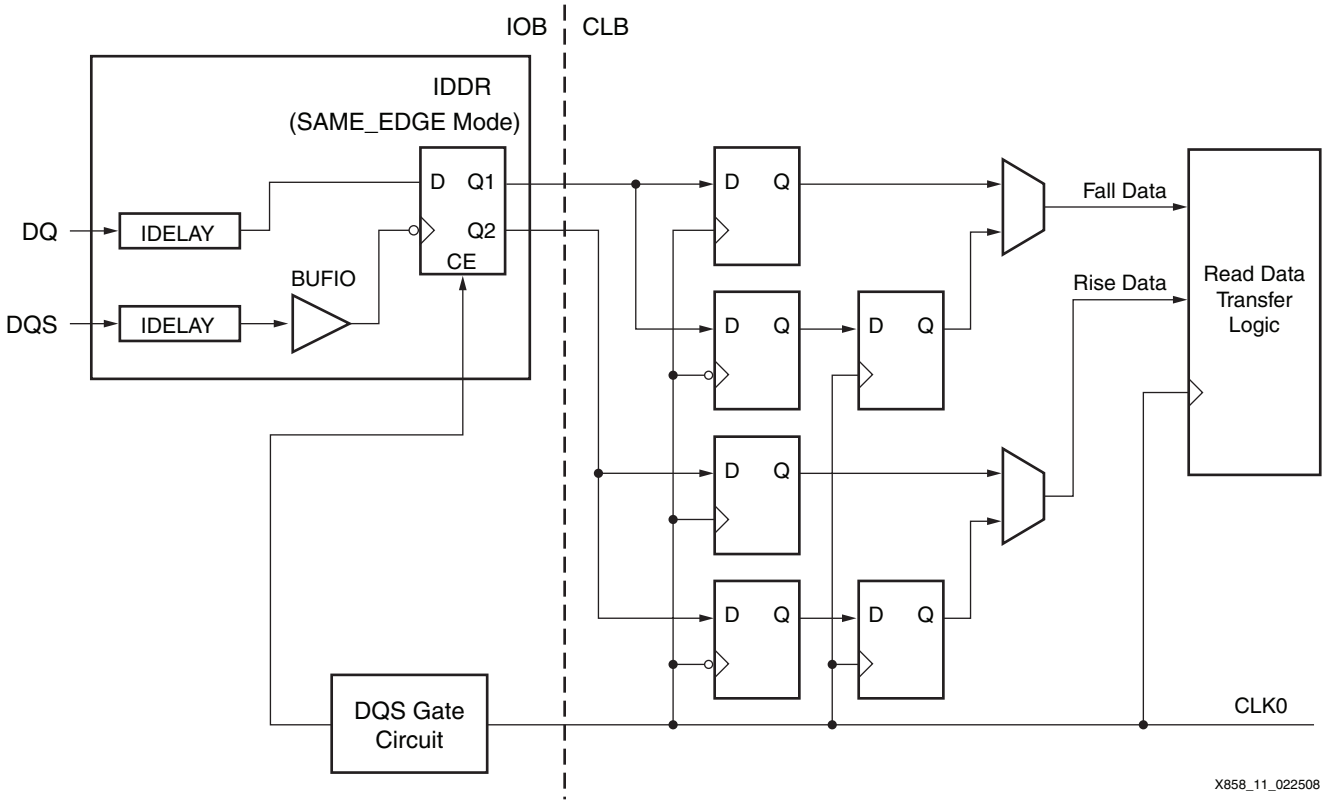


Figure 11: **Read Data Capture Using IDDR and CLB Flip-Flops**

Read Data Timing Calibration

To ensure reliable synchronization of read data from the memory to the FPGA core clock domain, a one-time calibration/timing training procedure is executed by the PHY layer logic after deassertion of system reset. This calibration procedure adjusts the timing of each of the synchronization stages using the IDELAY elements to factor out *static* timing uncertainties, such as PCB-related trace delays, process-related propagation delays on both the memory and FPGA.

The calibration algorithm varies the IDELAY for both DQ and DQS on a per-DQ (per-bit) basis. The IDELAY delays are incremented until the edges of the data valid window are found. The IDELAY values are then adjusted such that the sampling point occurs as close to the center of the data valid window as possible at the input to each of the synchronizing registers.

Read timing calibration is performed over four stages. Each stage consists of writing a specific training pattern to memory and then reading back the training pattern from memory. The calibration state machine logic is clocked using CLKDIV0 rather than CLK0 to ease the timing requirements on the design.

Figure 12 shows the overall calibration sequence. Each stage is discussed in succeeding sections.

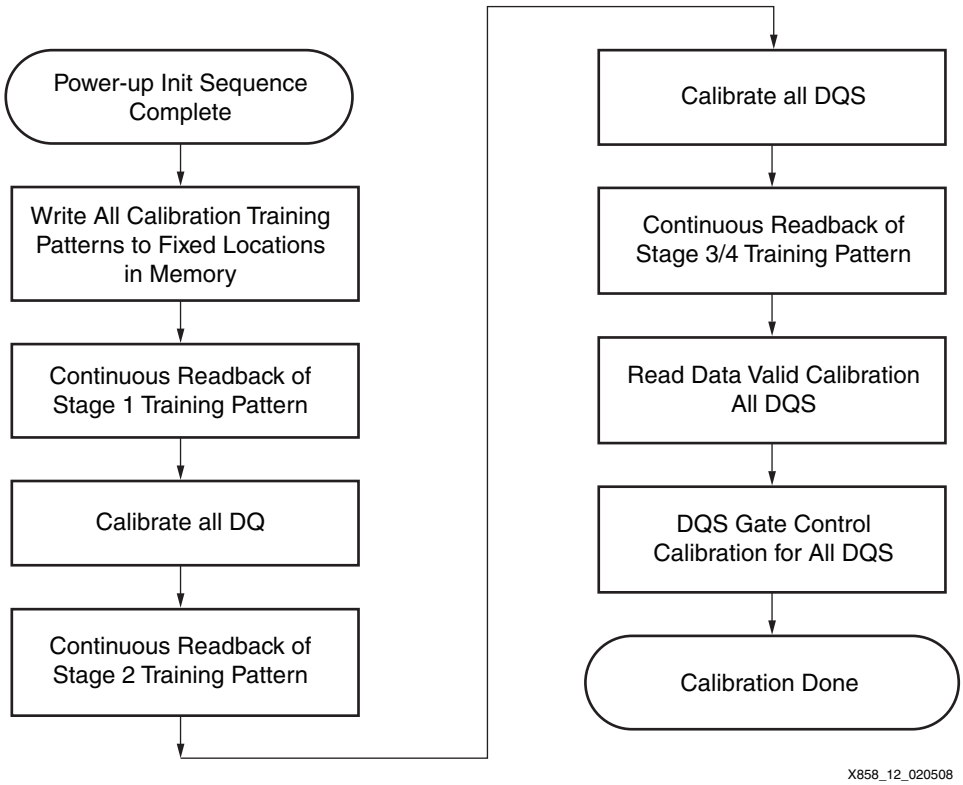


Figure 12: **Overall Calibration Sequence**

Stage 1 Read Timing Calibration: Per-bit DQS-DQ Alignment

The first stage of read timing calibration ensures reliable capture of DQ using DQS at the IDDR input flip-flop. This is required because DQ and DQS are roughly edge aligned coming out of the memory. After these signals enter the FPGA, each path is delayed by a different amount; for example, the DQS path is delayed more than the DQ path because it is routed through a BUFIO clock network.

This stage individually adjusts the IDELAY value for each DQ. IDELAY for DQS remains at 0 during this time. In this manner, static sources of skew can be accounted for. This includes the delay of the propagation delays within the FPGA as well as phase skews across the data bits in a DQS group.

This stage is performed before a phase relationship between DQS and CLK0 has been established. A training pattern of `10` (`1` = rising, `0` = falling) is used to calibrate DQ. When the DQS is aligned within the DQ data valid window at the IDDR, the IDDR returns a static value of `1` for its rising edge output, and `0` for its falling edge output. Therefore, data transfer between the first and second rank of flip-flops does not yet need to meet setup/hold requirements for the second rank (since the data is not toggling). If DQS is not in the DQ valid window, the rank 1 flip-flops return either `01` (`0` = rising, `1` = falling), `00`, or `11`. The last two possibilities can occur when the DQ transition occurs simultaneously with the DQS edge, and/or due to duty-cycle distortion of the DQ or DQS. This procedure is repeated for each DQ bit before the calibration state machine proceeds to Stage 2 calibration.

When DQS is aligned at the edge of the DQ window (by incrementing DQ IDELAY), the output of the first rank of flip-flops can toggle (e.g., from `10` to `11` or `00`), and metastability become a factor for the transfer between the IDDR (clocked by DQS) and the rank of CLB flip-flops

(clocked by CLK0). This possibility is addressed by registering the data using multiple ranks of flip-flops before reaching the calibration detection logic.

The calibration logic attempts to find two edges of the data valid window to calculate the bit time. At lower frequencies, it is possible that it only finds one edge. In this case, the logic assumes the bit period is equal to CLK_PERIOD/2. The calibration logic uses this information to determine the value to delay the DQ such that the sampling point of the IDDR takes place in the middle of the data valid window. At lower frequencies, it is possible that the sampling point cannot be located in the middle of the data valid window because of IDELAY tap and algorithmic limitations; however, there is sufficient timing margin because of the longer bit time.

Figure 13 gives an example of the change in IDDR input timing after stage 1 calibration.

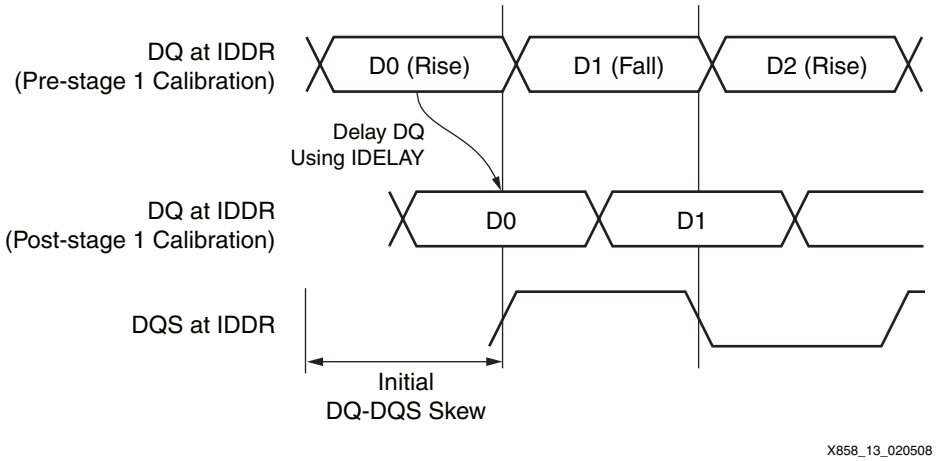


Figure 13: **First Stage Read Timing Adjustment**

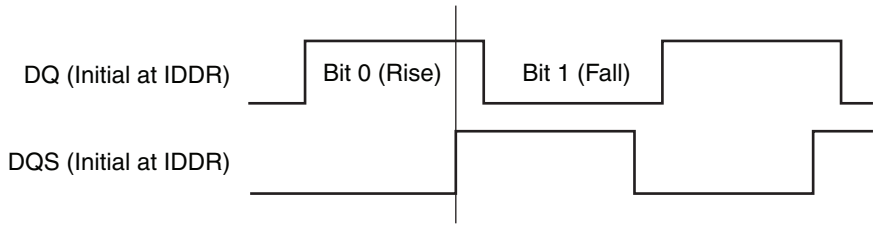
The IDELAY for DQS is never adjusted during this stage (it remains at 0). The path delay for DQS is significantly longer than that of DQ because DQS is routed through the local BUFIO clock network. At higher frequencies, it is necessary to add delay to the DQ path to compensate for the longer path formed by the DQS going through a BUFIO. At lower frequencies, the IDDR capture no longer occurs in the middle of the DQ valid window; instead, the inherent difference in delay between the DQ and DQS paths determines when sampling occurs. Because DQS IDELAY is not varied for this stage, at lower frequencies, stage 1 calibration does not result in the sampling point of the IDDR being positioned in the middle of the data valid window.

The calibration algorithm assumes that initially, DQS and DQ at each IDDR have one of two possible starting phase relationships. The phase depends on the memory frequency, the outgoing DQ-DQS skew from the DDR2 memory, PCB trace routing, and path delays internal to the FPGA.

- Case 1: DQS is in the corresponding DQ valid window, i.e., the rising edge of DQS is aligned to the first rising data valid window.
- Case 2: Delay on DQS pushes it out to start somewhere in the following bit time, i.e., the rising edge of DQS is aligned to first falling edge data valid window.

Both cases are shown in Figure 14.

Case 1: DQS Edge in Corresponding Bit Time



Case 2: DQS Edge in Following Bit Time

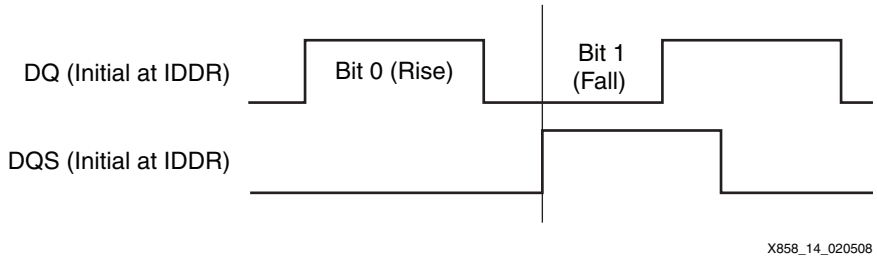


Figure 14: **Initial Phase between DQS and DQ**

Figure 15 shows the flowchart for first stage calibration.

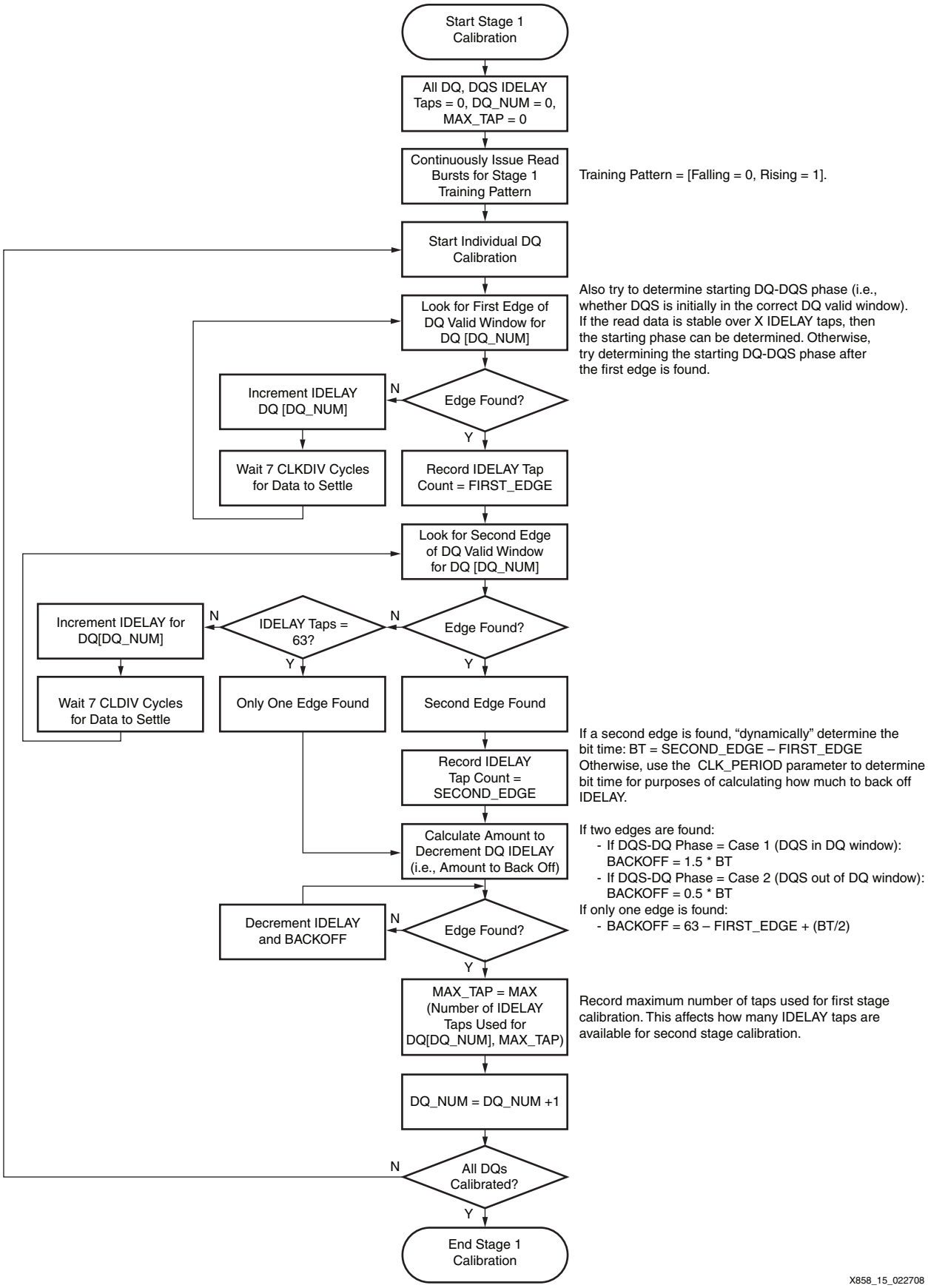


Figure 15: **Calibration Flowchart**

Stage 2 Read Timing Calibration: DQS-CLK0 Alignment

The second stage of read timing calibration aligns the rank 1 flip-flop outputs (clocked by DQS) to the sampling window of the rank 2 flip-flops (clocked by CLK0). Calibration is performed on a per-DQS basis.

Much like stage 1, an edge of the data valid window (in this case the data valid window of the IDDR output) is searched for. The IDELAY for each DQS and its corresponding DQs are incremented in lock-step during this stage to preserve the DQ-DQS relationship established in stage 1, and to vary the phase of the IDDR output with respect to CLK0.

There are two possible results when synchronizing the IDDR outputs to the CLK0 clock domain:

- Both IDDR rising and falling edge outputs are synchronized to the rising edge of CLK0. This will be known as *CLK0 rising edge synchronization* for the remainder of this section.
- Both IDDR rising and falling edge outputs are synchronized to the falling edge of CLK0. This will be known as *CLK0 falling edge synchronization* for the remainder of this section.

Both possibilities are supported because the rising and falling edge IDDR outputs each drive two CLK0 flip-flops, one clocked off the rising edge of CLK0 and the other off the falling edge of CLK0. Both CLK0 flip-flop outputs are MUXed to allow either output to be used in the rest of the design. This is shown in Figure 11.

As the logic searches for an edge of the data valid window, after every increment of IDELAY, the output MUX is varied between CLK0 rising and falling edge synchronization to check for an edge. After an edge is found, the output MUX setting is permanently set to the opposite value for which the edge was found. This has the effect of shifting the sampling point 180° from the edge of the window to the center of it. An example of this is shown in Figure 16.

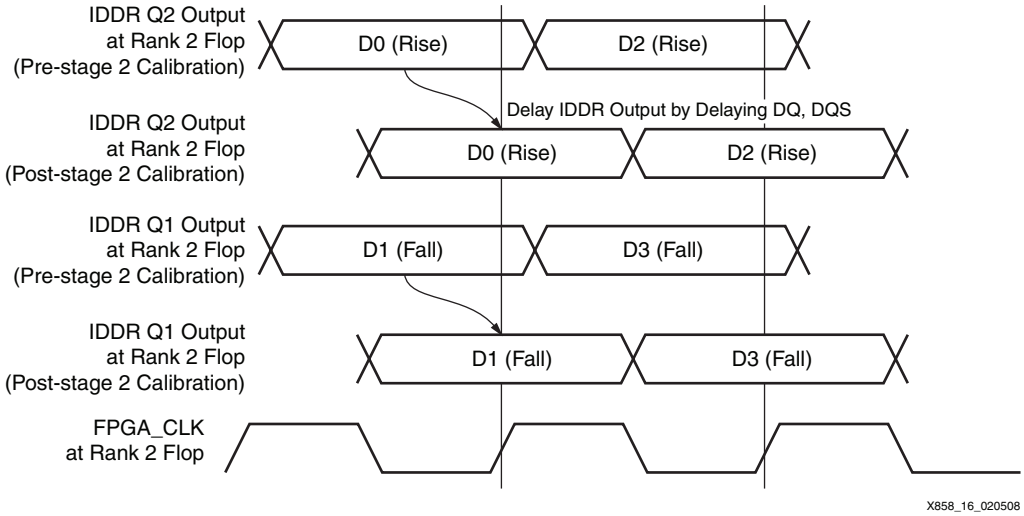


Figure 16: **Second Stage Read Timing Adjustment**

Because both DQ and DQS IDELAY taps are incremented during this stage, and because DQ IDELAY taps were consumed during stage 1 calibration, the maximum number of taps the DQ and DQS IDELAYs can be incremented for this stage is not 63; instead, the maximum number of taps is the difference of 63 minus the maximum number of taps used during stage 1.

Figure 17 shows the flow chart for second stage calibration.

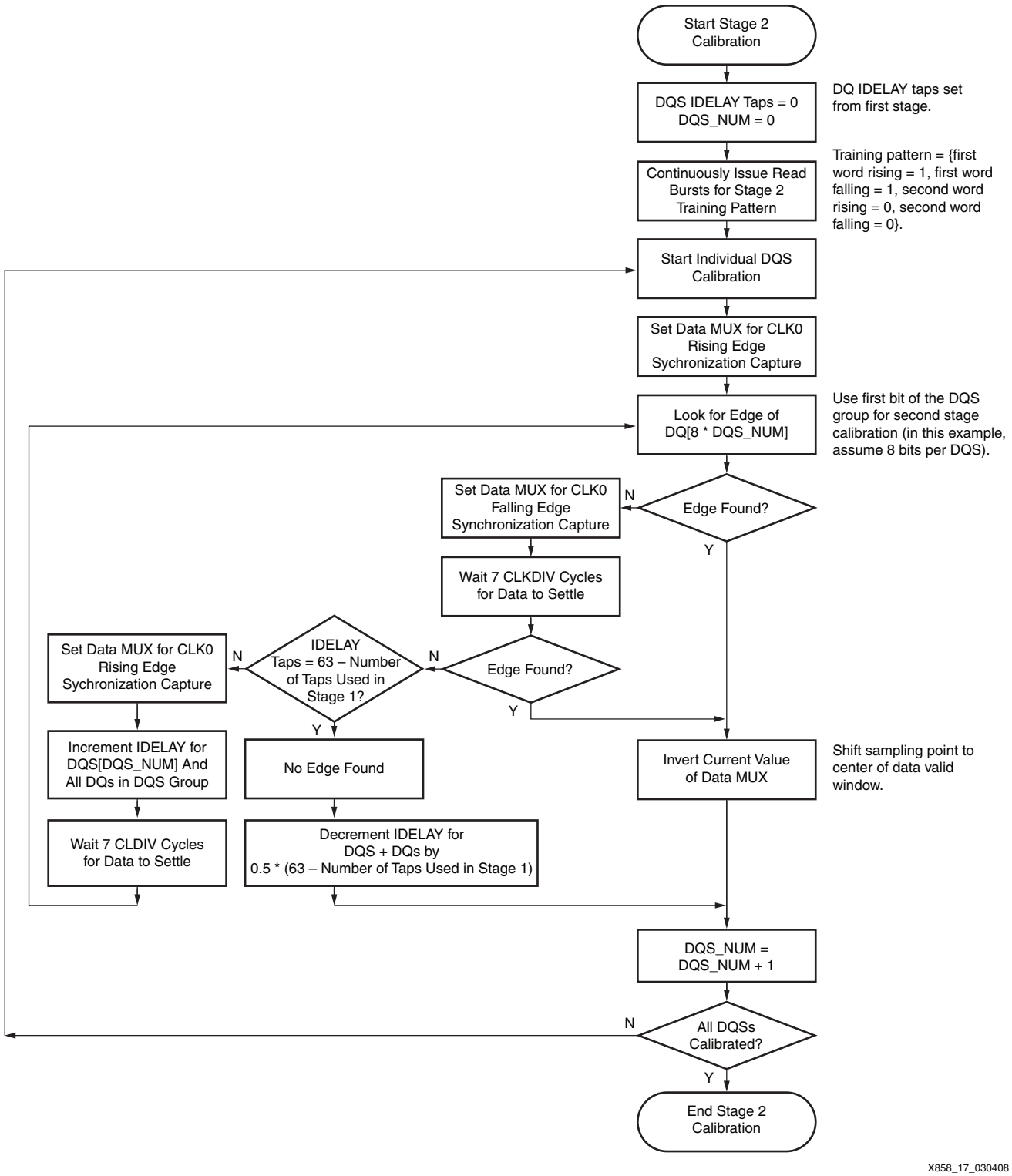


Figure 17: **Stage 2 Calibration Flowchart**

Stage 3 Read Timing Calibration: Read Data Valid Calibration

The amount of delay between when the controller issues a read command and when the corresponding read data is returned by the memory and synchronized to the CLK0 domain is dependent not only on the programmed read latency (CAS latency) of the memory, but also on various environmental factors (e.g., board layout, PCB trace lengths, and output and input path delays in both the FPGA and memory). Therefore, the controller does not know exactly on which CLK0 cycle the valid data will be available in the CLK0 domain when it issues a read command to the memory. Because the DDR2 SDRAM does not provide a read valid signal along with the read data, it is necessary to perform a calibration to determine the amount of this *round-trip* delay. In addition, differing skews can cause data for different DQS groups to be synchronized to CLK0 on different clock cycles. Calibration in effect generates an internal read data valid signal for each DQS group. In addition, some DQS groups in the data bus can need to be internally registered to align them with data from other DQS groups.

This sequence is performed for each DQS group.

1. A fixed data pattern is written to memory. This serves as a training pattern for read data valid calibration. The pattern used is eight words long: `0x11...1`, `0xEE...E`, `0xEE...E`, `0x11...1`, `0xEE...E`, `0x11...1`, `0xEE...E`, `0x11...1`. This corresponds to a sequence of `1`, `0`, `0`, `1`, `0`, `1`, `0`, `1` on the least significant DQ of each DQS group.
2. Data is read back from memory, and the read data on the least significant bit of the current DQS group is compared to the original training pattern.
3. The PHY_INIT module asserts PHY_INIT_RDEN when a read is issued, and keeps it asserted for four clock cycles. This signal is delayed until the received data matches the expected training pattern while this delayed signal is asserted. This delay is increased until a match is found.
4. This process is repeated for each DQS.

After the amount to delay PHY_INIT_RDEN is determined for all DQS groups, the final step is to decide if certain DQS groups in the data bus need to be delayed by one clock cycle to account for skews between different DQS groups. This step ensures that all bits of the data word arrive at the output of the User Interface block on the same clock cycle. The appropriate delay is determined by comparing the delay value for the PHY_INIT_RDEN across all the DQS groups; those DQS groups whose values are one less than for other DQS groups in the data word are delayed by one clock cycle. The DDR2 memory reference design can only account for one clock cycle of skew across all the DQS groups in the data bus.

After initialization/calibration is complete, the CTRL_RDEN signal from the controller logic is delayed by the amount determined during calibration to generate the RD_DATA_VALID to indicate when valid read data is present at the output of the User Interface block.

Capturing the Last Data of Read Burst

To ensure adequate timing margin on the last falling edge data capture of a read burst, the DDR2 memory design deasserts the clock enable (CE) input to all the DQ capture IDDRs. This deassertion of the IDDR CE inputs prevents any spurious DQS edges that can result following the last half clock cycle of the read (“read postamble”) when the DQS is 3-stated. The reference design uses a circuit to synchronize the control signal to deassert the IDDR CE pins from the CLK0 to the DQS domain. This circuit is referred to as the DQS Gate circuit in this document. The timing calibration for the DQS Gate circuit takes place over the fourth and final stage of calibration.

The DQS strobe driven by the memory is used to clock in read data into the first rank of capture flip-flop in the ISERDES. The DQS is a single-ended signal for DDR and can be either differential or single-ended for DDR2 memory. At the end of a DDR/DDR2 read burst, the memory device 3-states the DQS at the end of the read postamble. This occurs at a minimum 0.4 of the clock period after the last falling edge of DQS (e.g., 1.2 ns at 333 MHz).

When the DQS is 3-stated by the memory at the end of a read, DQS can be pulled up toward V_{REF} through any active termination, and DQS_N (if DQS is differential) can be pulled down toward V_{REF}. Two effects can cause this changing post-3-state DQS waveform to be interpreted as a false clock edge:

- DQS approaching crossing the DC threshold levels for logic Low and High as it is pulled toward V_{REF}.
- Overshoot and undershoot from transmission line reflections caused by the rising/falling edge of the post-3-state DQS signal. Depending on whether DQS is differential or single-ended, this can either cause DQS to cross VREF (in the case of single-ended DQS) or for the DQS and DQS_N signals to cross each other.

Typical SSTL termination on the DQS net(s) consists of either a single 50Ω pull-up resistor to V_{REF} (0.9V for DDR2 memory), or a 100Ω pull-up/100Ω pull-down Thevenin termination. These terminations can be placed at one or both ends of the net (i.e., near the memory and near the FPGA). Alternatively, only the FPGA end can be terminated with a static termination network (either an external resistor network or the on-chip Digitally Controlled Impedance (DCI) termination), with the memory end terminated using the ODT feature of the DDR2 memory.

Note: The option of external termination can only be used for slower frequencies, typically less than 200 MHz.

The read data synchronization technique employed by this design relies on two single-data-rate (SDR) streams being generated by the IDDR after the initial capture of the DQ with DQS at the IDDR. One SDR stream is for rising edge data and the other is for falling edge data. The synchronization of the IDDR outputs to the core clock (CLK0) can then take place over an entire clock cycle, rather than half a clock cycle. However, any glitches on DQS occurring after the read postamble can cause multiple clock edges to occur at the input of the IDDR. In particular, these “false” clock edges can cause the output of the IDDR to switch before the output can be held for one full clock cycle. Specifically, it is a false falling edge that can cause the IDDR output to change before it can be held for one full clock cycle.

This effect is shown in Figure 18.

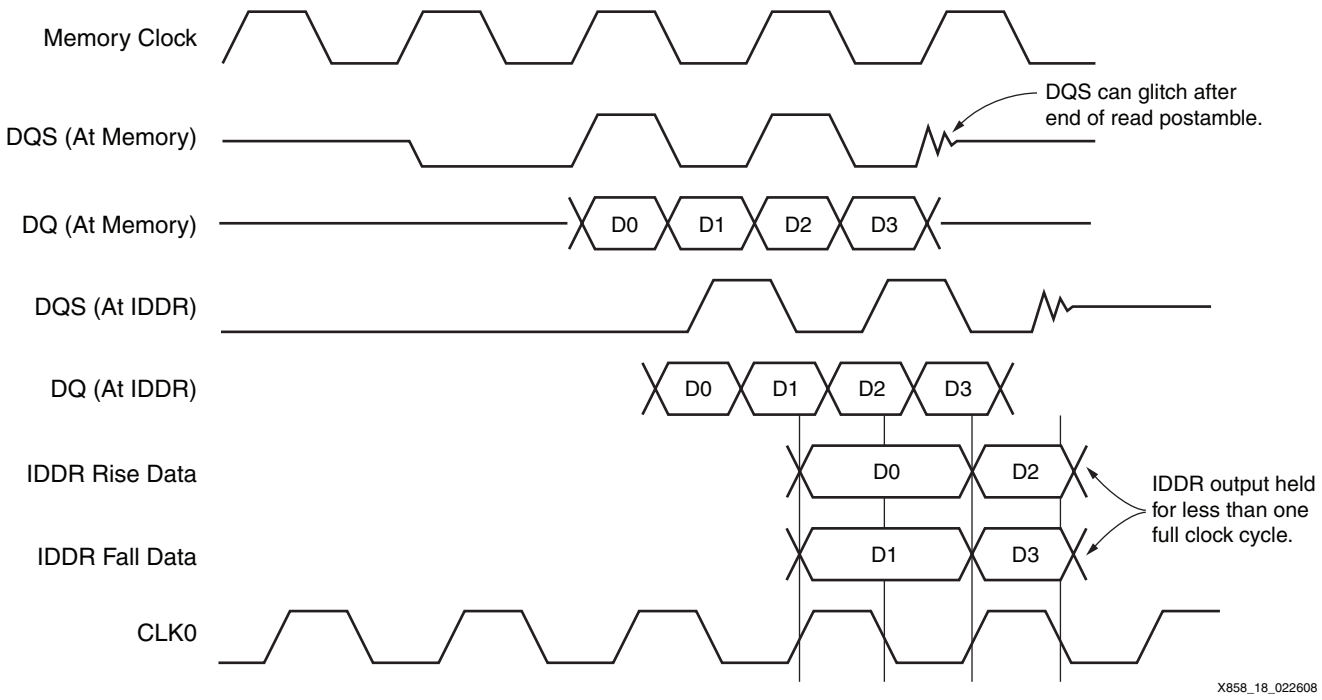


Figure 18: **Effect of DQS Postamble Glitch on Read Capture Synchronization**

The DDR2 memory interface design employs a circuit to disable the clock enable on all the DQ IDDRs after the last falling edge of the read burst. This circuit uses the following elements:

- The controller asserts the clock enable (EN_DQS) for all the DQ IDDRs in the interface. This signal is deasserted when a consecutive group of one of more reads is about to end. This deassertion is determined by the CTRL_RDEN signal going Low. EN_DQS is synchronous to CLK0.
- An IDELAY receiving its input from a CLB flip-flop (rather than the corresponding IOB input) is used to synchronize EN_DQS signal from the CLK0 domain to the DQS domain. One such IDELAY is used for every DQS to compensate for timing differences between DQS groups. This IDELAY must be taken from an available I/O site—one that is not already using its IODELAY block. Possible sites include output-only sites (e.g., data mask (DM) or DDR2 control/address outputs), unused I/O, V_{RN}/V_{RP}, V_{REF}, or sites that correspond the N-side of differential outputs (e.g., DQS_N). This reference design uses the IDELAY in each DQS_N I/O.
- The IDELAY output drives the asynchronous set and D input to the IOB flip-flop in the same site. This flip-flop is clocked by the falling edge of DQS, and its output drives the clock enable (CE) inputs to all the DQ IDDR flip-flops in that DQS group. When EN_DQS is asserted, the output of the IDDR is asynchronously set to `1`. When it is deasserted, the D input of the IDDR is `0` and the output of the IDDR is driven to `0` on the next falling edge of DQS.

This circuit is shown in Figure 19 for a single DQS group.

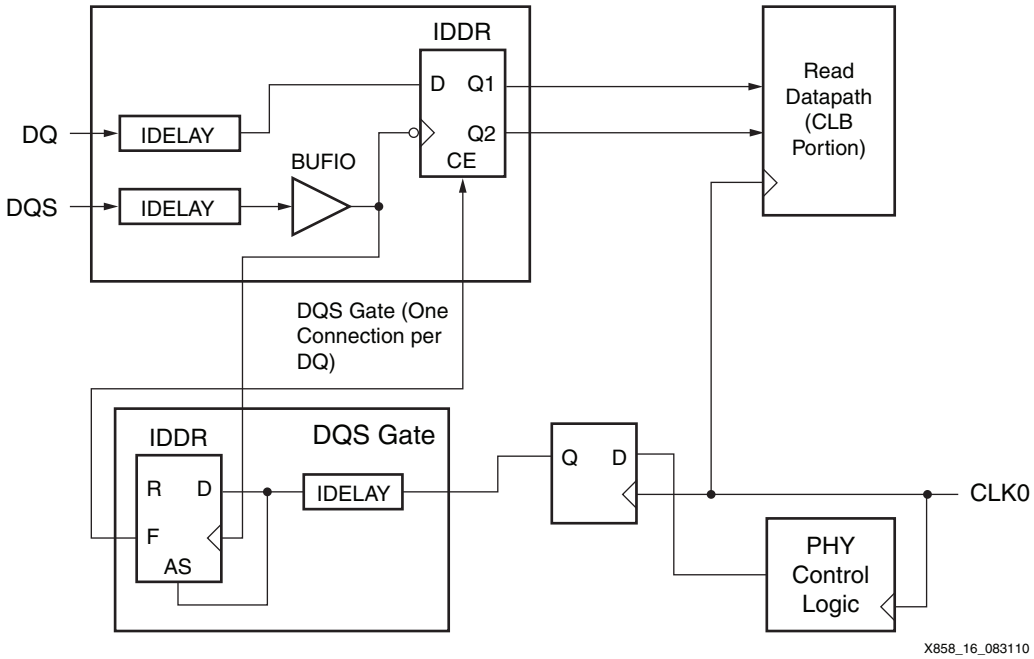


Figure 19: **DQS Postamble Glitch Gating Circuit**

The value of the IDELAY necessary to synchronize the CLK0 EN_DQS flip-flop output to the DQS domain is determined during the fourth stage of read timing calibration. Each DQS group is calibrated separately.

Because the DQS is no longer active after a read burst is completed, there must be a method to re-enable DQS to each of the DQ ISERDES before the next read burst. This is accomplished by having the DDR controller drive the asynchronous set input of the GATE_SYNC flip-flop after the last word from the current read burst has been transferred to the FPGA clock domain.

One very important note: There is not enough time to allow the effect of this asynchronous set to propagate before the next rising edge of DQS in the case of two reads separated by one idle

cycle. This reference design assumes that the DQS is either held Low between the two reads in this case, or does not have sufficient time to glitch, causing a false edge. Therefore, in the case of two reads separated by one idle cycle, the DQS is not gated.

The DDR2 memory interface design assumes the use of ODT at the DDR2 load, and the built-in Digital Controlled Impedance (DCI) or external V_{TT} pull-up termination at the FPGA load, rather than the use of separate external V_{TT} pull-up terminations at both the FPGA and DDR2 loads. This requirement only applies at higher memory bus frequencies and is a result of the timing achievable with the DQS Gate circuit. At lower frequencies, the DQS Gate circuit is still required; however, the requirement that ODT be used at the DDR2 load is not. The requirement for the use of ODT at the DDR2 load is described below.

The path between the DQS Gate IDDR and CE inputs to the DQ IDDRs in that DQS group must be timing constrained because it is not a full cycle path. At a minimum, the path delay must be less than 0.4 * (clock cycle); this is the time between the last falling edge of DQS and when DQS is 3-stated. However, on the PCB, it takes some time for DQS/DQS_N to slew toward V_{REF} and possibly result in a rogue clock edge. The slew rate of DQS is dependent on the resistance of the termination pull-ups, the line capacitance, and the trace routing/topology. There is some additional time after the DQS is 3-stated before a false clock edge can occur. To decrease the rise time of the DQS after it is 3-stated, the effective pull-up termination on the bus must be minimized. Therefore, the DDR2 reference design ensures that the only pull-up terminator active at the time of the last data word is the DCI (or external pull-up) termination at the FPGA end. During this time, ODT at the DDR2 memory is disabled by the reference design logic.

This DDR2 reference design assumes 400 ps extra time after DQS is 3-stated before any falling edge glitch can occur. A different value can be required depending on the user's particular PCB routing, memory bus loading, and termination scheme.

Stage 4 Read Timing Calibration: DQS Gate Calibration

The actual logic that determines when to gate the DQS needs to be generated in the CLK0 domain. This is because the clock gating signal is generated by the DDR2 controller logic, which is clocked off CLK0. Consequently, the clock gating signal that is produced by the DDR2 controller must be synchronized to the DQS clock domain. This synchronization is done by routing the FPGA clock domain signal first through an IDELAY (accessible by a CLB for Virtex-5 devices).

The synchronization between the CLK0 and DQS clock domains does require a calibration step to ensure that the IDDR flip-flop clocked by the DQS directly driving the CE for the DQ capture IDDRs (GATE_SYNC) is sampling the clock gating control signal from the DDR2 controller near the center of the clock cycle.

Stage 4 calibration determines:

- The number of CLK0 clock cycles to delay the EN_DQS signal so that the set input to the GATE_SYNC flip-flop is deasserted just prior to the last falling edge of DQS on a read.
- The IDELAY setting such that within the correct clock cycle, the EN_DQS signal routed to the set input of the IDDR is deasserted near the center of the DQS clock cycle.

This sequence is performed for each DQS group.

1. A fixed data pattern is written to memory. This serves as a training pattern for read data valid calibration. The pattern used is eight words long: `0x11...1`, `0xEE...E`, `0xEE...E`, `0x11...1`, `0x11...1`, `0xEE...E`, `0xEE...E`, `0x11...1`. This corresponds to a sequence of `1`, `0`, `0`, `1`, `1`, `0`, `0`, `1` on the least significant DQ of each DQS group.
2. The PHY_INIT module asserts PHY_INIT_RDEN when a read is issued and keeps it asserted for four clock cycles. EN_DQS is generated from the falling edge of PHY_INIT_RDEN (i.e., this falling edge indicates when the read has ended).

3. The calibration pattern checks for the presence of the following target pattern on the least significant DQ of the DQS group: `1, 0, 0, 1, 1, 0, 1, 0`. This is the same sequence as the original training pattern with the exception of the last two bits. This sequence is produced when the CE of the DQ IDDR capture flip-flops is disabled after the second-to-last falling edge of DQS.
4. EN_DQS is driven without any delay to the DQS Gate circuits. This value is guaranteed to be at least one cycle too early than the earliest deassertion timing requirement for EN_DQS signal. EN_DQS is deasserted for three clock cycles at the end of the read burst.
5. The read data is checked for the presence of the target pattern. Since the EN_DQS deassertion is too early, the data returned does not match the target pattern, and instead either falls into one of two cases: (1) if the EN_DQS was so early that it deasserted and reasserted before the read data was synchronized, then the captured read data will be the training pattern itself, (2) if the EN_DQS was deasserted at least some time during the synchronization of the read data, some of the captured read data will differ from the original training pattern based on when the EN_DQS was deasserted.
6. The tap value of the IDELAY used to synchronize EN_DQS to the DQS domain is incremented by one. The read data is checked for the presence of the target pattern.
7. The previous step is repeated until either the target pattern is found or the IDELAY tap count reaches either 32 or (bit time/78 ps), whichever comes first. If the target pattern has not yet been found, the delay for EN_DQS is increased by one clock cycle, and the IDELAY tap value is reset to `0`.
8. If the target pattern is found while incrementing the IDELAY, then the “right” edge of the valid window has been found. At this point, the phase of EN_DQS and DQS at the IDDR are aligned. Subtract the lesser of 32 or (bit time/78 ps) from the IDELAY to the sampling of EN_DQS by the IDDR flip-flop to move the sampling point away from the edge.
9. If the EN_DQS delay is incremented by one, and the target pattern is found immediately, then the “left” edge of the valid window is found. Add the lesser of 32 or (bit time/78 ps) from the IDELAY to the sampling of EN_DQS by the IDDR flip-flop to move the sampling point away from the edge.
10. This process is repeated for each DQS.

After initialization/calibration is complete, the CTRL_RDEN signal from the controller logic is used to generate EN_DQS.

Read Timing Analysis

For error-free read data capture, read data and strobe must be delayed to meet the setup and hold time requirements of the flip-flops in the FPGA clock domain. Read data (DQ) and strobe (DQS) are received edge aligned at the FPGA. The differential DQS pair must be placed on a clock-capable I/O pair to access the BUFIO resource. The received read DQS is then routed through the BUFIO resource to the CLK input of the IDDR of the associated data bits. The delay through the BUFIO and clock routing resources shifts the DQS to the right with respect to data.

Table 8 is a sample read timing analysis for a DDR2 interface at 333 MHz for a -3 speed grade Virtex-5 device. ISI, crosstalk, user input clock jitter, and contributors to dynamic skew are not considered in this analysis.

Table 8: **Read Timing Analysis at 333 MHz**

Parameter	Value (ps)	Description
T_{CLOCK}	3003.00	Clock period.
T_{PHASE}	1501.50	Data period, half of the clock period.
Memory Uncertainties	580	Tdqsq + Tqhs = 240 ps + 340 ps = 580 ps. For -3E DDR2 speed grade.
T_{SAMP_BUFIO}	350	Sampling error at input register.

Table 8: **Read Timing Analysis at 333 MHz *(Cont'd)***

Parameter	Value (ps)	Description
T_{DCD_JITTER}	100	DCD due to BUFIO network.
Number of IDELAY Taps Used	35	Worst case number of taps used to align DQ to DQS for capture using the IDDR flip-flop. This is a function of the clock period, and differing arrival times for the DQ and DQS at the IDDR (e.g., due to BUFIO additional delay on DQS before reaching the IDDR).
T_{IDELAY_JIT}	350	10 ps/tap in HIGH_PERFORMANCE_MODE
Uncertainties	1380	
Data Valid Window	121.50	Worst-case window

Notes:

1. Parameters such as BUFIO skew, package skew, and PCB layout skew are calibrated with the read data capture timing calibration technique.

PHY Layer Interface

The PHY layer can be used independently of the Controller and User Interface modules. In this case, the user must supply custom DDR2 controller logic. Functions like opening/closing of rows, bank management, memory refresh, and read/write access timing must be managed by this controller.

The modules that form the PHY layer are shown in Table 9. The top-level parameters/generics for the DDR2 interface design are discussed in "Reference Design," page 40. Some of the parameters listed in Table 16 are available at PHY_TOP for those designs using only the PHY portion of the code. A general description of each module in the PHY layer is shown in Table 9.

Table 9: **PHY Design Files**

Module/Filename	Description
PHY_TOP	PHY interface top-level module.
PHY_CTL_IO	Address/control and forwarded memory clock output flip-flops.
PHY_INIT	Initialization state logic. Performs power-up initialization of memory. Issues the appropriate memory accesses during read timing calibration.
PHY_WRITE	Generation of internal and external control signals associated with memory writes, such as output DQS and ODT.
PHY_IO	Instantiates PHY_CALIB, PHY_DQS_IOB, and PHY_DQ_IOB modules.
PHY_CALIB	Read data capture timing calibration logic.
PHY_DQ_IOB	Data write output and read input paths.
PHY_DQS_IOB	Strobe write output and read input paths.

Table 10 lists the I/O ports of the PHY layer. The PHY layer provides an interface to both the controller (for initiating commands) and the user backend write/read datapaths (for providing data on a write, and accepting data on a read). Control signals with a "_n" suffix in the name are active-Low. Unless otherwise noted, all signals are synchronous with CLK0. Table 10 does not show the optional PHY Debug Port signals; these signals are discussed in Appendix D of the *MIG User Guide* [Ref 3].

Table 10: **PHY Interface Signals**

Signal	I/O	From/To	Width	Description
clk0	I	BUFG	1	System clock.
clk90	I	BUFG	1	90° shifted version of system clock.
clkdiv0	I	BUFG	1	Divided-by-2 (and synchronous) version of system clock.
rst0	I		1	Synchronous reset (for clk0 logic).
rst90	I		1	Synchronous reset (for clk90 logic).
rstdiv0	I		1	Synchronous reset (for clkdiv0 logic).
ctrl_addr	I	Controller	ROW_WIDTH	Row/column address to memory.
ctrl_ba	I	Controller	BANK_WIDTH	Bank address to memory.
ctrl_ras_n	I	Controller	1	Row address strobe to memory.
ctrl_cas_n	I	Controller	1	Column address strobe to memory.
ctrl_we_n	I	Controller	1	Write enable to memory.
ctrl_cs_n	I	Controller	CS_NUM	Chip select to memory.
ctrl_wren	I	Controller	1	Asserted during a write to the PHY. The PHY uses this signal to toggle DQS, control the 3-state pin of the DQ/DQS bidirectional bus, and generate the read enable for the Write Data FIFO. Must be asserted starting on the same cycle as the CTRL_CAS_N, CTRL_WE_N, etc., signals indicating a write command, and held for 2 * {burst length} cycles.
ctrl_rden	I	Controller	1	Asserted during a read from the PHY. The PHY uses this signal to indicate when the read data from the memory is valid. Must be asserted starting on the same cycle as the CTRL_CAS_N, CTRL_WE_N, etc., signals indicating a read command, and held for 2 * {burst length} cycles.
ctrl_ref_flag	I	Controller	1	One-clock-cycle-wide pulse asserted after every auto refresh interval (7.8 µs). This signal is used by the PHY logic to ensure the proper refresh interval during initialization and calibration.
wdf_rden	O	Write Datapath	1	Read enable for Write Data FIFO (synchronous with CLK90). The timing on this signal with respect to ctrl_wren.
wdf_data	I	Write Datapath	2*DQ_WIDTH	Write data (CLK90).
wdf_mask_data	I	Write Datapath	DQ_WIDTH/4	Write data mask (CLK90).
phy_init_done	O	Controller	1	This signal is asserted when memory initialization and read data capture timing calibration is complete. This indicates when the controller can begin issuing commands to the memory. All control to the PHY except the ctrl_ref_flag signal cannot be asserted until at least two cycles after the assertion of phy_init_done.

Table 10: **PHY Interface Signals *(Cont'd)***

Signal	I/O	From/To	Width	Description
phy_calib_rden	O	Read Datapath	DQS_WIDTH	Data valid strobe. Asserted when rd_data_rise and rd_data_fall output of the PHY contain valid read data.
phy_calib_rden_sel	O	Read Datapath	DQS_WIDTH	Used to account for skews between DQS groups in the read data. Indicates the difference in delays between the various phy_calib_rden outputs. This signal should be used in the read datapath logic to account for different read delays between different DQS group lanes. Each bit in this bus corresponds to a DQS group. = 1: All DQ bits in that DQS group must be delayed by one clock cycle = 0: No cycle delay is necessary for the DQ bits in that DQS group
rd_data_rise	O	Read Datapath	2*DQ_WIDTH	Rising edge read data from memory. Data from the memory does not necessarily arrive aligned on the same clock cycle. There can be a skew of one clock cycle across all DQS group lanes. Use phy_calib_rden_sel to account for possible skew.
rd_data_fall	O	Read Datapath	2*DQ_WIDTH	Falling edge read data from memory. Data from the memory does not necessarily arrive aligned on the same clock cycle. There can be a skew of one clock cycle across all DQS group lanes. Use phy_calib_rden_sel to account for possible skew.
ddr_ck	O	DDR2	2*CLK_WIDTH	Forwarded differential clock to DDR2 (P-side).
ddr_ck_n	O	DDR2	2*CLK_WIDTH	Forwarded differential clock to DDR2 (N-side).
ddr_addr	O	DDR2	ROW_WIDTH	DDR2 row/column address.
ddr_ba	O	DDR2	BANK_WIDTH	DDR2 bank address.
ddr_ras_n	O	DDR2	1	DDR2 row address strobe.
ddr_cas_n	O	DDR2	1	DDR2 column address strobe.
ddr_we_n	O	DDR2	1	DDR2 write enable.
ddr_cs_n	O	DDR2	CS_NUM	DDR2 chip select.
ddr_cke	O	DDR2	CKE_WIDTH	DDR2 clock enable.
ddr_odt	O	DDR2	ODT_WIDTH	DDR2 on-die termination control.
ddr_dm	O	DDR2	DM_WIDTH	DDR2 data mask.
ddr_dqs	O	DDR2	DQS_WIDTH	DDR2 bidirectional differential data strobe (P-side).
ddr_dqs_n	O	DDR2	DQS_WIDTH	DDR2 bidirectional differential data strobe (N-side).
ddr_dq	O	DDR2	DQ_WIDTH	DDR2 bidirectional data.

Figure 20 shows the PHY layer write path timing for these conditions:

- Burst length = 8
- Two consecutive writes followed by another write two clock cycles later. Only the very first write is to a non-open row.
- RAS to CAS strobe spacing = 4 clock cycles
- Write latency = 2 clock cycles

The corresponding signals on the DDR2 bus are also shown for the first two write bursts.

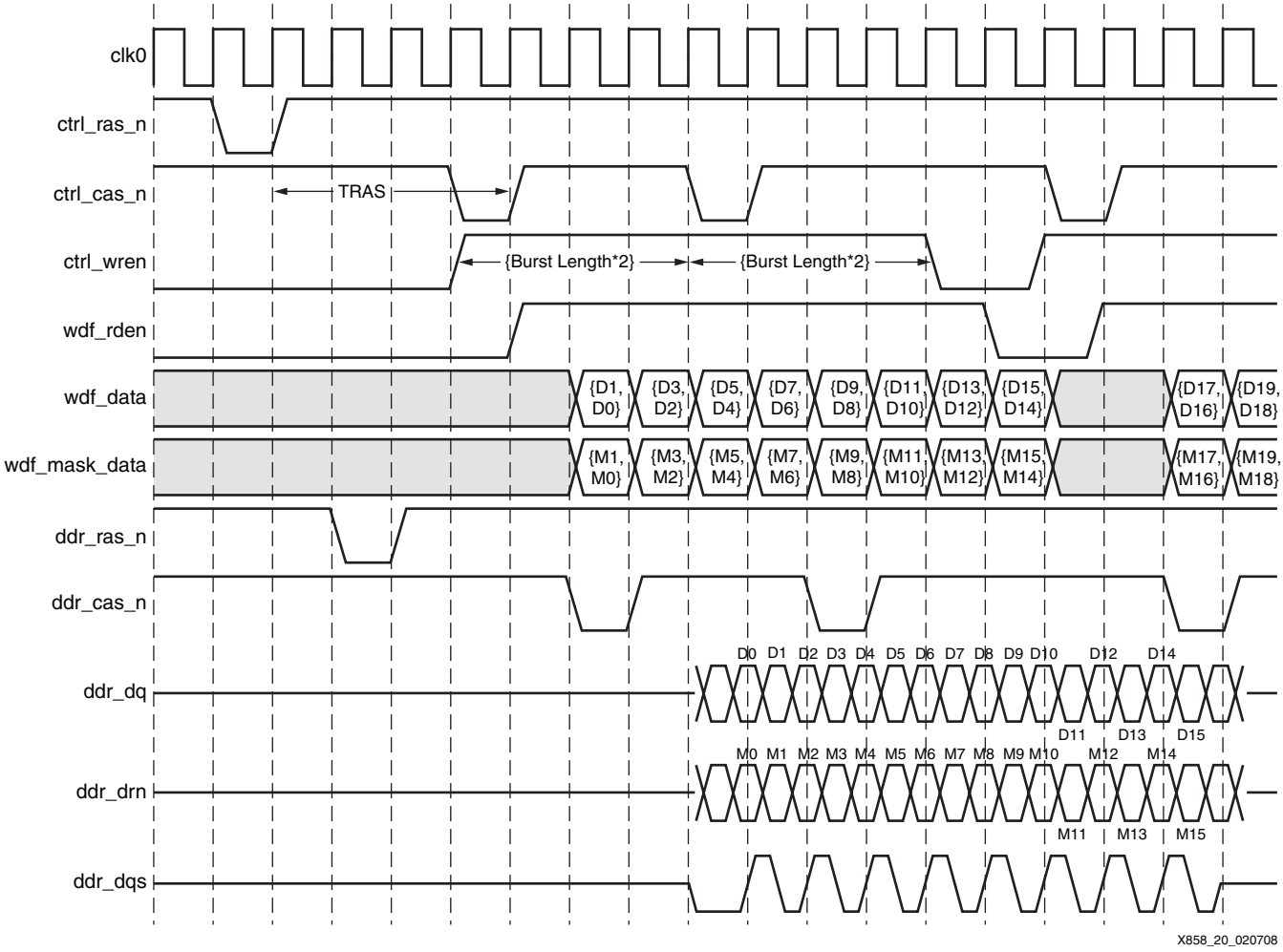


Figure 20: **Controller-PHY-Memory Timing (Write Path, BL=8)**

Figure 21 shows the strobe signal and ctrl_wren timing for a write followed by two consecutive writes with a burst length of 4.

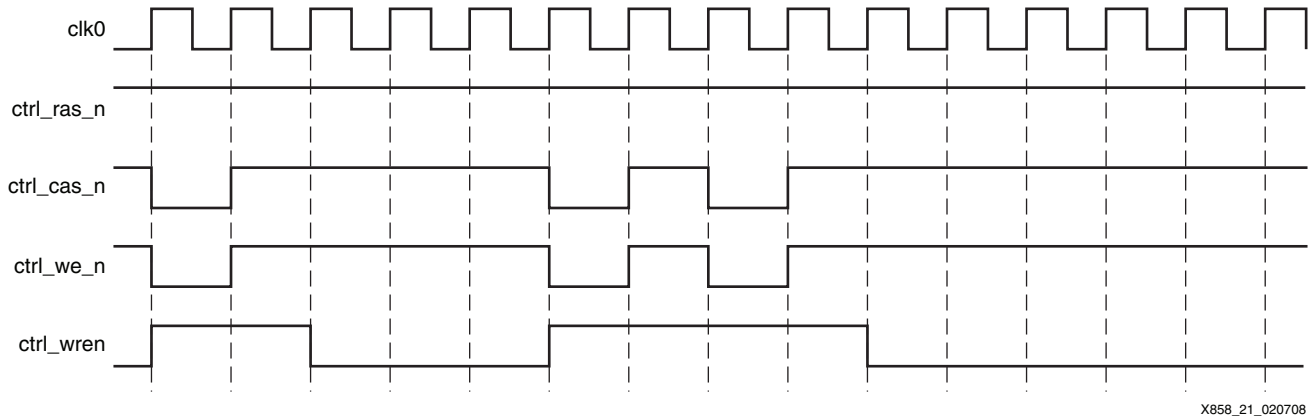


Figure 21: **Controller-to-PHY Timing (Consecutive Writes, BL=4)**

Figure 22 shows the PHY layer read path timing for these conditions:

- Burst length = 8
- Two consecutive reads followed by another read two clock cycles later. Only the very first read is to a non-open row.
- RAS to CAS strobe spacing = 4 clock cycles
- CAS latency = 3 clock cycles

The corresponding signals on the DDR2 bus are also shown for the first two read bursts. The latency between assertion of ctrl_rden and when read data is available at the PHY outputs (TPHY_RD_LATENCY as shown in the diagram) is not only a factor of CAS latency, but also other delays, such as PCB board trace lengths, and propagation delay uncertainties in the DDR2 data output path, and FPGA input capture path. This latency is determined dynamically during read capture timing calibration.

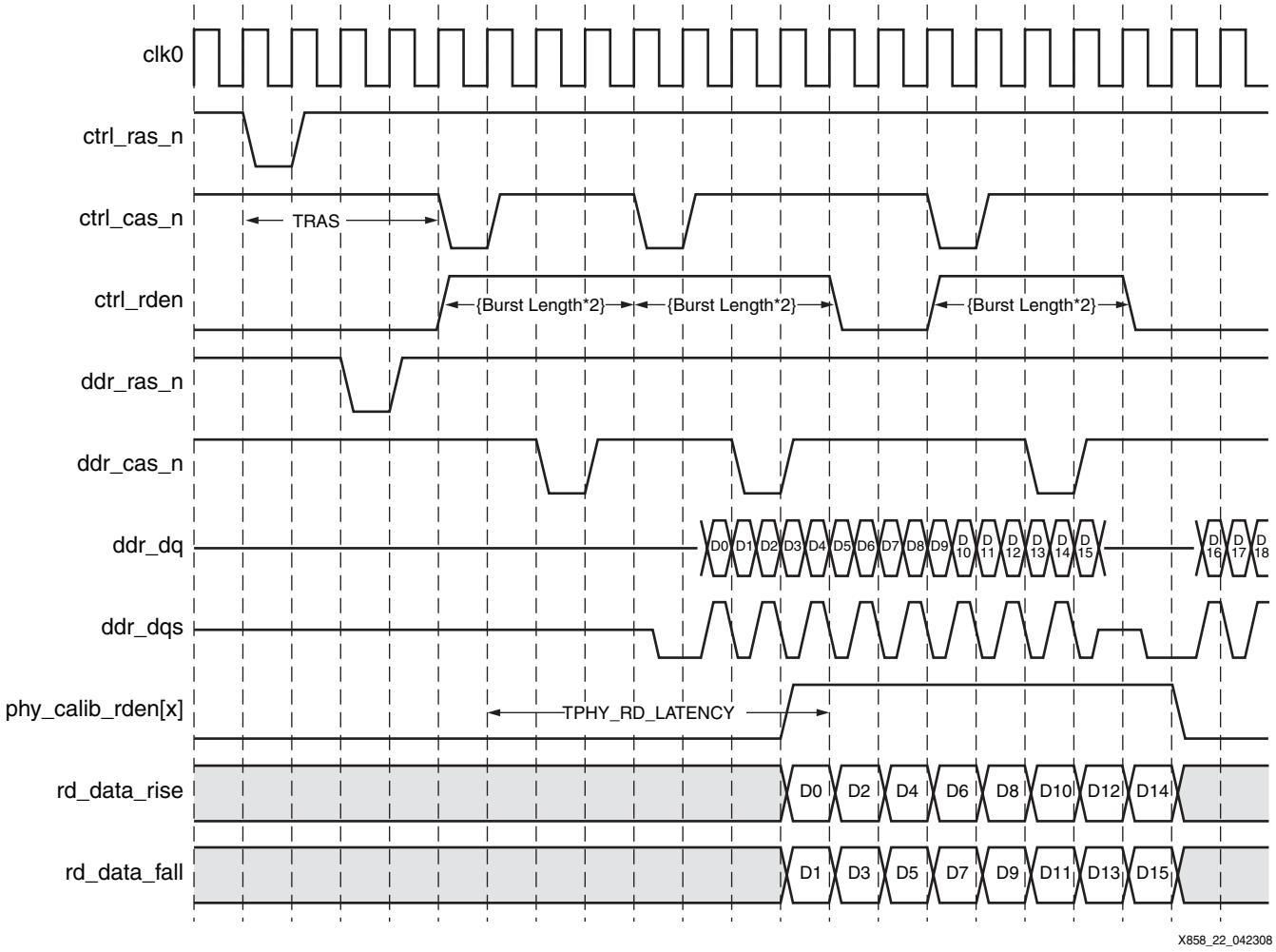


Figure 22: **Controller-PHY-Memory Timing (Read Path, BL=8)**

Figure 23 shows the strobe signal and ctrl_rden timing for a read followed by two consecutive reads with a burst length of 4.

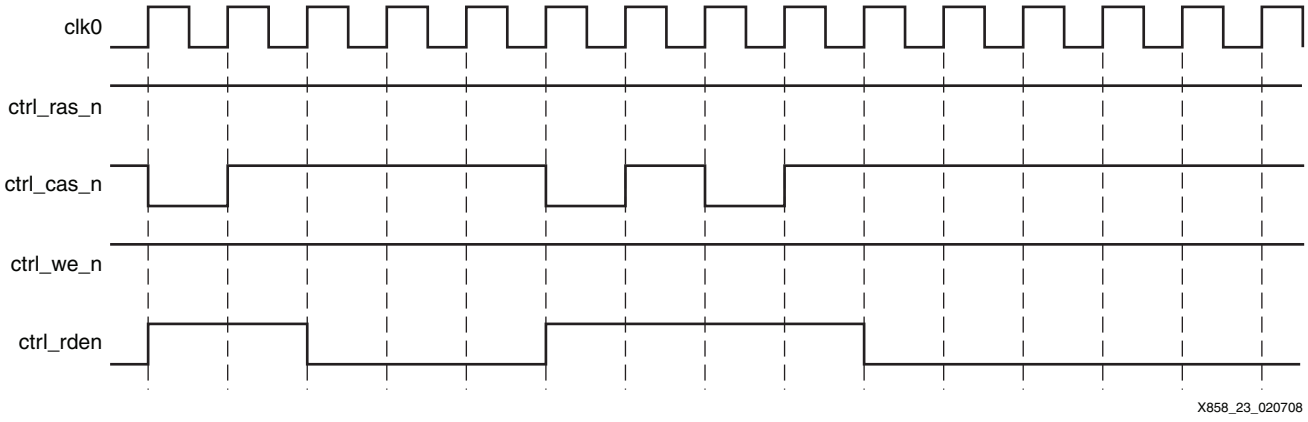


Figure 23: **Controller-to-PHY Timing (Consecutive Reads, BL=4)**

Controller Implementation

Controller Overview

The CTRL module is the main controller for the DDR2 interface design. It processes the commands from the User Interface block and issues the required read, write, activate, precharge, and auto refresh commands to the DDR2 SDRAM.

The controller state machine is held in an idle state while the initialization state machine in the PHY layer initializes the DDR2 SDRAM device and issues read and writes for read data timing calibration. The initialization state machine asserts the phy_init_done signal after the completion of the initialization and calibration. The controller state machine remains in the IDLE state until assertion of phy_init_done.

The controller issues periodic auto refresh commands according to the DDR2 SDRAM specification.

Controller to User Interface

The user backend logic issues read and write requests through the User Interface Address/Control FIFO. The User Interface Address/Control FIFO signal af_empty indicates the status of user requests. The af_empty signal is deasserted when there is a user request pending in the Address/Control FIFO. The controller state machine processes user command requests passed through this FIFO.

Table 11 lists the command/address buses passed between the User Interface block and the controller logic.

Table 11: **Output of User I/F Address/Control FIFO**

Port Name	Port Width (in Bits)	Port Description
af_addr	31	Full memory address - This is formed by the concatenation of column, row, and bank addresses, as well as the chip-select.
af_cmd	3	Command for the controller: 000: Write 001: Read 010 – 111: Reserved

Memory Address Mapping

The DDR2 memory is accessed as a linear address space by the user backend application. The user provides the target address to the DDR2 controller through the app_af_addr bus via the User Interface Address/Control FIFO. The exact mapping of the column, row, and bank bits to the address provided by the user interface is dependent on the specific memory device (in particular the density of the device). The formula used to determine this is shown in Table 13.

Bank Management

The controller supports two possible modes of operation for bank management.

1. Single bank mode
2. Multi-bank mode

The bank management mode is selected via the MULTI_BANK_EN top-level parameter.

Single Bank Mode

In the single bank mode, the controller keeps one bank and row combination open at any given time. The target bank/row of the current command from the User Interface block determines which bank/row is first opened. This bank/row is kept open as long as accesses from the User Interface block continue to target only that bank/row. If the access targets a different row in the opened bank or a different bank, the controller detects that a conflict has occurred. It then issues a precharge command to close the current opened bank, followed by an activate command to open the bank and row.

The controller always closes the currently opened bank/row whenever it needs to issue a periodic auto-refresh command. After the completion of the auto refresh command, accesses from the User Interface block again determine which banks are opened.

Multi-Bank Mode

In the multi-bank mode, the controller keeps up to four bank/rows open at any given time. The banks are opened when commands are issued from the User Interface block. When a command is received from the User Interface block, the controller makes a decision as to what it needs to do to support this access:

- Bank conflict: The target bank address does not match any of the currently opened banks.
- Bank hit: The target bank address matches one of the four currently opened banks.
- Row conflict: The target bank address matches one of the currently opened banks, however, the target row address is different than the currently open row in that bank.

The controller indicates a conflict has occurred if there is a bank conflict or if there is a bank hit and a row conflict. For a bank conflict, if the controller has already opened four banks, it will close the least recently opened bank using the precharge command and open the new bank using an activate command. If the controller has not already opened four banks, then the controller will issue an activate command to open the new bank. In the case of a bank hit and row conflict, the controller will close the bank that had the bank hit using a precharge command and will open the new row in the bank using the activate command.

The controller closes all opened banks before issuing an auto refresh command. After the completion of the auto refresh command, accesses from the User Interface block again determine which banks are opened.

Figure 24 shows the bank management logic flow for the multi-bank case.

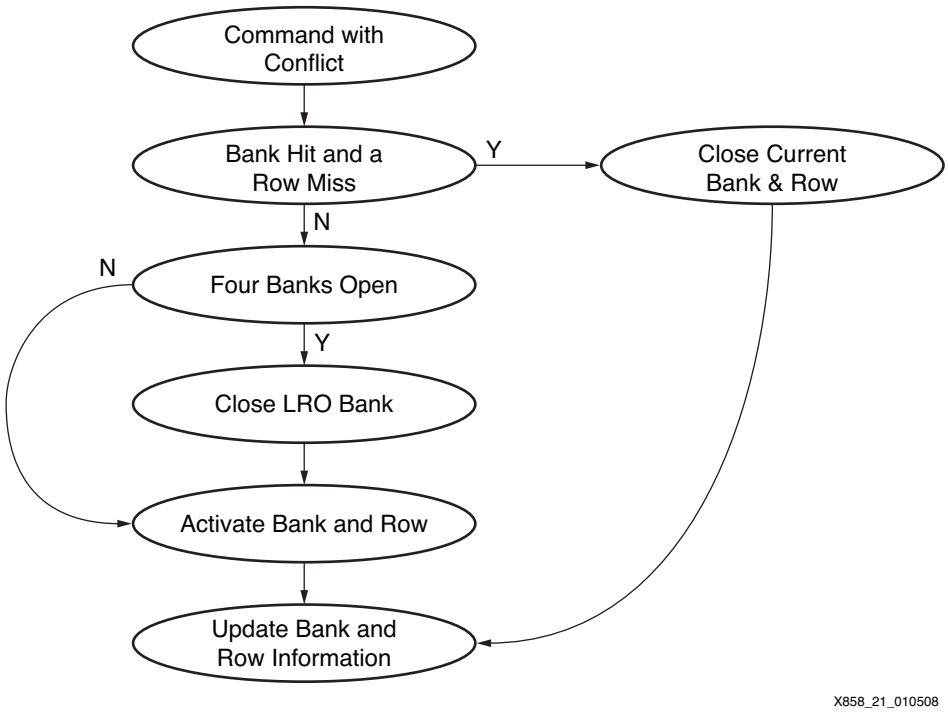


Figure 24: **Multi-bank Bank Management Flow**

Controller Commands

Table 12 shows the commands that the controller can issue to the DDR2 SDRAM. The user through the user backend can issue only read and write commands. The controller issues the required SDRAM commands listed below to perform the read and write transactions.

Table 12: **DDR2 Controller Supported Commands to Memory**

Command	Description
Activate	An activate command to an unopened bank/row is issued for write and read accesses from the User Interface block. The controller also issues an activate command for the last accessed bank/row following an auto-refresh.
Read	A read command is issued to an open bank/row for read requests from the User Interface block.
Write	A write command is issued to an open bank/row for write requests from the User Interface block.
Precharge	A precharge command is issued under these conditions: • A PRECHARGE ALL command (to close all banks) is issued before an auto refresh command • A precharge of a particular bank is issued when a bank/row conflict is detected
Auto Refresh	An auto refresh command is issued periodically based on the memory auto-refresh interval timing requirements. Auto refresh commands can take precedence over pending read and write requests. If the controller is in the middle of a read or write transaction, and if the auto-refresh interval flag is asserted, the controller finishes the current burst and proceeds with the auto-refresh command.

Controller State Machine

The state diagram for the controller logic is shown in Figure 25.

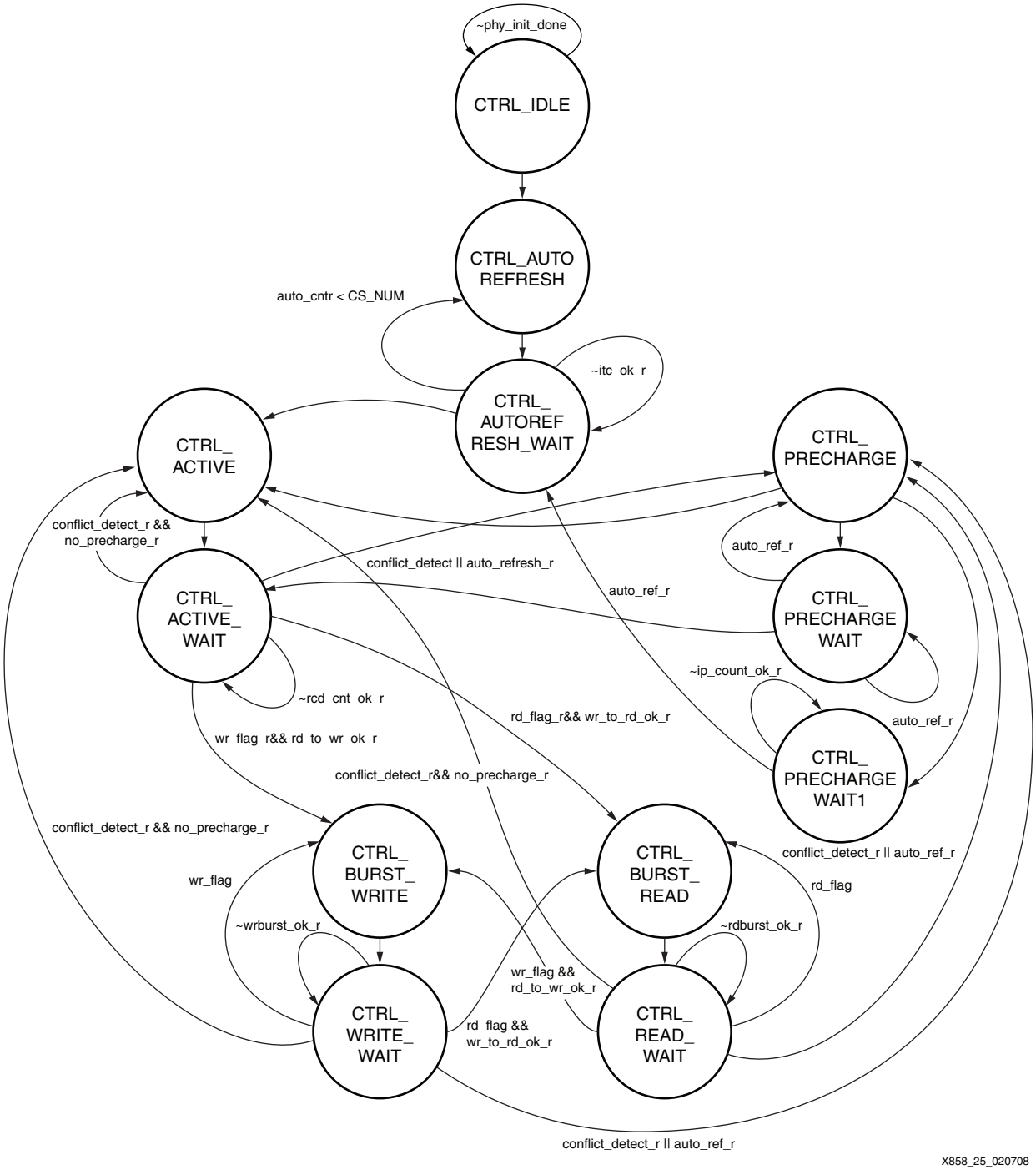


Figure 25: **Controller State Machine Logic**

User Interface Implementation

User Interface Overview

The User Interface block forms the bridge between the user-specific backend application logic, and the DDR2 controller and PHY layer logic. The User Interface block provides a FIFO interface for issuing commands and sending write data to the DDR2 memory, and a valid strobe-based interface for returning read data to the user logic.

Description

The User Interface block consists of two FIFOs: the Address/Command FIFO and the Write Data FIFO. Commands (read/write) and target addresses to the memory controller are written to the Address/Command FIFO. If the issued command is a write, the corresponding output data must be written into the Write Data FIFO. The DDR2 design instantiates built-in block RAM FIFO blocks for each of these FIFOs. One FIFO36 is used for the Address/Command FIFO. FIFO36_72 blocks are used for the write data, with the exact number used depending on the data bus width. The Address/Command FIFO is configured as a synchronous FIFO, where both write and read ports are clocked by CLK0. The Write Data FIFO is configured as an asynchronous FIFO; the write port is clocked by CLK0, and the read port by CLK90.

When issuing a write command, the first write data word must be written to the Write Data FIFO either prior to or on the same clock cycle as the when the corresponding write command is written to the Address/Command FIFO. In addition, the write data must be written by the user over consecutive clock cycles; there cannot be a break between words. These restrictions arise because the controller assumes that write data is available when it receives the write command from the user.

During reads, the read data is presented at the output of the User Interface block on the read data bus and is qualified by a read valid signal.

User Interface to Backend Logic Interface Signals

Table 13 lists the bus signals between the DDR2 User Interface block and the specific user backend application. All control signals are active-High unless otherwise noted.

Table 13: **User Interface Signals**

Signal	Direction	Width	Description
app_af_addr	Input	31	Target address from user logic. This bus is mapped to the DDR2 address bus as follows: Column address = app_af_addr[COL_WIDTH-1:0] Row address = app_af_addr[ROW_WIDTH+COL_WIDTH - 1:COL_WIDTH] Bank address = app_af_addr[BANK_WIDTH+ ROW_WIDTH+COL_WIDTH-1:ROW_WIDTH+COL_WIDTH]
app_af_cmd	Input	3	Command from user for Address/Command FIFO: 000: write 001: read Other bit combinations are reserved.
app_af_wren	Input	1	Address FIFO write enable.
app_wdf_data	Input	2*DQ_WIDTH	Write data from the user for write data FIFO. Rising edge data = app_wdf_data [DQ_WIDTH-1:0] Falling edge data = app_wdf_data [2*DQ_WIDTH-1:DQ_WIDTH]

Table 13: **User Interface Signals *(Cont'd)***

Signal	Direction	Width	Description
app_wdf_mask_data	Input	2*MASK_WIDTH	Write data mask from user for write data FIFO. Rising edge mask data = app_wdf_mask_data [DM_WIDTH-1:0] Falling edge mask data = app_wdf_mask_data [2*DM_WIDTH-1:DM_WIDTH]
app_wdf_wren	Input	1	Write data FIFO write enable.
app_af_afull	Output	1	Address/Command FIFO almost full flag. This signal is asserted when only 12 more locations are available in the FIFO.
app_wdf_afull	Output	1	Write data FIFO almost full flag. This signal is asserted when only 12 more locations are available in the FIFO.
rd_data_fifo_out	Output	2*DQ_WIDTH	Read data bus. Data on this bus is valid only when rd_data_valid = 1. Rising edge data = rd_data_fifo_out [DQ_WIDTH-1:0] Falling edge data = rd_data_fifo_out [2*DQ_WIDTH-1:DQ_WIDTH]
rd_data_valid	Output	1	Read data valid strobe.

Figure 26 shows the timing for sending a write command (via the Address/Command FIFO), and the corresponding write data (via the Write Data FIFO) to the DDR2 interface for a burst length of 4.

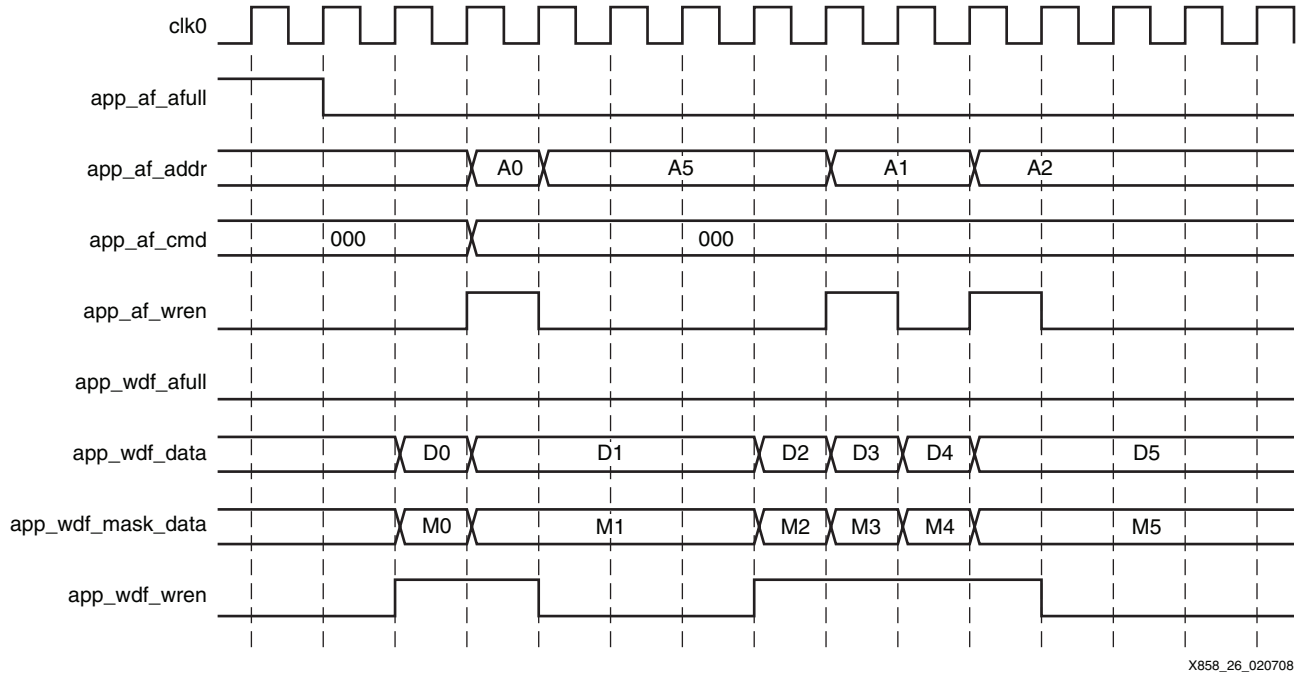


Figure 26: **User Interface Write Command/Data Timing**

Figure 27 shows the timing for sending a read command to the DDR2 controller and the timing for the return of the corresponding read data for a burst length of 8.

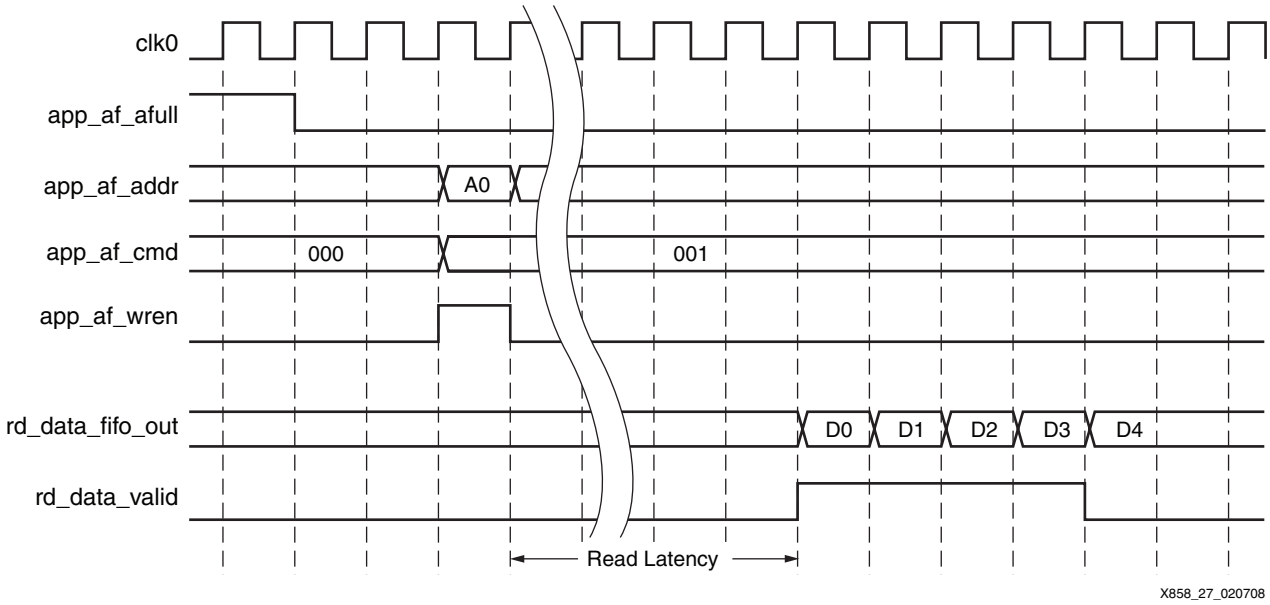


Figure 27: **User Interface Read Command/Data Timing**

2T Timing for Address/ Control

Description

The DDR2 interface design supports optional 2T timing for the address and control signals (with the exception of chip select). This can be used to relax the setup and hold time requirement on heavily loaded address buses (e.g., in the cases of unbuffered DIMM modules where the address bus can go to many loads). In the case of 2T timing, the controller asserts the address and control signals except for chip select one cycle early and keeps them asserted for two clock cycles. The chip select is toggled at the correct time to indicate to the memory when to latch in the address and control signals. 2T timing increases the setup and hold times on the control/address lines at the memory, at the expense of only allowing a new command to be issued every two clock cycles.

2T timing is enabled in the design by setting the TWO_T_TIME_EN top-level parameter to `1`. 2T timing is supported in the both the controller and PHY layer logic.

Timing Diagrams

Figure 28 shows control/address timing for 1T timing.

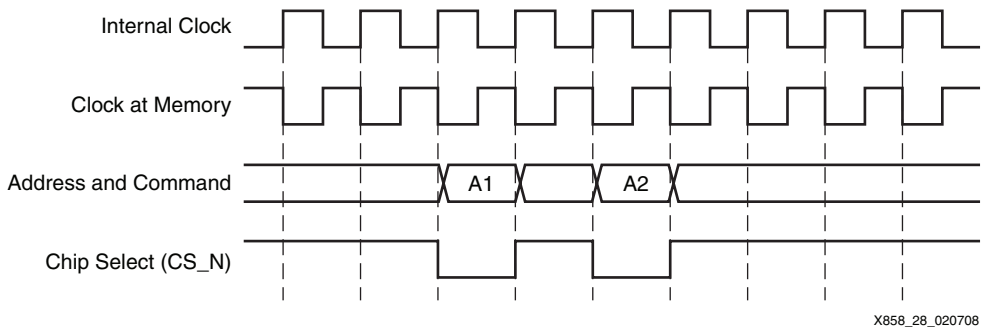


Figure 28: **1T Timing**

Figure 29 shows control/address timing for 2T timing.

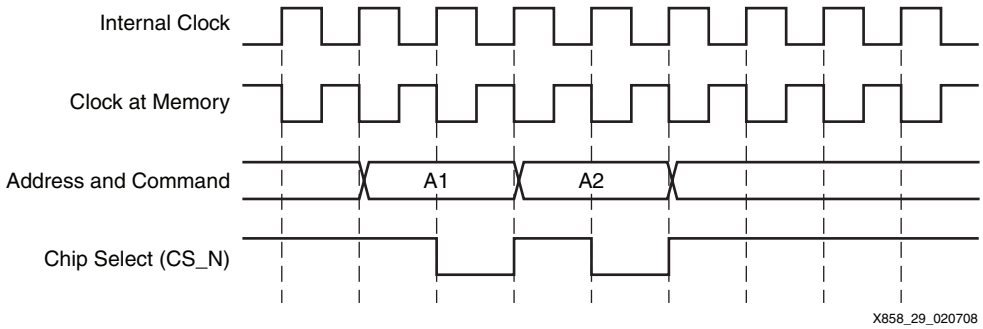


Figure 29: **2T Timing**

Chip Select Timing in High Loading Situations

Timing on the chip select signal becomes critical on heavily loaded DIMMs because the chip select signal remains a 1T signal, even when 2T timing is enabled. In some cases, the loading can be so large that the chip select signal cannot meet the setup time requirements for the next clock edge. In this case, the chip select signal can be asserted one-quarter or one-half clock cycle earlier than is normally done in the DDR2 interface design. By default, CLK0 is used to clock the chip select output flip-flop; instead, CLK270 (inverted version of CLK90), or CLK180 (inverted version of CLK0) can be used to clock the chip select output flip-flop. The DDR2 design must be manually modified to support this. The user must take care that the chip select is not advanced so much that a hold time violation now occurs.

Figure 30 shows the early assertion of chip select.

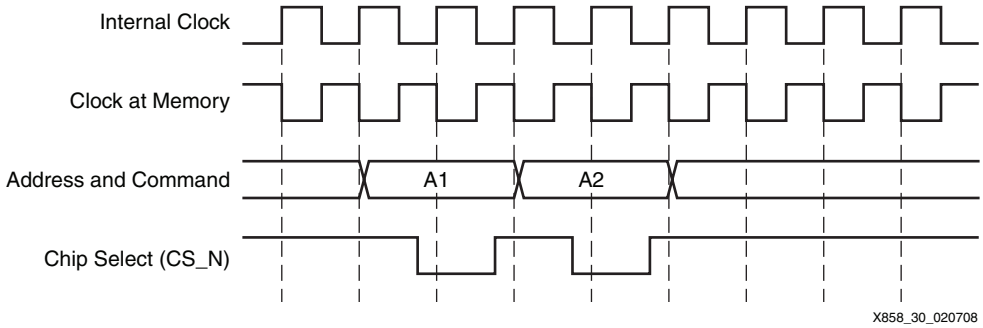


Figure 30: **Early Assertion of Chip Select with 2T Timing**

Error Correcting Code Support

The DDR2 interface design supports optional error correcting code (ECC) support. ECC is supported on a 64-bit boundary. The built-in ECC hardware support in the Virtex-5 block RAM (FIFO36) is used for ECC encode and decode. In the transmit path, the ECC block generates the ECC parity bits using the ECC encode function in the FIFO36. In the receive path, the ECC block in FIFO36 decodes the data and parity bits. The ECC block automatically corrects single bit errors and also indicates an error condition if a single- or two-bit error is detected.

The ECC function is enabled in the design when the top-level HDL parameter ECC_ENABLE is set to `1`. The ECC logic resides in the User Interface block of the DDR2 design.

The ECC error signals are shown in Table 14.

Table 14: **ECC Error Signals**

Signal	Description
rd_ecc_error[0]	This bit in the signal is asserted when a single bit error is detected and corrected by the ECC block. This signal is valid when rd_data_valid signal is asserted.
rd_ecc_error[1]	This bit in the signal is asserted when two bit errors are detected. This signal is valid when rd_data_valid signal is asserted.

The design writes the ECC parity bits to the memory when at least one of the data bytes is not masked. The ECC parity bits are not written to the memory when all the data bytes are masked. When some of the data bytes are masked, the design displays an error in simulation. No error is logged and sent to the user when some of the data bytes are masked out. When partial writes are performed (parts of the data bytes are masked out), an ECC error occurs during reads. The ECC parity generation is always on the entire data bus. On reads to the locations that had partial writes, the ECC parity bits do not match the data on the data bus.

The three additional FIFO36 blocks are added to the User Interface block logic for a 64-bit ECC implementation. One FIFO is added to store the data mask data during writes, and two more FIFOs are added in the receive path to decode the read data from memory. In the receive path, the error signals are logically OR'ed together before being available at the User Interface block. Figure 31 shows the FIFO usage for a 64-bit ECC implementation.

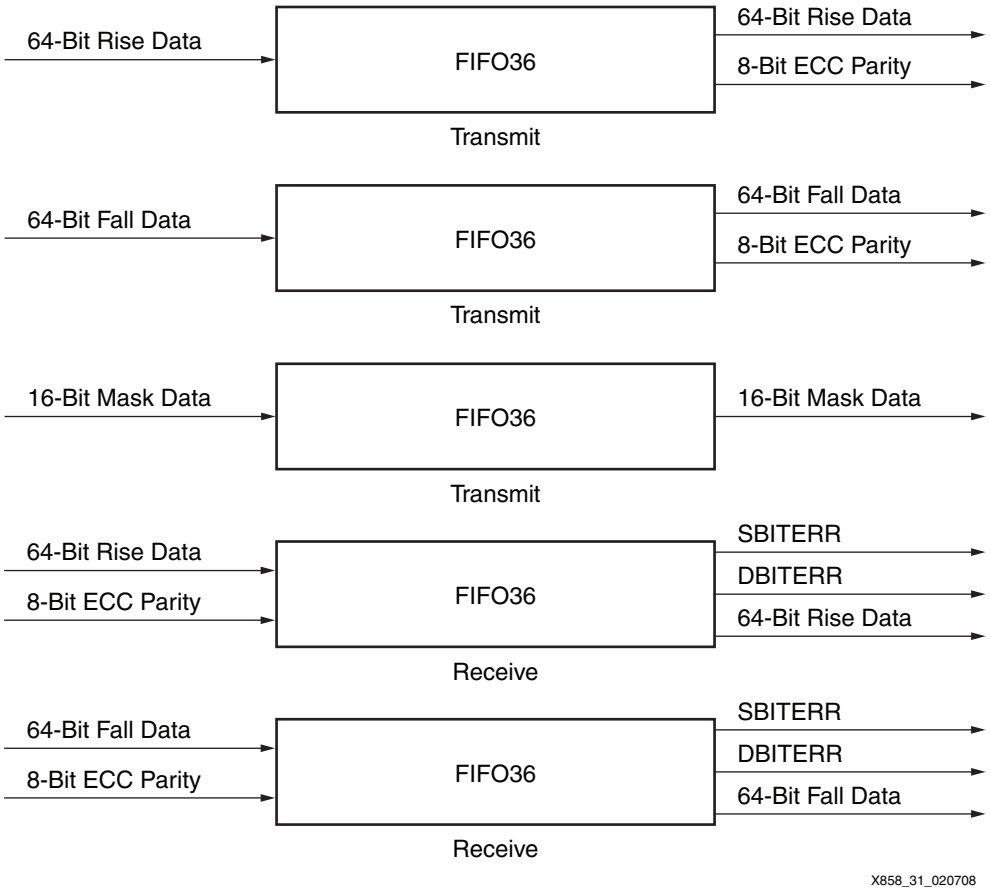


Figure 31: **FIFO Usage for 64-bit ECC Implementation**

Reference Design

The reference design for interfacing Virtex-5 FPGAs to DDR2 SDRAM devices is available both in Verilog and VHDL, and has been integrated with the 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

Reference Design Matrix

Table 15 shows the reference design matrix.

Table 15: **Reference Design Matrix**

Parameter	Description
General	
Developer name	Xilinx
Target Device	Virtex-5 FPGA
Source code provided	Yes, use MIG tool to generate HDL
Source code format	VHDL, Verilog
Design uses code/IP from existing Xilinx application note/reference design, CORE Generator™ software or 3rd party	Yes
Simulation	
Functional simulation performed	Yes
Timing simulation performed	No
Testbench provided for functional/timing simulation	Yes
Testbench format	VHDL, Verilog
Simulator software/version used	ModelSim 6.3
SPICE/IBIS simulations	Yes
Implementation	
Synthesis software tools/version used	XST 10.1, Synplify Pro 8.8.04
Implementation software tools/version used	ISE® Design Suite 10.1
Static timing analysis performed	Yes
Hardware Verification	
Hardware verified	Yes
Hardware platform used for verification	ML561 memory interfaces development board

Design Hierarchy

The HDL code structure for the DDR2 design is shown in Figure 32:

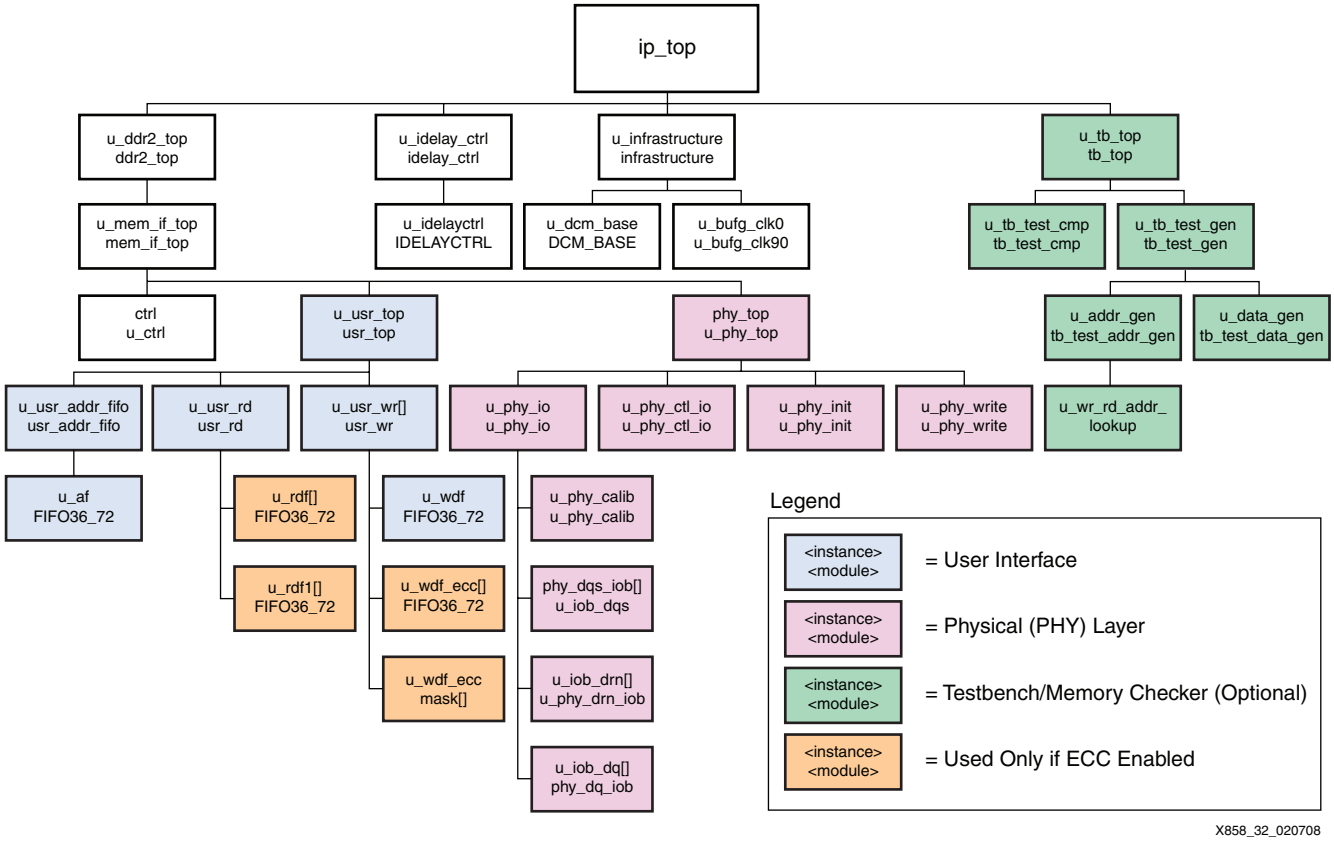


Figure 32: **DDR2 Interface Design HDL Hierarchy**

Parameterization

The DDR2 interface design is provided as parameterizable HDL code. Parameters accessible at the HDL top-level module allow the user to customize the interface for design-specific parameters, such as data bus width, operating frequency, and memory-specific timing parameters without having to modify lower level files.

Table 16 shows the top-level HDL parameters available in the design.

Table 16: **Top-level Parameters**

Category	Parameter Name	Description	Other Notes	Allowed Value
Memory Width	BANK_WIDTH	Number of memory bank address bits		
	CKE_WIDTH	Number of memory clock enable outputs		
	CLK_WIDTH	Number of differential clock outputs	Determined by number of components/modules (one pair per component).	
	COL_WIDTH	Number of memory column bits		
	CS_BITS	Log_2(CS_NUM) rounded up to nearest integer	Used for chip-select related address decode. See "Other Notes," page 42 for CS_NUM and CS_WIDTH.	
	CS_NUM	Number of ranks of CS signals	This number indicates the number of ranks or depth of the memory controller.	The only supported value is 1
	CS_WIDTH	Number of physical CS output signals	This is the number of CS output pins from the controller. One output per component for a component interface is recommended. Use one output per DIMM CS for the DIMM interface. For example, in a 32-bit interface created with 2 x 16 components, set CS_NUM to 1 and CS_WIDTH to 2. With these settings, one internal signal drives two output pins.	CS_WIDTH ÷ CS_NUM must be an integer
	DM_WIDTH	Number of data mask bits	Can be different value than DQS_WIDTH if x4 components are used.	= (DQS_WIDTH * DQ_PER_DQS)/8
	DQ_WIDTH	Number of data bits	Must set to DQS_WIDTH * DQ_PER_DQS. Equal to total number of data bits, including ECC bits.	= (DQS_WIDTH * DQ_PER_DQS)
	DQ_BITS	Log_2(DQ_WIDTH) rounded up to nearest integer	Used for data bus calibration decode.	
	DQ_PER_DQS	Number of memory DQ data bits per strobe		
	DQS_BITS	Log_2(DQS_WIDTH) rounded up to nearest integer		
	DQS_WIDTH	Number of memory DQS strobes		
	ODT_WIDTH	Number of ODT control outputs	Determined by number of components/modules (one per component).	

Table 16: **Top-level Parameters *(Cont'd)***

Category	Parameter Name	Description	Other Notes	Allowed Value
Memory Width	ROW_WIDTH	Number of memory address bits		
	APPDATA_WIDTH	Number of data bits at User Backend Interface (= twice memory bus width - ECC bits)		= (DQ_WIDTH*2) if ECC disabled; = 2*(DQ_WIDTH - 8*(DQ_WIDTH/72)) if ECC enabled
Memory Options	ADDITIVE_LAT	Additive latency		= (0-4)
	BURST_LEN	Burst length		= (4,8) for DDR2, = (2,4,8) for DDR
	BURST_TYPE	Burst type (= 0 sequential, =1 interleaved)		= (0,1)
	CAS_LAT	CAS latency		= (3-5) for DDR2 = (2,3,25) for DDR (25 for CL = 2.5)
	ECC_ENABLE	Enable ECC		Set to 0
	MULTI_BANK_EN	Enable bank management	If enabled, up to four banks kept open; otherwise, one bank kept open.	= (0,1)
	ODT_TYPE	ODT termination value	= 0 if ODT disabled = 1 (75Ω) = 2 (150Ω) = 3 (50Ω)	= (0-3)
	REDUCE_DRV	Enable reduced strength memory I/O	Not supported for all DDR/DDR2 widths = 0 for normal drive = 1 for reduced drive	= (0,1)
	REG_ENABLE	Set for registered memory module	Accounts for extra clock cycle delay on address/control for registered module.	= (0,1)
	TWO_T_TIME_EN	Enable 2T timing for control/address signals	= 0 to disable 2T timing = 1 to enable 2T timing	= (0, 1)

Table 16: **Top-level Parameters *(Cont'd)***

Category	Parameter Name	Description	Other Notes	Allowed Value
Memory Timing	TREFI_NS	Auto refresh interval (in ns)	Take directly from appropriate DDR2 SDRAM vendor data sheet.	
	TRAS	Active->precharge delay (in ps)	Take directly from appropriate DDR2 SDRAM vendor data sheet.	
	TRCD	Active->read/write delay (in ps)	Take directly from appropriate DDR2 SDRAM vendor data sheet.	
	TRFC	Refresh->refresh, refresh->active delay (in ps)	Take directly from appropriate DDR2 SDRAM vendor data sheet.	
	TRP	Precharge->command delay (in ps)	Take directly from appropriate DDR2 SDRAM vendor data sheet.	
	TRTP	Read->precharge delay (in ps)	Take directly from appropriate DDR2 SDRAM vendor data sheet.	
	TWR	Used to determine write->precharge (in ps)	Take directly from appropriate DDR2 SDRAM vendor data sheet.	
	TWTR	Write->read (in ps)	Take directly from appropriate DDR2 SDRAM vendor data sheet.	

Table 16: **Top-level Parameters *(Cont'd)***

Category	Parameter Name	Description	Other Notes	Allowed Value
Miscellaneous	CLK_PERIOD	Memory clock period (in ps)	Used for PHY calibration and DCM setting (if applicable).	
	DEBUG_EN	Enable debug features of code	Used for debug of calibration - visibility to IDELAY results and other calibration results. Also allows setting of IDELAY taps after calibration.	= (0, 1)
	DLL_FREQ_MODE	DCM Frequency Mode	Determined by CLK_PERIOD. Needed only if DCM option is selected.	= ("LOW", "HIGH")
	DDR_TYPE	Select either DDR, DDR2, or DDR3 interface.	Must set to `1` for DDR2. This parameter is not available at the top-level module.	= (1)
	DQS_IO_COL	Placement parameter specifying I/O column locations for each DQS in interface.	Set according to each DQS I/O column location: `00` = left `01` = center `10` = right For more details, refer to Appendix B of the *MIG User Guide* [Ref 3].	Array size = 2 * DQS_WIDTH. Each array element must be = (`00`, `01`, `10`)
	DQ_IO_MS	Placement parameter specifying master/slave I/O placement for each DQ in interface.	Set according to each DQ I/O location: 0 = Slave I/O used 1 = Master I/O used For more details, refer to Appendix B of the *MIG User Guide*.	Array size = DQ_WIDTH. Each array element must be = (`0`, `1`)
	RST_ACT_LOW	Polarity of reset.	Set to `1` for active-Low reset, to `0` for active-High reset	= (`0`, `1`)
	SIM_ONLY	Enable to bypass initial 200 µs power-on delay, and a portion of initial calibration/training sequence.	Only set to `1` for simulation only. For the actual synthesized design, this must be set to `0`.	= (`0`,`1`)

Resource Utilization

Overall Controller Resource Utilization

Table 17 lists FPGA resource utilization for the DDR2 interface. Two different designs are shown: a 32-bit component design, and a 64-bit registered DIMM design. Both designs were synthesized with the testbench/memory checker module available with the MIG DDR2 design, and do not include any additional backend logic. The ISE software includes XST, which synthesizes VHDL, Verilog, or mixed language designs to create Xilinx-specific netlist files known as NGC files.

These build parameters were used to generate both designs:

- Target device: XC5VLX50T-3FF1136
- Target frequency: 333 MHz
- Synthesis tool: XST
- ISE software version: 9.2.03i

Table 17: **Resource Utilization for MIG 2.0 DDR2 Interface**

Resource	32-Bit Component	64-Bit RDIMM
Slice Flip-Flops	2108	2967
Occupied Slices	974	1620
Slice LUTs	1548	1936
Block RAM	3	4
IOB	80	128
BUFIO	4	8
BUFG	4	4

Board Design Considerations

Board-level implementation guidelines must be carefully followed for:

- Pin assignments
- Board layout (e.g., termination and trace matching)

Board-level considerations for the DDR2 reference design are discussed in detail in Appendix A of the *MIG User Guide* [Ref 3].

Design Verification

The ML561 Memory Interfaces Development Board is the main hardware platform used for verification of the DDR2 reference design. Detailed information on the ML561 board is available in the *Virtex-5 ML561 Memory Interfaces Development Board User Guide* [Ref 4].

Physical Layer Debug Port

The DDR2 interface design contains optional logic to allow debug and monitoring of the physical layer read timing calibration logic and timing. This port consists of signals brought to the top-level HDL from the PHY_CALIB module (where the read timing calibration logic resides). These signals provide information in debugging hardware issues when calibration does not complete or read timing errors are observed in the system even after calibration completes. They also allow the user to adjust the read capture timing by adjusting the various IDELAY elements used for data synchronization.

Specifically the Debug Port allows the user to:

- Observe calibration status signals.
- Observe current values for IDELAYs used for read data synchronization.
- Dynamically vary these IDELAY values. Possible uses of this functionality include:
  - Debug read data corruption issues
  - Support periodic readjustment of the read data capture timing by adjusting the IDELAY values
  - Use as a tool during product margining to determine actual timing margin available on read data capture

Appendix D of the *MIG User Guide* discusses the Physical Layer Debug Port in detail.

Required UCF and HDL Modifications for Pinout Changes

This DDR2 reference design requires a large number of UCF constraints whose values are dependent on the specific pinout of the DQ and DQS bits. In addition, there are two top-level HDL parameters whose values are also pinout dependent: DQS_IO_COL and DQ_IO_COL. These UCF constraints and HDL parameters are not present for DDR2 designs either using an earlier version of this reference design, or generated with MIG 1.73 or earlier versions.

The MIG tool generates a UCF file and HDL code with the correct constraints and top-level parameters based on the pinout. When using the MIG tool, the specific rules and procedures for generating these constraints do not need to be known.

The UCF constraints must be manually generated under these conditions:

- The user has a pinout based on a DDR2 design generated using an older version of the MIG tool (e.g., MIG 1.7) and wants to update the design to the MIG 2.0 or later DDR2 interface.
  - The older MIG-generated pinout is compatible with the MIG 2.0 or later design. However, the user needs to generate the additional constraints that the MIG 2.0 or later design requires.
  - MIG 2.0 and later versions have a slightly different algorithm for selecting the DQ and DM (data mask) sites; they choose different pins for the DM and some of the corresponding DQ pins. Therefore, running MIG 2.0 or later with the same bank pinout selection setting that was used for the original pre-2.0 design does not result in a UCF and HDL top-level file with the same pinout and related constraints/parameters. Some of the DQ and DM pins are allocated to different pins. However, the MIG 2.0 or later UCF can be used as a baseline for modifications.
- A design was generated using MIG 2.0 or later but modifications need to be made to the pinout (e.g., swapping DQ bit locations).
- A design was generated for one device/package combination but needs to be migrated to a different device/package. Even if the new package is pin-compatible with the current package, the constraints must be regenerated.
- A pinout was generated independent of the MIG tool.
  - This is not strongly recommended; the MIG tool should be used to generate the pinout.
  - It is recommended in this case that a UCF file be generated using the MIG tool, and then used as a baseline for constraint modifications.

These additional constraints are required because of changes to the read data capture architecture used in this design; specifically, a combination of the IOB-based IDDR flip-flop and flip-flops located in the CLBs is used to capture read data, rather than the ISERDES. Starting with MIG 2.0, a circuit to gate a glitch on the DQS strobe at the end of read bursts also requires additional constraints.

The Update UCF feature in the MIG tool, available beginning in MIG 2.1, should be used to update a MIG 1.73 or earlier UCF to the MIG 2.0 or later requirements. This feature generates an updated UCF file and a text file showing the correct value for the additional top-level HDL parameters. The *MIG User Guide* [Ref 3] discusses the Update UCF feature. Appendix B of the *MIG User Guide* details the specific steps necessary to generate the HDL and UCF constraints.

References

The following references are used in this application note:

1. Micron MT9HTF6472Y-667 DDR2 SDRAM RDIMM Data Sheet
   http://download.micron.com/pdf/datasheets/modules/ddr2/HTF9C32_64_128x72.pdf
2. Micron MT47H32M16CC-3 DDR2 SDRAM Data Sheet
   http://download.micron.com/pdf/datasheets/dram/ddr2/512MbDDR2.pdf
3. UG086, *Xilinx Memory Interface Generator (MIG) User Guide*
4. UG199, *Virtex-5 ML561 Memory Interfaces Development Board User Guide*

Additional Resources

The following resources provide additional information useful to this application note:

1. DS202, *Virtex-5 FPGA Data Sheet: DC and Switching Characteristics*
2. UG190, *Virtex-5 FPGA User Guide*

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
05/12/06	1.0	Initial Xilinx release.
01/09/07	1.1	Updated link to reference design.
03/13/08	2.0	Major document revision. Physical layer, controller, and reference design descriptions updated to reflect MIG 2.0 design.
05/08/08	2.1	• Added Table 1. • Updated Table 16. • Updated Figure 22. • Updated "Extended Mode Register Definition" section. • Added "Board Design Considerations," "Design Verification," "References," and "Additional Resources" sections.
09/14/10	2.2	Updated DQ and DM signals in Figure 3. Updated fixed data pattern for DQS groups in "Stage 3 Read Timing Calibration: Read Data Valid Calibration" and "Stage 4 Read Timing Calibration: DQS Gate Calibration." In "Capturing the Last Data of Read Burst," added note about external termination and replaced "reset" with "set" in description of IDELAY output. Replaced AR with AS in lower IDDR flip-flop in Figure 19.

Notice of Disclaimer