Xilinx Memory Interface Generator (MIG) 1.5 User Guide

DDR SDRAM, DDRII SRAM, DDR2 SDRAM, QDRII SRAM, and RLDRAM II Compilers

UG086 (v1.5) February 15, 2006

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/01/04	1.0	Initial MIG 1.0 release.
01/01/05	1.1	MIG 1.1 release.
05/01/05	1.2	MIG 1.2 release.
08/18/05	1.3	MIG 1.3 release.
11/04/05	1.4	MIG 1.4 release.
02/15/06	1.5	MIG 1.5 release.

Table of Contents

SECTION 1: VIRTEX-4 FPGA TO MEMORY INTERFACES

Chapter 2: Implementing DDR SDRAM Controllers

Chapter 3: Implementing DDR2 SDRAM Controllers

Chapter 4: Implementing QDRII SRAM Controllers

Chapter 5: Implementing DDRII SRAM Controllers

Chapter 6: Implementing RLDRAM II Controllers

SECTION 2: SPARTAN-3 FPGA TO MEMORY INTERFACES

Chapter 7: Implementing DDR SDRAM Controllers

Chapter 8: Implementing DDR2 SDRAM Controllers

Appendix A: Memory Implementation Guidelines

About This Guide

The Memory Interface Generator (MIG) 1.5 tool generates DDRII SRAM, DDR SDRAM, DDR2 SDRAM, QDRII SRAM, and RLDRAM II interfaces for Virtex™-4 FPGAs. It also generates DDR and DDR2 SDRAM interfaces for Spartan™-3 FPGAs and DDR SDRAM interfaces for Spartan-3E FPGAs. The tool takes inputs such as the memory interface type, FPGA family, FPGA devices, frequencies, data width, memory mode register values, and so forth, from the user through a graphical user interface (GUI). The tool generates RTL, SDC, UCF, and document files as output. RTL or EDIF (EDIF is created after running a script file, where the script file is a tool output) files can be integrated with other design files.

Guide Contents

This manual contains the following chapters:

- Chapter 1, "Using the MIG 1.5 Tool," shows how to install and use the MIG 1.5 design tool.

- Section 1: "Virtex-4 FPGA to Memory Interfaces"
  - Chapter 2, "Implementing DDR SDRAM Controllers," describes how to implement DDR SDRAM interfaces that the MIG 1.5 tool creates for Virtex-4 FPGAs.
  - Chapter 3, "Implementing DDR2 SDRAM Controllers," describes how to implement DDR2 SDRAM interfaces that the MIG 1.5 tool creates for Virtex-4 FPGAs.
  - Chapter 4, "Implementing QDRII SRAM Controllers," describes how to implement QDRII SRAM interfaces that the MIG 1.5 tool creates for Virtex-4 FPGAs.
  - Chapter 5, "Implementing DDRII SRAM Controllers," describes how to implement DDRII SRAM interfaces that the MIG 1.5 tool creates for Virtex-4 FPGAs.
  - Chapter 6, "Implementing RLDRAM II Controllers," describes how to implement RLDRAM II interfaces that the MIG 1.5 tool creates for Virtex-4 FPGAs.
- Section 2: "Spartan-3 FPGA to Memory Interfaces"
  - Chapter 7, "Implementing DDR SDRAM Controllers," describes how to implement DDR SDRAM interfaces that the MIG 1.5 tool creates for Spartan-3 FPGAs.
  - Chapter 8, "Implementing DDR2 SDRAM Controllers," describes how to implement DDR2 SDRAM interfaces that the MIG 1.5 tool creates for Spartan-3 FPGAs.

- Appendix A, "Memory Implementation Guidelines," provides helpful rules for reference designs.

References

The following documents provide supplementary material useful with this user guide.

1. Samsung Data Sheet k7i321884m_R04
   http://www.samsung.com/Products/Semiconductor/SRAM/SyncSRAM/DDRII_CIO_SIO/36Mbit/K7I321884M/K7I321884M.htm
2. Micron Data Sheet MT47H16M16FG-37E
   http://www.micron.com/products/dram/ddr2sdram/partlist.aspx
3. Samsung Data Sheet k7r323684m
   http://www.samsung.com/Products/Semiconductor/common/product_list.aspx?family_cd=SRM020302
4. Micron Data Sheet MT49H16M18FM-25
   http://www.micron.com/products/dram/rldram/part.aspx?part=MT49H16M18FM-25
5. Micron Data Sheet MT46V16M16FG-5B
   http://www.micron.com/products/dram/ddrsdram/partlist.aspx
6. UG070, *Virtex-4 User Guide*
7. UG072, *Virtex-4 PCB Designer's Guide*
8. UG079, *Virtex-4 ML461 Memory Interfaces Development Board User Guide*
9. XAPP645, *Single Error Correction and Double Error Detection*
10. XAPP701, *Memory Interfaces Data Capture Using Direct Clocking Technique*
11. XAPP702, *DDR-2 Controller Using Virtex-4 Devices*
12. XAPP703, *QDR II SRAM Interface*
13. XAPP709, *DDR SDRAM Controller Using Virtex-4 FPGA Devices*
14. XAPP710, *Synthesizable CIO DDR RLDRAM II Controller for Virtex-4 FPGAs*
15. XAPP721, *High-Performance DDR2 SDRAM Memory Interface Data Capture Using ISERDES and OSERDES*
16. XAPP768c: *Interfacing Spartan-3 Devices With 166 MHz or 333 Mb/s DDR SDRAM Memories* (available under click license)
17. DS099, *Spartan-3 FPGA Family: Complete Data Sheet*
18. DS312, *Spartan-3E FPGA Family: Complete Data Sheet*

Additional Resources

To search the database of silicon and software questions and answers, or to create a technical support case in WebCase, see the Xilinx website at:
http://www.xilinx.com/support.

Typographical Conventions

This document uses the following typographical conventions. An example illustrates each convention.

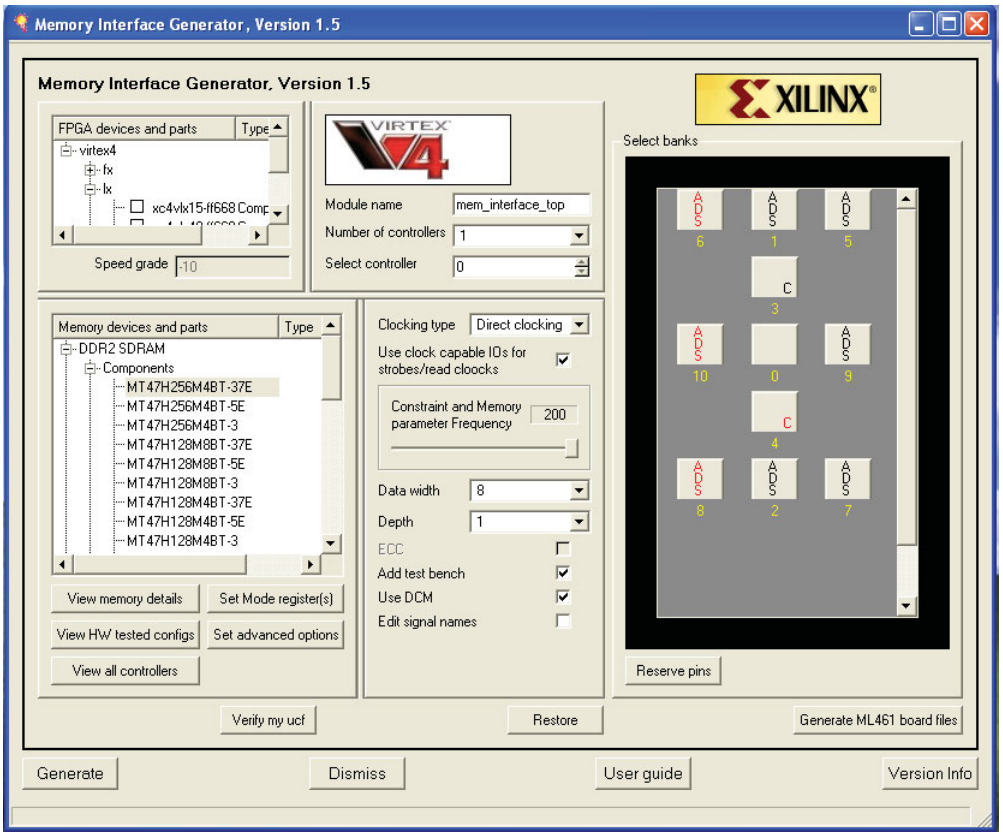
Convention	Meaning or Use	Example
Italic font	References to other documents	See the Virtex-4 *Configuration Guide* for more information.
	Emphasis in text	The address (F) is asserted *after* clock event 2.
Underlined Text	Indicates a link to a web page.	http://www.xilinx.com/virtex4

Chapter 1

Using the MIG 1.5 Tool

The MIG 1.5 tool is used to generate memory interfaces for Xilinx FPGAs. This document describes the user interface and various options in the MIG 1.5 tool. The output files are also described.

Figure 1-1 is a screen shot of the MIG 1.5 GUI.



UG086_c1_01_02806

Figure 1-1: **MIG 1.5 GUI**

MIG 1.5 Tool Changes from MIG 1.4

The new features of the MIG 1.5 design tool are summarized in this section:

- GUI Changes
  - Clock capable I/Os for strobes and read clocks for direct clocking method
  - Programmable Mode Register options
  - Verify my UCF feature
  - Programmable pin allocation limit for selected banks
  - Reserved Pin list
  - Save option to a file
- DDR2 SDRAM Direct clocking (Virtex-4 interfaces) support:
  - Synplicity Synplify 8.2 support
  - SO DIMM support
  - Modified Read Enable implementation
- ISE 8.1.01i support (all MIG 1.5 designs support this ISE version)
- DDR2 SDRAM SerDes clocking (Virtex-4 interfaces) support
- DDR SDRAM for Virtex-4 interfaces:
  - Synplicity Synplify 8.2 support
  - CL = 2, 2.5, and 4
  - BL = 2 and 8
  - SO DIMMs
  - Support for more memory devices
  - Modified Read Enable implementation
- DDR SDRAM for Spartan-3/Spartan-3E devices:
  - CL = 2 and 2.5
  - BL = 2 and 8
  - Synplicity Synplify 8.2
  - Registered DIMMs
  - Support for more memory devices
- DDR2 SDRAM for Spartan-3 devices:
  - Synplicity Synplify 8.2
  - BL = 8
  - Registered DIMMs
- RLDRAM II:
  - Synplicity Synplify 8.2 support
- QDRII and DDR2 SRAMs:
  - Synplicity Synplify 8.2 support
- Supports skip wait 200 µs delay for Verilog simulations. This feature is not supported for VHDL cases.
  - To skip 200 µs initial delay, users should use the following run-time options for Verilog in ModelSim.

- For DDR SDRAM for Virtex-4 interfaces:

  ```
  vlog +define+simulation modulename_ddr_controller_0.v
  ```

  Where:

  - `simulation` is the parameter.
  - `modulename_ddr_controller.v` is the file with the parameter 'simulation'. The file `modulename_ddr_controller.v` must be present in the `sim` folder.
- For DDR2 SDRAM for Virtex-4 interfaces:

  ```
  vlog +define+simulation modulename_ddr2_controller_0.v
  ```
- For Spartan-3 interfaces:

  ```
  vlog +define+simulation modulename_ddr_infrastructure_top.v
  ```

Tool Features

The key features of the MIG 1.5 design tool are listed below:

- Supported memory types for Virtex™-4 interfaces:
  - DDR2 SDRAM components, and registered and unbuffered DIMMs. The DDR2 controller supports deep memory depths from one to four.
  - DDR SDRAM components, and registered and unbuffered DIMMs
  - QDRII and DDRII SRAM
  - RLDRAM II CIO and SIO memories
- Supported memory types for Spartan-3 interfaces:
  - DDR SDRAM components, registered and unbuffered DIMMs
  - DDR2 SDRAM components, registered and unbuffered DIMMs
- Supported memory types for Spartan-3E interfaces:
  - DDR SDRAM components, registered and unbuffered DIMMs
- Supported Virtex-4 Synthesis tools:
  - XST and Synplicity Synplify 8.2
- Supported Spartan-3 Synthesis tools:
  - Synplicity, XST, and Precision 2005B for the DDR SDRAM Local clocking type
- Supported Spartan-3E Synthesis tools:
  - XST and Synplicity Synplify 8.2
- Supported components:
  - For DDR2 SDRAM, RLDRAM II SDRAM, and QDRII and DDRII SRAMs, most of the popular components and DIMMs are supported.
  - Some DDR SDRAM components and DIMMs are supported.
  - An equivalent component can be found for any unsupported component.
- All currently available Virtex-4, Spartan-3, and Spartan-3E FPGAs are supported.
- Direct and SerDes clocking techniques for data capture for Virtex-4 interfaces.

  Direct clocking is explained in XAPP701 [Ref 10]. With this technique, it is not necessary to use clock-capable I/Os for strobes or read clocks. SerDes clocking is explained in XAPP721 [Ref 15]. The use of clock-capable I/Os for strobes and read

clocks is recommended for maximum flexibility with higher frequency designs (200 MHz and above).

- Local clocking technique for data capture for all Spartan-3 and Spartan-3E interfaces.

  The data capture technique using Spartan-3 FPGAs is explained in XAPP768c [Ref 16].
- Both VHDL and Verilog are supported.
- Variable data widths in multiples of 8 up to 144 bits.

  The actual width depends upon the selected component. For a 16-bit wide component, data widths of 16, 32, 48, and 64 are supported. For an 8-bit wide component, data widths of 8, 16, 24, 32, 40, 48, 56, 64, and 72 are supported.

  For DDR2 SDRAM, most of the components support up to a 144-bit data width.
- User-selectable banks for address, data, system control, and system clock signals.

  For QDRII and RLDRAM II memories, the user selects the data banks for reads and RLD (SIO) writes separately.
- Different banks are supported with different I/O standards.

  The MIG 1.5 tool uses different banks for groups of signals whose I/O standards are different. If the I/O voltages for different groups (such as address, data, and system control) are different, the user must ensure enough banks are selected for the MIG 1.5 tool to use. If insufficient banks are selected, the tool cannot allocate pins.
- Various configurations are supported through changing bits in the Mode and Extended Mode Registers.
- All fields not highlighted in the GUI either are not supported or are not relevant for that type of memory.
- Only one type of component is supported per interface.

  Users cannot mix different components to create an interface.
- Multiple DDR2 interfaces.

  Users can select up to eight controllers.
- Pin compatibility.

  Users can select multiple devices using the same package to generate compatible pinouts.
- Supported Operating Systems:

  The MIG tool is available only on these ISE platforms:

  - Windows XP Home
  - Professional

Xilinx Design Tools Version Requirements

To use this IP update, ISE 8.1i with Service Pack 1 (8.1.01i) must be installed. ISE 8.1i service packs can be obtained from the Download Center at:
http://www.xilinx.com/xlnx/xil_sw_updates_home.jsp

The MIG 1.5 tool requires ISE 8.1i IP Update 1 or later. If not already installed, IP Update 1 is automatically installed by the Updates Installer before the ISE 8.1i MiG 1.5 IP Update is installed. See Xilinx Answer 21938 for issues related to ISE 8.1i IP Update 1. Acrobat Reader Version 5 or later must be installed. The latest Acrobat software can be downloaded from the Adobe website at: http://www.adobe.com/products/acrobat/readstep.html.

Installation

The MIG tool is accessible via the Xilinx CORE Generator™ System, starting with MIG version 1.3.

There are two ways to install the MIG 1.5 toolset.

Method 1

Users who are behind a firewall and do not know their proxy settings use this method.

1. Users must verify that they have the latest ISE 8.1i Service Pack and the latest IP Update from the Download Center at: http://www.xilinx.com/xlnx/xil_sw_updates_home.jsp. For ISE, users receive a proper installer. For the IP Update, users receive a ZIP file to unzip into the 8.1i installation. For the MIG 1.5 tool, ISE sp1 and the latest IP Update are required.
2. Users who are not registered at the Memory Corner can register at: http://www.xilinx.com/xlnx/xil_entry2.jsp?sMode=login&group=memory_customers
3. The MIG 1.5 tool is downloaded from: http://www.xilinx.com/support/software/memory/protected/ise_81i_mig15.zip

   (Enter the xilinx.com account name and password when prompted.)
4. This ZIP file is unzipped into the root ISE 8.1i installation (`C:\Xilinx` by default).

Method 2

1. The CORE Generator system is launched by selecting **Start -> Xilinx ISE 8.1i -> Accessories -> CORE Generator** from the Windows Start menu.
2. When the CORE Generator GUI opens, select **Tools -> Updates Installer**.
3. The CORE Generator system displays a dialog box with a warning indicating that it will exit after the installation is complete. Click the **Accept** button.
4. The CORE Generator system connects to www.xilinx.com and might ask for the user xilinx.com User ID and password. Users who are behind a firewall might have to enter their appropriate proxy settings.
5. The IP Updates Installer dialog box opens and displays a panel listing the available updates.
6. Select ISE 8.1i MIG 1.5 and click the **Install Selected** button. The program might indicate that other installs are required. These informational messages can be accepted. The CORE Generator system downloads and installs the requested products and exits.

Note: The installation process must not be interrupted. During this process, various pop-up messages must be accepted. If other windows are open, the pop-ups might be hidden behind them.

Getting Started

These steps launch the MIG tool:

1. The CORE Generator system is launched by selecting **Start -> Xilinx ISE 8.1i -> Accessories -> CORE Generator**.
2. Create a CORE Generator project.
3. The Xilinx part must be correctly set because it cannot be changed inside the MIG tool. Virtex-4 and Spartan-3/Spartan-3E devices are supported. Select the part via the part's Project Options menu in the CORE Generator system. The generation tab is used to

select between Verilog or VHDL by "design entry" under "flow". The "flow settings" "vendor" are chosen appropriately.

4. Remember the location of the CORE Generator project directory. The "View by Function" tab to the left shows the available cores organized into folders.
5. The MIG tool is launched by selecting **Memories & Storage Elements -> MiG -> Memory Interface Generator**.
6. The name of the module to be generated is entered in the Module Name text box. Click **Generate** to generate the module files in a directory with the same name as the module name in the CORE Generator project directory.
7. After generation, the GUI is closed by selecting the **Dismiss** button.

The "Generated IP" tab to the left lists the generated modules. The generated `ise_flow.bat` script or the ISE GUI is used to manually add the generated HDL files to a project.

User Options

This section summarizes the selectable parameters accessible through the GUI.

Part Selection – Target and Compatible Part

The CORE Generator system selects the part before invoking the MIG tool. The compatible part can be selected from the MIG tool.

Clocking Type

The Clocking Type browsing window selects the desired clocking type for the design. DDR2 SDRAM supports all three types of clocking: *direct* and *SerDes* clocking types (for Virtex-4 devices) and *local* clocking types (for Spartan-3 and Spartan-3E devices). The MIG 1.5 tool enables the corresponding clocking types when memory component is selected.

Use CC (for Direct Clocking)

This option selects clock capable I/Os for strobes and read clocks. Only Virtex-4 Direct clocking designs support this feature.

Frequency

The frequency bar selects the desired interface frequency. The desired frequency is limited by factors such as speed grade of the device, the selected memory part, and the design.

The MIG 1.5 tool does not limit the frequency based on CAS latency. For example, DDR SDRAM memory supports a maximum frequency of 167 MHz for CL = 2, but the MIG 1.5 tool can go up to 200 MHz for CL = 2. The user must select the frequency according to the CAS latency.

For Spartan-3E designs, the tool-generated frequency is more than what the stepping 0 devices support. Users must use Spartan-3E stepping 1 devices for frequencies from 90 MHz to 133 MHz.

Memory

The supported memory components and DIMMs are displayed in this section. In the current tool version, most DDR and DDR2 SDRAM, QDRII and DDR2 SRAM, and RLDRAM II devices are supported. The MIG 1.5 tool automatically identifies the different memory parameters based on the user's selection. The different memory parameters, such as data bits per strobe and address widths, are displayed when the user clicks on the **Memory details** button.

Data Width and Depth

The user selects the required data width through this menu. The data width selection is overridden by the DIMM selection. In other words, if the user selects a DIMM module, the data width is automatically determined by the width of the DIMM module. The supported data widths depend upon the memory device as well. For instance, it is not possible to create a 72-bit interface using 16-bit wide components. Currently, interfaces cannot consist of mixes of different components.

Depth is relevant only for DDR2 SDRAM devices. The DDR2 SDRAM controller generates different depths. Also, signal integrity is a key concern in deep-memory designs. Users must verify the signal integrity of their boards.

View Memory Details

This option opens a pop-up window to view the details of the selected memory.

View HW Tested Configs

This option displays the hardware tested configurations for that design.

Set Mode Register(s)

This option allows users to select various memory options from the Mode and Extended Mode Registers, such as CAS latency and burst length.

Set Advanced Options

This button has different functionality for Spartan-3 designs and Virtex-4 designs.

For Virtex-4 designs, this button can restrict the number of pins to be allocated in a bank. On clicking this button, a table pops up with the bank number, number of pins in that bank, and a column to select number of pins to allocate in that bank. By default, the tool allocates all the pins for the interface. This feature is implemented only for Virtex-4 designs.

For Spartan-3 designs, the user clicks on this button to select the number of write pipelines in the design. The MIG 1.5 tool supports 0 to 4 pipelines, where the default is 4 pipelines. When the user selects 0 to 3 pipelines, the remaining pipe stages are to be included in the user interface. For example, if the user selects 0 pipelines from the tool, the write data should be passed through a 4-stage pipeline before it is given to the controller user interface.

View All Controllers

This option shows all the selected parameters for all controllers, either a single controller or multiple controllers.

Design Parameters

This selection allows the user to change the values of the Mode and Extended Mode Registers and other design parameters. This option changes depending upon the selected memory type. The specific memory data sheet contains detailed explanations of the different parameters in the Mode and Extended Mode Registers.

The MIG 1.5 tool supports the direct and SerDes clocking methods for Virtex-4 FPGAs. Direct clocking is explained in detail in XAPP701 [Ref 10]. This method does not require the strobe/read clock to be in a clock capable (_CC) input/output buffer (IOB). Hence, the tool can select any IOB for strobe/read clocks. SerDes clocking is explained in detail in XAPP721 [Ref 15]. Memory interfaces with Spartan-3 FPGAs use the local clocking method.

VHDL and Verilog are supported for all indicated memory interfaces.

The user can choose to have the digital clock manager (DCM) within the generated design or have all the required clocks as inputs to the design. This option is the function of the DCM check box. This option is available for DDR2 SDRAM and DDR SDRAM Spartan-3 designs.

The error correction code (ECC) check box enables generation of ECC along with the code. ECC is supported in the following configurations:

1. 32-bit DDR2 SDRAM with 7 bits of ECC. This option generates a 40-bit design.
2. 64-bit DDR2 SDRAM with 8-bit ECC. This option generates a 72-bit design.
3. 128-bit DDR2 SDRAM with 16-bit ECC. This option generates a 144-bit design.

Only 40-bit, 72-bit, and 144-bit widths enable the ECC check box. ECC can impact the timing of the design.

The user can chose to have a synthesizable test bench by checking the “test bench” box.

Add Test Bench

When this box is checked, the design is generated along with test bench. The test bench drives the read and write commands, compares the read data, and generates an error signal if the read data does not match the write data.

Use DCM

This option allows the user to generate the design with or without DCM in the design. If the design is generated with the “Use DCM” option enabled, the required clocks are generated from the DCM within the design only, and SYS_CLK and SYS_CLKb are to be provided as inputs to the design. If this option is disabled, users must provide the required clocks as inputs.

Edit Signal Names

When this box is checked, before the design is generated, a table is presented with all the top-level signal names. Users can modify the signal names from this table. The tool outputs the design with the modified signal names.

Output Directory

The desired output directory is selected here. Before clicking the **Generate** button, the user has to select the banks to be used for the data, address, and system control signals (see "Select Banks").

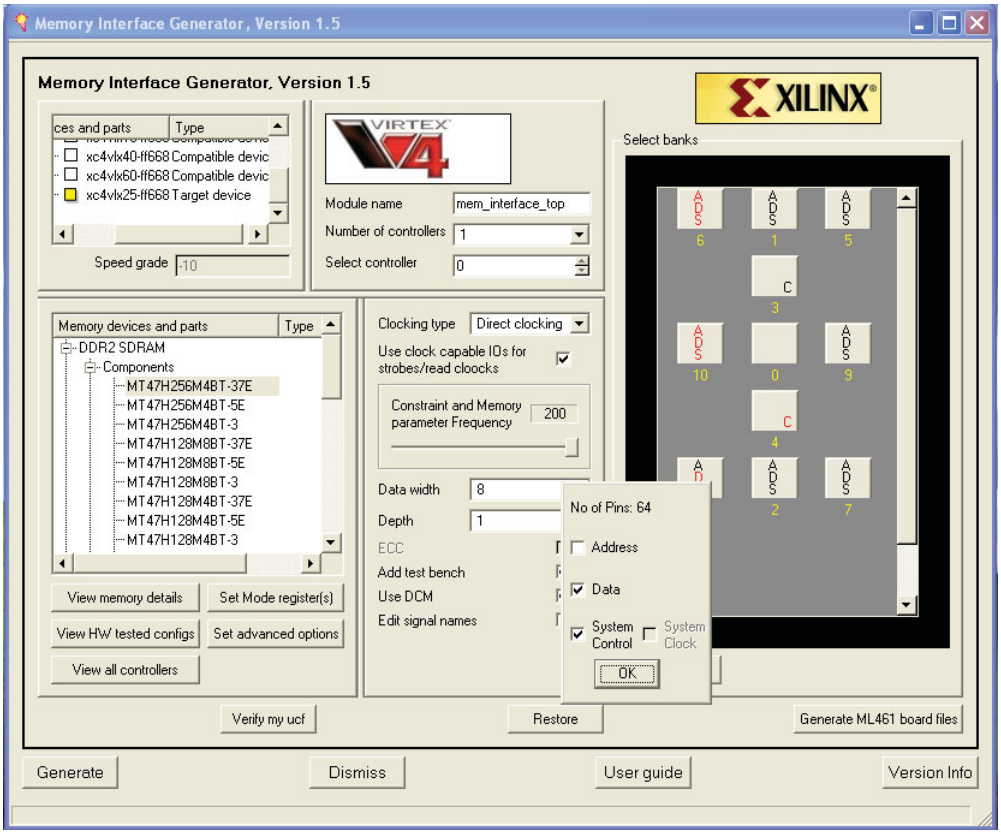
Module Name

The MIG 1.5 tool accepts any name that is prepended to all module and RTL file names of the design.

Select Banks

On the right side of the GUI, an approximate figure appears with the FPGA I/O banks (see Figure 1-2). The user selects the banks to be used for address, data, system control, and system clock signals by right-clicking on the I/O banks in the figure.

The user selects both read and write data banks for QDRII and RLDRAM II SIO devices.



UG086_c1_03_012806

Figure 1-2: **Bank Selection**

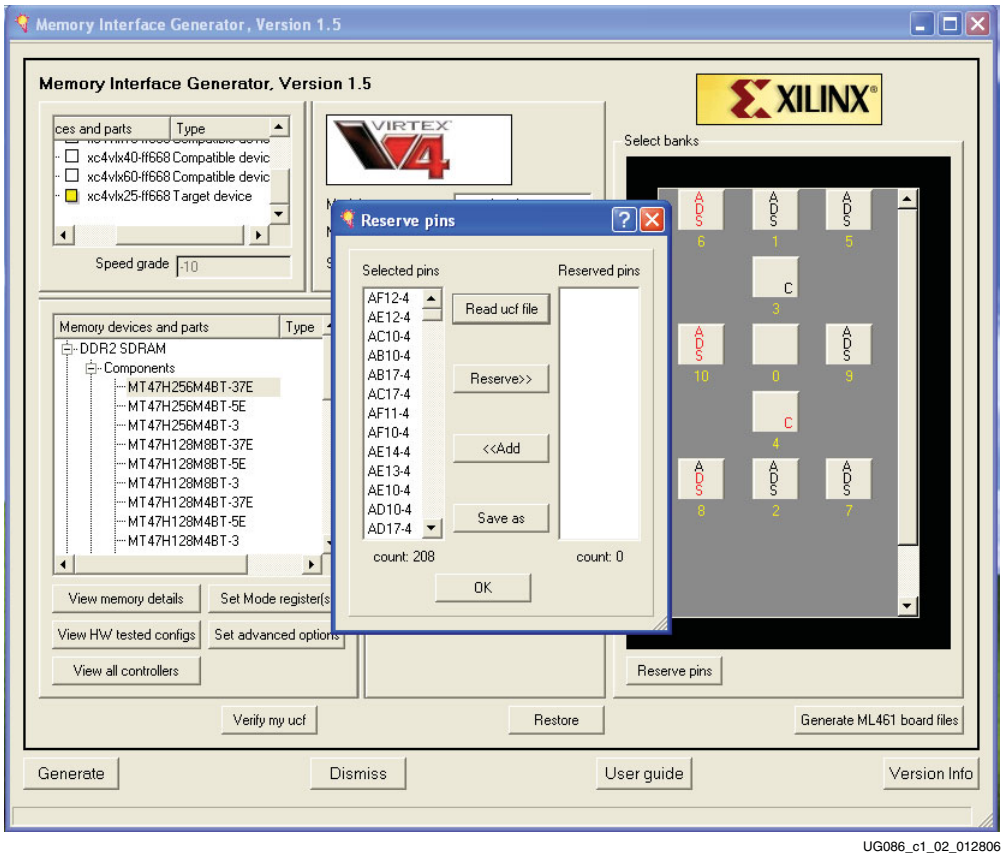
Reserve Pins

The user can reserve some FPGA pins from the selected banks by clicking on the **Reserve pins** button. There is also a provision to read the list of reserved pins from a UCF file using the **read ucf file** button. Any LOC constraints in the UCF file are added to the reserved pin list.

A sample UCF file format looks like:

```
NET "cntrl0_DDR_DQ[3]" LOC = AJ4;
NET "cntrl0_DDR_DQ[2]" LOC = AJ3;
NET "cntrl0_DDR_DQS[1]" LOC = AH3;
```

Figure 1-3 shows the window that appears when the **Reserve pins** button is clicked. The user selects the required pins from the Selected pins list and clicks on the **Reserve** button to reserve the pins. To remove some of the pins from the reserved list, the user selects the pins from the Reserved pins list and clicks on the **Add** button, which moves the pins back to the Selected pins list. The **Save as** button allows the user to save the reserved pins into a user-defined file. When the user clicks on the **Save as** button, a browsing window is opened to select the folder and assign a file name that contains the reserved pins. When the user clicks on the **Read ucf file** button, a browsing window is opened to select the UCF file. All the LOC constraints from the UCF file are considered for the Reserve pins.



UG086_c1_02_012806

Figure 1-3: **Reserve Pins**

Verify my ucf

This feature allows the user to test whether a given UCF file containing signal names and their corresponding pin names comply with the pin allocation rules for the selected options from the tool. The user first inputs all the standard information, such as part/package, banks for address, and data. All the user inputs that are required for generating a UCF/design are required for verification. The verification is performed only for the I/O signals required for the memory design (in other words, other user signals in the UCF are ignored).

This feature verifies:

- Whether all the dq bits are allocated in the selected banks.
- The associated groups are allocated in the same bank. For example dq bits corresponding to a dqs are treated as a group. All the signals within the same group should be in the same bank.
- The I/O banking rules. In a given bank, all the signals should have the I/O standard with the same I/O voltage.
- The signal names
- The selected data width. For example if the data width is 32 bits and the reference UCF has more bits, the tool verifies the required bits and ignores the excess data.
- The uniqueness of the pins. It flags an error if two signals are allocated to the same pin or vice versa, or if the same signal is allocated to more than one pin.
- The strobe signals are allocated to the CC pins when the CC pins option is enabled.
- The signals are allocated within the selected banks.

Error Messages

This section describes the different error messages that can be generated when verifying the UCF file.

The reference UCF must follow the MIG naming conventions (refer to the UCF files generated by the MIG tool).

The *Edit signal names* option is not supported for the Verify my ucf mode.

For example, DDR2 SDRAM controller 0 should have cntrl0_DDR2_DQ[0] for data bits, and RLDRAM controller 0 should have cntrl0_RLD2_DQ[0] for data bits.

1. Uniqueness

   If two signals are allocated to the same pins in the reference UCF, an error message is listed in the directed file with a user-assigned name.

   The error message format is “<signal_name1> and <signal_name2> are allocated to the same pins.”

   For example, if cntrl0_DDR2_DQ[0] and cntrl0_DDR2_DQS[0] are allocated to the same pin, such as:

   ```
   NET  "cntrl0_DDR2_DQ[0]"              LOC = "D12" ;
   NET  "cntrl0_DDR2_DQS[0]"             LOC = "D12" ;
   ```

   Then the following error message is printed:

   ```
   ERROR: cntrl0_DDR2_DQ[0] and cntrl0_DDR2_DQS[0] are allocated to the
   same pins.
   ```

   Pins are not unique.

2. Association

   Signals in the same group (for example, assume DQS[0] and DQ[0:7] form the same group) should go to the same bank, otherwise an error message is printed in the same user directed file.

   The error message format is "<signal_name1> and <signal_name2> are not allocated in the same banks."

   For example:

```
NET  "cntrl0_DDR2_DQ[0]"                LOC = "D12" ; #bank 6
NET  "cntrl0_DDR2_DQ[1]"                LOC = "C12" ; #bank 6
NET  "cntrl0_DDR2_DQ[2]"                LOC = "B10" ; #bank 6
NET  "cntrl0_DDR2_DQ[3]"                LOC = "C10" ; #bank 7
```

   Assume cntrl0_DDR2_DQ[3] and cntrl0_DDR2_DQ[2] are allocated to pins of different banks, such as bank 7 and bank 6, respectively. The following error messages are printed:

```
ERROR: cntrl0_DDR2_DQ[0](6) and cntrl0_DDR2_DQ[3](7) are not allocated
in the same banks
ERROR: cntrl0_DDR2_DQ[1](6) and cntrl0_DDR2_DQ[3](7) are not allocated
in the same banks
ERROR: cntrl0_DDR2_DQ[2](6) and cntrl0_DDR2_DQ[3](7) are not allocated
in the same banks
```

   These types of error messages are printed for each pair of signals of same group, but are allocated to different banks.

3. I/O Voltage

   The signals allocated to the same bank should have the same I/O voltage. If not, an error message is printed in the same user directed file.

   The error message format is "<signal_name1> and <signal_name2> have different I/O Voltages."

   For example, cntrl0_DDR2_DQ[2] and cntrl0_DDR2_DQ[4] are allocated to the same bank (expected because they are from the same group). If these pins have different I/O voltages, the following error message is printed:

```
ERROR: cntrl0_DDR2_DQ[2] and cntrl0_DDR2_DQ[4] have different I/O
Voltages.
```

4. Clock Capable I/Os for strobes/read clock

   Check for CC pins if Use CC for direct clocking is clicked. In this case, the strobe/read_clock signals should be allocated to the CC pins only. If not, an error message is displayed.

   The error message format is "<signal_name> should be allocated to the CC Pins."

   For example, cntrl0_DDR2_DQS[0] is a strobe. Assume it is allocated to the K12 pin, which is not a clock capable I/O pin. The following error message is printed:

```
ERROR: cntrl0_DDR2_DQS[0 should be allocated to the CC Pins.
```

5. Absence of signals

   If one or more signal-pin pair is missing and/or commented in the given UCF file against the selected inputs, the verification result indicates the absence of those signal-pin pairs as a warning.

   The warning message format is" <signal_name> is forbidden in the given UCF File against the selected inputs."

For example, assume the reference UCF file has 8 bits DQ[0:7] and the selected data width in the GUI is 16 bits. While checking, the MIG tool verifies only eight bits, and reports the other bits as forbidden as follows:

```
WARNING : cntrl0_DDR2_DQ[8] is forbidden in the given UCF File against
the selected inputs.
WARNING : cntrl0_DDR2_DQ[9] is forbidden in the given UCF File against
the selected inputs.
WARNING : cntrl0_DDR2_DQ[10] is forbidden in the given UCF File against
the selected inputs.
WARNING : cntrl0_DDR2_DQ[11] is forbidden in the given UCF File against
the selected inputs.
WARNING : cntrl0_DDR2_DQ[12] is forbidden in the given UCF File against
the selected inputs.
WARNING : cntrl0_DDR2_DQ[13] is forbidden in the given UCF File against
the selected inputs.
WARNING : cntrl0_DDR2_DQ[14] is forbidden in the given UCF File against
the selected inputs.
WARNING : cntrl0_DDR2_DQ[15] is forbidden in the given UCF File against
the selected inputs.
```

6. Bank selection

   If one or more banks are not selected and one or more pins from that (those) bank(s) is (are) used for some purpose, an error message is printed.

   The error message format is "<signal_name> (<signal_group>) is not allowed to be allocated in Bank (<bank_number>) against the selected inputs."

   For example:

   ```
   NET  "cntrl0_DDR2_DQS[0]"          LOC = "D12" ;#bank 6
   ```

   Bank 6 is not selected for Data (as cntrl0_DDR2_DQS[0] from Data). Assume that cntrl0_DDR2_DQS[0], which belongs to the strobe group is allocated to a pin belonging to bank 6. The following error message is printed:

   ```
   ERROR: cntrl0_DDR2_DQS[0] (strobe) is not allowed to be allocated in
   bank 6 against the selected inputs.
   ```

Generate SL361/ML461 Board Files

The **Generate SL361/ML461 board files** button allows users to generate the hardware board files for the selected boards. Most of the designs that the MIG 1.5 tool supports are hardware tested. The board files generated out of the tool have the bit files. The bit files can be programmed onto the reference boards for demonstration purposes.

Restore

Users click on the **Restore** button to restore the default options.

Generate

After selecting all the design options, the user clicks on the **Generate** button to generate the design files. The MIG 1.5 tool generates an output directory with a name and location specified by the user. The following files and directories are created under the top-level directory:

- `usr`

  This directory contains the outputs generated by the MIG 1.5 tool.

- `datasheet`

  This is a brief data sheet of the generated interface.

- `log`

  This is a log file.

The `usr` directory contains the following subdirectories:

- `docs`

  Any relevant documents, such as application notes and timing analysis spreadsheets, are in this directory.

- `par`

  Contains the UCF file with constraints for the design. A script file called `ise_flow` is generated.

  The user can double-click on the `ise_flow` script file to run the design through synthesis, build, map, and par. This script file sets all the required options.

- `rtl`

  Contains all the RTL files (Verilog and VHDL) for the generated design.

- `synth`

  Contains the SDC file for Synplify pro with the design constraints. This folder also has the script files, which set various tool options. There is also a project file, through which the RTL files are passed for synthesis.

- `sim`

  Contains the sample files required for simulation. Users modify the test bench files according to their designs.

Dismiss

When the user clicks on the **Dismiss** button, the tool is closed.

User Guide

When the user clicks on this button, the user guide is opened in PDF format.

Version Info

This option shows the MIG release version, the date of release, the simulator version, the synthesis tool versions, and the ISE version.

Section 1: Virtex-4 FPGA to Memory Interfaces

XILINX®

Chapter 2

Implementing DDR SDRAM Controllers

This chapter describes how to implement DDR SDRAM interfaces for Virtex-4 FPGAs created with the MIG 1.5 design tool. This design is based on XAPP701 [Ref 10] and XAPP709 [Ref 13].

Feature Summary

The DDR SDRAM controller design supports:

- Burst lengths of two, four, and eight
- Sequential and interleaved burst types
- CAS latencies of 2, 2.5, and 3
- Precharge based on the row to be accessed or the user input
- Registered DIMMs
- Different memories (density/speed)
- Auto refresh
- VHDL and Verilog

The supported features are described in more detail in "Architecture."

Architecture

Interface Model

DDR SDRAM interfaces are source-synchronous and double data rate. They transfer data on both edges of the clock cycle. A memory interface can be modularly represented as shown in Figure 2-1. A modular interface has many advantages. It allows designs to be ported easily and also makes it possible to share parts of the design across different types of memory interfaces.

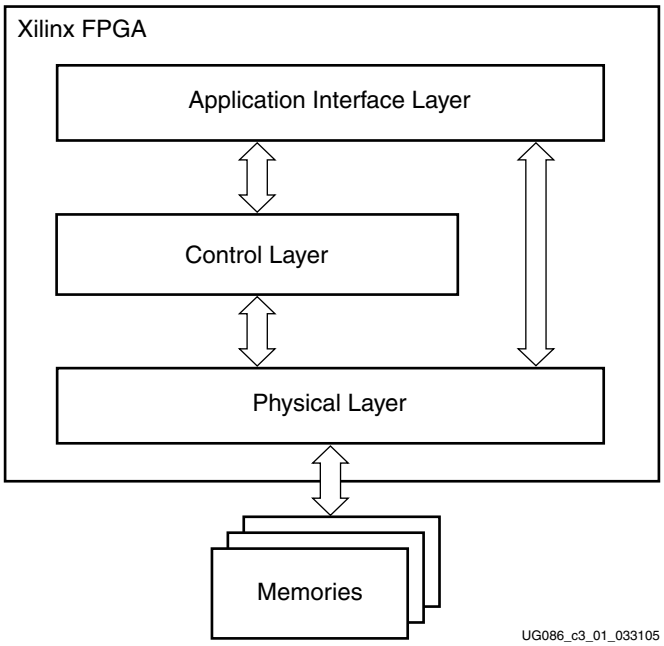


Figure 2-1: **Modular Memory Interface Representation**

Implemented Features

This section provides details on the supported features of the DDR SDRAM controller. Based on user selection, the tool generates a parameter file, which is used to program various features of the memory and to generate the control signals accordingly.

The parameter file provides the settings for burst length, CAS latency, sequential or interleaved addressing, number of row address bits, number of column address bits, bank address, and the timing parameters based on the frequency and the speed grade selected by the user in the GUI tool. The controller takes these parameters directly.

The user can also issue a command through the FIFOs (user_interface). The user address (i.e., APP_AF_ADDR that is written into the FIFO as shown in Figure 2-6) is decoded in a sequence. The total width is 32 bits consisting of column address (least-significant), row address, bank address, chip address, command to be issued, and whether the row to be accessed is same as that of the previous row. The controller takes the row and column address bits based on the selected component. Table 2-1 lists the commands that the user can issue through the User interface.

Table 2-1: **User Commands**

Command	APP_AF_ADDR[34:32]
READ	101
WRITE	100
LOAD MODE REGISTER	000
REFRESH	001
PRECHARGE	010

Burst Length

Bits M0:M3 of the Mode Register define the burst length and burst type. Read and write accesses to the DDR SDRAM are burst-oriented. The burst length is programmable to either 2, 4, or 8 through the GUI tool. It determines the maximum number of column locations accessed for a given READ or WRITE command.

The DDR SDRAM ddr_controller module implements a burst length that is programmed via the generated parameter file based on the user selection from the tool.

CAS Latency

Bits M4:M6 of the Mode Register define the CAS latency (CL). CL is the delay in clock cycles between the registration of a READ command and the availability of the first bit of output data. CL can be set to 2, 2.5, or 3 clocks through the GUI tool from the parameter file. CAS latency is implemented in the ddr_controller module.

During read data operations, the generation of the read_en signal varies according to the CL in the ddr_controller module.

Registered DIMMs

DDR SDRAM supports registered DIMMs. This feature is implemented in the ddr_controller module. For registered DIMMs, the READ and WRITE commands have one additional clock latency than unbuffered DIMMs. Also for registered DIMMs, the controller delays the data and the strobe by one clock because the command has one clock latency due to the register in the DIMM.

Precharge

The PRECHARGE command is issued before the next read or write is issued for another row or bank, but not if the read or write is in the same row and bank. The AUTO PRECHARGE command via the A10 column bit is not supported because it can cause the controller to get into an unexpected state.

Auto Refresh

The DDR SDRAM controller issues AUTO REFRESH commands at specified intervals for the memory to refresh the charge required to retain the data in the memory. The user can also issue a REFRESH command through the user interface by setting bits 34, 33, and 32 of the app_af_addr signal in the user_interface module to 3'b001. If the REFRESH command is issued within a read or a write burst, the controller issues the REFRESH command and then returns to the ongoing READ or WRITE command.

Different Memories (Density/Speed)

This feature supports different memory components and DIMMs. The component densities can vary from 128 Mb to 1 Gb, and the DIMM densities can vary from 128 MB to 1 GB. To support this feature, the design can decode write and read addresses from the user in the DDR SDRAM controller module. The user address consists of row, column, bank, and chip addresses, and the user command. Apart from the address decoding, timing parameters vary according to the density and speed grade.

Table 2-2 lists the timing parameters for components, and Table 2-3 lists the timing parameters for DIMMs.

Table 2-2: **Timing Parameters for Components**

Parameter	Description		Micron 128 Mb		Micron 256 Mb		Micron 512 Mb		Micron 1 Gb	
			-5	-75	-5	-75	-5	-75	-5	-75
T_{CK}	Clock Cycle Time	CL = 3	5 ns	NA	5 ns	NA	5 ns	NA	5 ns	NA
		CL = 2.5	6 ns	7.5 ns	6 ns	7.5 ns	6 ns	7.5 ns	6 ns	7.5 ns
		CL = 2	7.5 ns	10 ns	7.5 ns	10 ns	7.5 ns	10 ns	7.5 ns	10 ns
T_{MRD}	LOAD MODE Command Cycle Time		10 ns	15 ns	10 ns	15 ns	10 ns	15 ns	10 ns	15 ns
T_{RP}	PRECHARGE Command Period		15 ns	20 ns	15 ns	20 ns	15 ns	20 ns	15 ns	20 ns
T_{RFC}	REFRESH Time		70 ns	75 ns	70 ns	75 ns	70 ns	75 ns	120 ns	120 ns
T_{RCD}	ACTIVE to READ or WRITE Delay		15 ns	20 ns	15 ns	20 ns	15 ns	20 ns	15 ns	20 ns
T_{RAS}	ACTIVE to PRECHARGE Command		40 ns	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns
T_{RC}	ACTIVE to ACTIVE (Same Bank) Command		55 ns	65 ns	55 ns	65 ns	55 ns	65 ns	55 ns	65 ns
T_{WTR}	WRITE to READ Command Delay		$2 * T_{CK}$	$1 * T_{CK}$	$2 * T_{CK}$	$1 * T_{CK}$	$2 * T_{CK}$	$1 * T_{CK}$	$2 * T_{CK}$	$1 * T_{CK}$
T_{WR}	WRITE Recovery Time		15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns

Table 2-3: **Timing Parameters for DIMMs (Unbuffered and Registered)**

Parameter	Description		Micron 128 MB	Micron 256 MB	Micron 512 MB	Micron 1 GB
			-40	-40	-40	-40
T_{CK}	Clock Cycle Time	CL = 3	5 ns	5 ns	5 ns	5 ns
		CL = 2.5	6 ns	6 ns	6 ns	6 ns
		CL = 2	7.5 ns	7.5 ns	7.5 ns	7.5 ns
T_{MRD}	LOAD MODE Command Cycle Time		10 ns	10 ns	10 ns	10 ns
T_{RP}	PRECHARGE Command Period		15 ns	15 ns	15 ns	15 ns
T_{RFC}	REFRESH Time		70 ns	70 ns	70 ns	70 ns
T_{RCD}	ACTIVE to READ or WRITE Delay		15 ns	15 ns	15 ns	15 ns
T_{RAS}	ACTIVE to PRECHARGE Command		40 ns	40 ns	40 ns	40 ns
T_{RC}	ACTIVE to ACTIVE (Same Bank) Command		55 ns	55 ns	55 ns	55 ns

Table 2-3: **Timing Parameters for DIMMs (Unbuffered and Registered) *(Continued)***

Parameter	Description	Micron 128 MB	Micron 256 MB	Micron 512 MB	Micron 1 GB
		-40	-40	-40	-40
T_{WTR}	WRITE to READ Command Delay	2 * T_{CK}	2 * T_{CK}	2 * T_{CK}	2 * T_{CK}
T_{WR}	WRITE Recovery Time	15 ns	15 ns	15 ns	15 ns

Hierarchy

Figure 2-2 shows the hierarchical structure of the modular design. The physical and control layers are clearly separated in this figure. The MIG 1.5 tool generates the entire DDR SDRAM controller as shown in this hierarchy, including the test bench.

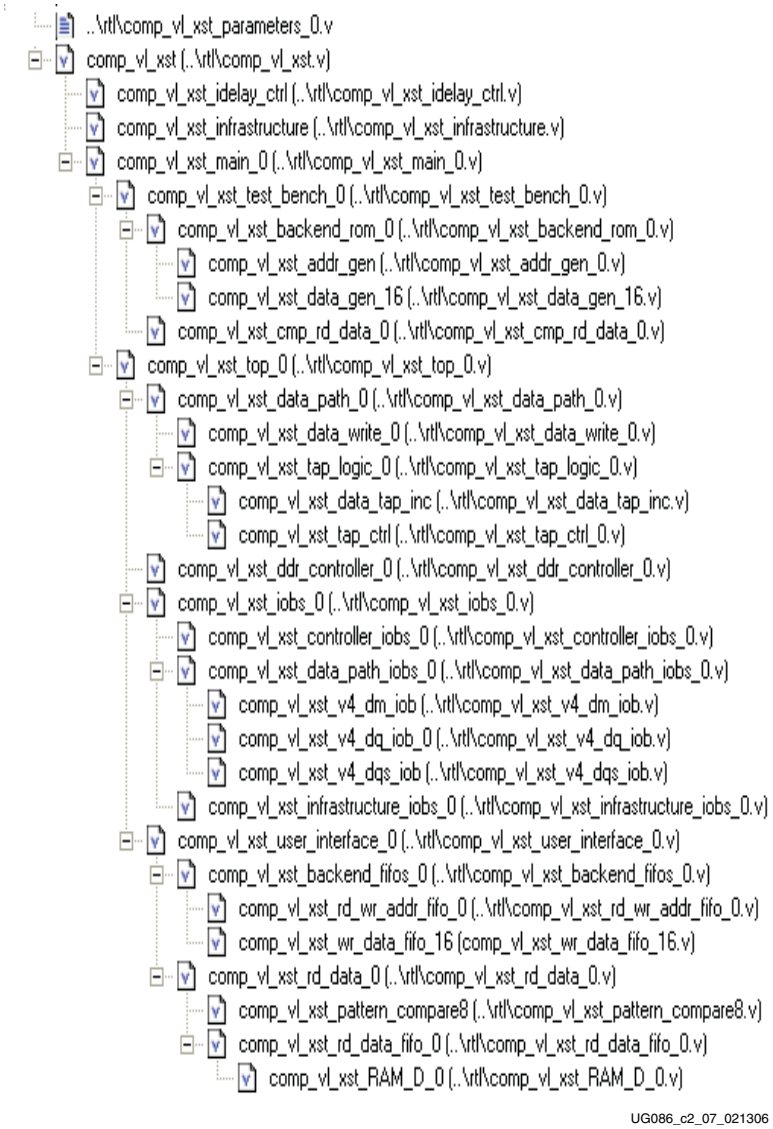


Figure 2-2: **Hierarchical Structure of the DDR SDRAM Virtex-4 Design**

DDR Controller Submodules

Figure 2-3 is a detailed block diagram of the DDR SDRAM controller. The six blocks shown in this figure are the sub-blocks of the top module. The functions of these blocks are explained in the subsections following the figure.

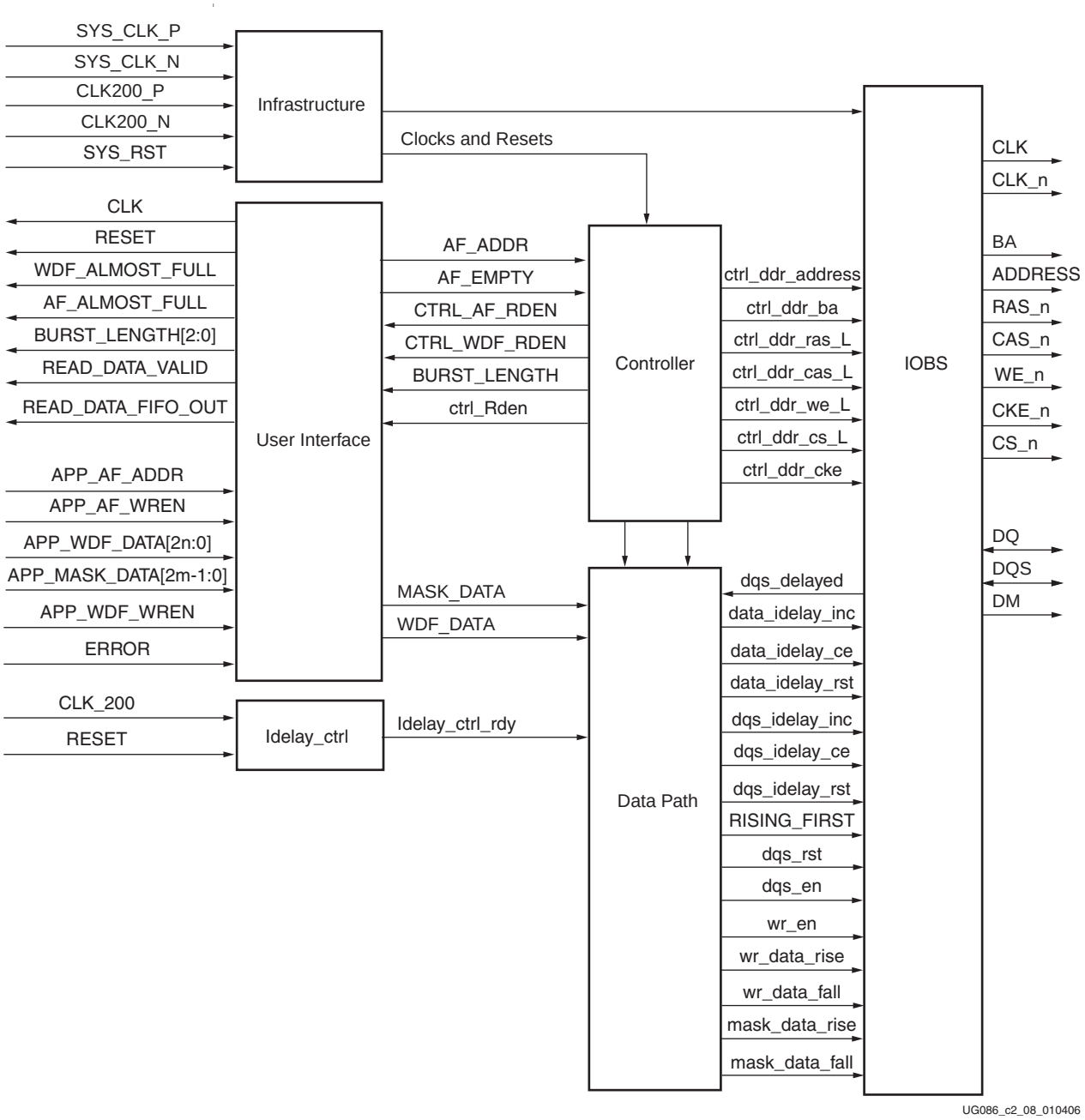


Figure 2-3: **DDR SDRAM Controller Block Diagram**

Controller

The DDR SDRAM controller initializes the memory, accepts and decodes user commands, and generates READ, WRITE, and REFRESH commands. The DDR SDRAM controller also generates signals for other modules. The memory is initialized and powered-up using a defined process. The controller state machine handles the initialization process upon

power-up. If the AUTO REFRESH command is to be issued between any user read or write commands, then the read or write command will be suspended until the ar_done flag is deasserted. The user must not issue any other command until the ar_done flag is deasserted.

Datapath

This module transmits data to the memories. Its major functions include storing the write data and calculating the tap value for the read datapath. The data_write and data_path_IOBs modules do the actual write functions. The Idelay_ctrl, tap_ctrl and data_tap_inc modules do the calibration.

User Interface

This module stores write data in its Write Data FIFO (wr_data_fifo), stores write and read addresses in its Read/Write Address FIFO (rd_wr_addr_fifo), and stores received read data from memory in its Read Data FIFO (rd_data_fifo). The width of the Write Data FIFO is twice the data width and mask width of the memory. For example, for a 16-bit width, the width of the FIFO is 36 because the data width is 32 and the mask width is 4. The rd_wr_addr_fifo and wr_data_fifo modules store the data and address in block RAMs. The rd_data_fifo module captures the data in the LUT-based RAMs.

The controller also generates user commands, such as READ and WRITE. To execute a WRITE command, the user loads both READ and WRITE commands. The controller starts a WRITE operation only when both READ and WRITE commands are loaded into their corresponding FIFOs. The controller processes all subsequent commands alternately (READ, WRITE, READ, WRITE, and so forth).

The pattern_compare module registers the delay between the command and the data received from the IOBs. This delay is then applied to the Rden signal generated from the ddr_controller module during the actual read to register the valid data in the internal FIFOs.

Infrastructure

The infrastructure module generates the FPGA clocks and reset signals. A DCM generates the phase-shifted clocks (clk0, clk90), refresh clock, and calibration clock. All the reset signals required for the design are also generated.

IOBS Module

All DDR SDRAM address, control, and data signals are transmitted and received in the through the input and output buffers.

DDR SDRAM Memory Initialization and Calibration

DDR memory is initialized through a specified sequence as shown in Figure 2-4. Following the initialization, the relationship between the data and the FPGA clock is calculated using the TAP logic. The controller issues a dummy read command to the memory. As the data and the memory strobe are edge-aligned, the strobe is passed through the IDELAY elements of the Virtex-4 device and the taps are adjusted to find the center of the strobe pulse. The number of taps is then used to delay the data during normal reads to register the valid data in the FPGA. XAPP701 [Ref 10] provides more information.

Following the strobe detection, the controller does a pattern write and then a pattern read to find the delay of the data it takes from the IOBs. As soon as the read command is issued, the data from the IOB is registered at every clock cycle and is checked for the fixed pattern written. The number of clocks required for the valid data is registered and used during normal reads to capture valid data in the internal block RAM.

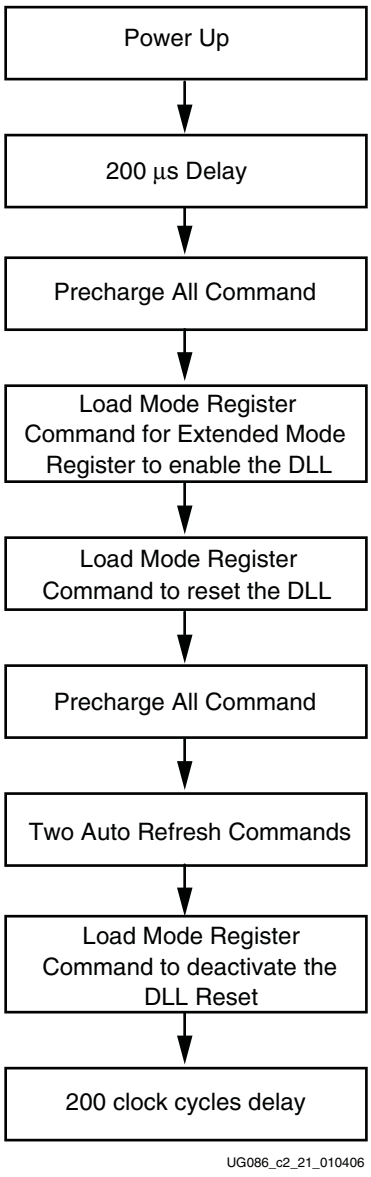


Figure 2-4: **DDR Memory Initialization Sequence**

DDR SDRAM System and User Interface Signals

Table 2-4 describes the DDR SDRAM system interface signals. The system interface signals are the clocks and the reset signals given by the user to the FPGA. SYS_CLK_P and SYS_CLK_N are the two clocks to be provided to the design. These two clocks must have a phase difference of 180 degrees with respect to each other. SYS_RST resets all the logic.

Table 2-4: **DDR SDRAM System Interface Signals**

Signal Name	Direction	Description
SYS_CLK_P, SYS_CLK_N	Input	Differential input clock to the DCM. The DDR SDRAM controller and memory operate on this frequency.
SYS_RST	Input	Active-Low reset to the DDR SDRAM controller.
CLK200_P, CLK200_N	Input	Differential clock used in the idelay_ctrl logic.

Table 2-5 describes the DDR SDRAM user interface signals.

Table 2-5: **DDR SDRAM User Interface Signals**

Signal Name	Direction	Description
CLK, CLK90, CLK50	Output	Output clocks from the DCM to the user interface and the controller. These clocks are input to the user. CLK is the memory clock.
RESET	Output	Active-High system reset for the user interface.
BURST_LENGTH	Output	Indicates the number of bursts that can be written to or read from the memory.
READ_DATA_VALID	Output	Status of the Read Data FIFO. This signal is asserted when read data is available in the Read Data FIFO.
READ_DATA_FIFO_OUT	Output	Read data from memory, where *n* is the data width of the memory. The read data is stored into the Read Data FIFO. This data can be read from the FIFO depending upon the status of the FIFO.
WDF_ALMOST_FULL	Output	ALMOST FULL status of the Write Data FIFO. When this signal is asserted, the user can write five more data words into the FIFO.
AF_ALMOST_FULL	Output	FULL status of the Read Address FIFO.
APP_AF_ADDR	Input	User write address for the write data. This address consists of the row, column, bank, and chip addresses. The write address is decoded in the DDR SDRAM controller module and is synchronized with the WRITE command.
APP_AF_WREN	Input	Write-enable signal to the Write Address FIFO. This signal is synchronized with the write address. The write address is written to the Write Address FIFO only when this signal is asserted High.

Table 2-5: **DDR SDRAM User Interface Signals *(Continued)***

Signal Name	Direction	Description
APP_WDF_DATA	Input	User write data to the memory, where *n* indicates the data width of the memory. The user write data is twice the data width of the memory. The most-significant bits contain the rising-edge data, and the least-significant bits contain the falling-edge data. Memory write data is written into the Write Data FIFO, and the write address is written into the Write Address FIFO from the user interface. The DDR SDRAM controller reads the Write Address FIFO and Write Data FIFO.
APP_MASK_DATA	Input	User mask data, where *m* indicates the data mask width of the memory. The mask data is written into the Write Data FIFO along with the write data.
APP_WDF_WREN	Input	Write-enable signal to the Write Data FIFO. This signal is synchronized with the write data. The write data is written to the Write Data FIFO only when this signal is asserted High.
ERROR	Input	Signal that indicates whether the read data and the write data are the same or not.

DDR SDRAM Writes (User Interface)

Figure 2-5 shows the timing diagram for a user write burst to the DDR SDRAM with burst lengths set to four. The user interface signals consist of WDF_ALMOST_FULL, APP_AF_WREN, APP_AF_ADDR, APP_WDF_WREN, and APP_WDF_DATA. The user interface consists of a Write Address FIFO and a Write Data FIFO for memory write operations.

The user initiates a write to memory by writing to the User Write Data FIFO and User Write Address FIFO when WDF_ALMOST_FULL is deasserted. The WDF_ALMOST_FULL signal is asserted when the Write Data FIFO is full. The APP_AF_WREN signal is the write-enable signal for the Address FIFO, and APP_AF_ADDR is the write address to the Write FIFO. The APP_WDF_WREN signal is the write-enable signal for the Write Data FIFO, and APP_WDF_DATA is the user write data to the memory. APP_WDF_DATA is twice the width of the memory width, consisting of rising-edge and falling-edge data.

For burst lengths of four, each write address consists of two write data cycles. Figure 2-5 shows a write burst with four consecutive bursts.

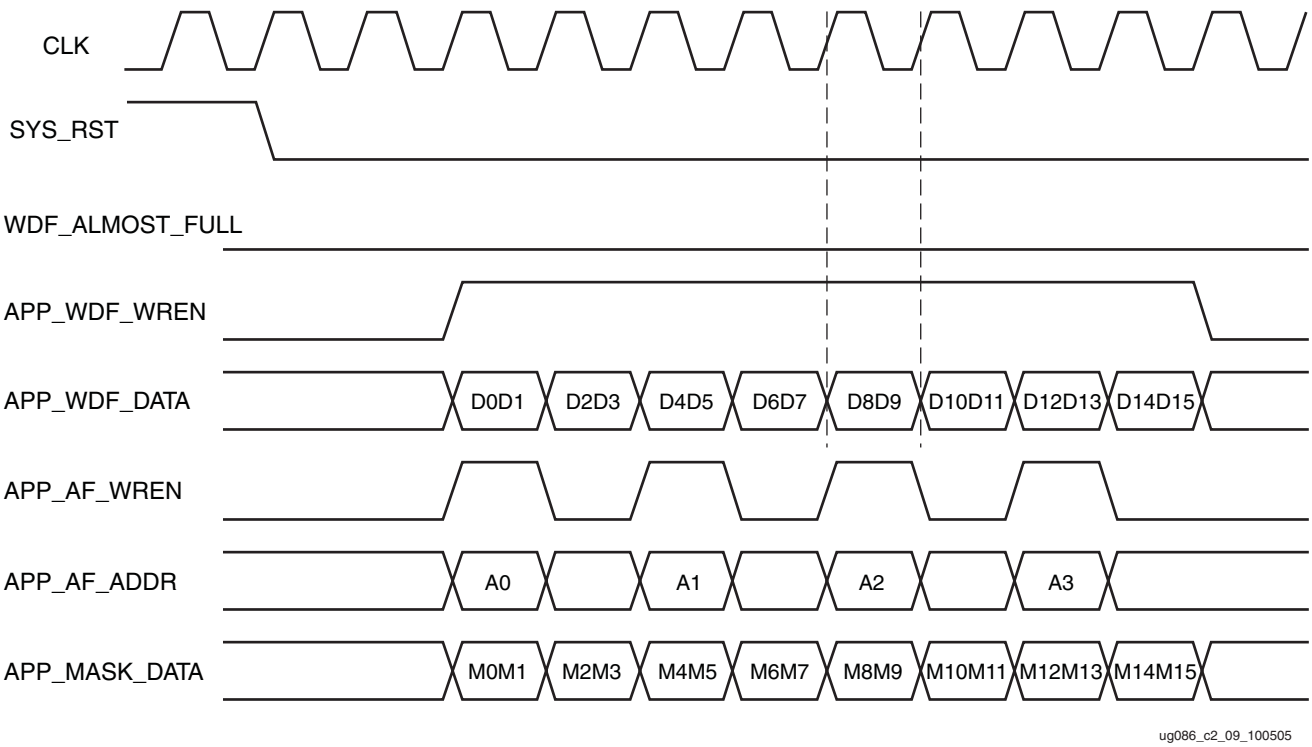


Figure 2-5: **DDR SDRAM Write Burst for Four Bursts (BL = 4)**

DDR SDRAM Reads (User Interface)

Figure 2-6 shows the timing diagram for a user read burst with burst length set to four. The user interface signals consist of AF_ALMOST_FULL, APP_AF_WREN, APP_AF_ADDR, READ_DATA_VALID, and READ_DATA_FIFO_OUT.

To initiate a memory read, the user writes the read address to the Read Address FIFO when the AF_ALMOST_FULL signal is deasserted (not full). The APP_AF_WREN signal is the write-enable signal for the Read Address FIFO, and APP_AF_ADDR is the memory read address. The read_data_valid signal is asserted when the read data is available to the user, and READ_DATA_FIFO_OUT is the read data from the memory to the user.

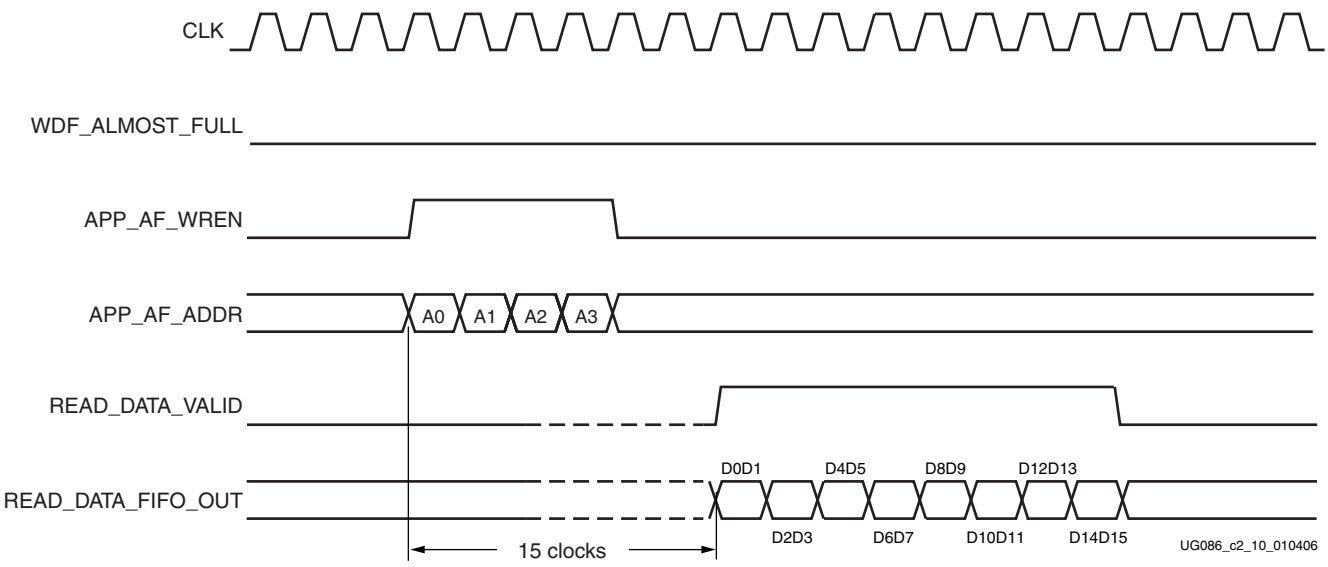


Figure 2-6: **DDR SDRAM Read Burst for Four Bursts (BL = 4)**

The 15 clocks from the Read command (indicated by assertion of APP_AF_WREN) to the read data (data is valid on READ_DATA_FIFO_OUT when READ_DATA_VALID is asserted) are broken down as indicated in Table 2-6.

Table 2-6: **Read Command to Read Data Clock Cycles**

Parameter	Number of Clock Cycles
Read Address to Read Command	4 clocks
Read Command to Read Data (the CAS latency is 3)	3 clocks
Read Data to IOB Output	2 clocks
IOB Output to Data Valid	6 clocks
Total:	**15 clocks**

Chapter 3

Implementing DDR2 SDRAM Controllers

This chapter describes how to implement DDR2 SDRAM interfaces for Virtex-4 FPGAs created with the MIG 1.5 design tool. The MIG 1.5 tool supports two implementations of DDR2 SDRAM interfaces: direct clocking and SerDes clocking. The direct clocking interface supports frequencies up to 267 MHz. This design is based on XAPP702 [Ref 11]. The SerDes clocking design supports frequencies up to 330 MHz and is based on XAPP721 [Ref 15].

Interface Model

DDR2 SDRAM interfaces are source-synchronous and double data rate. They transfer data on both edges of the clock cycle. A memory interface can be modularly represented as shown in Figure 3-1. A modular interface has many advantages. It allows designs to be ported easily and also makes it possible to share parts of the design across different types of memory interfaces.

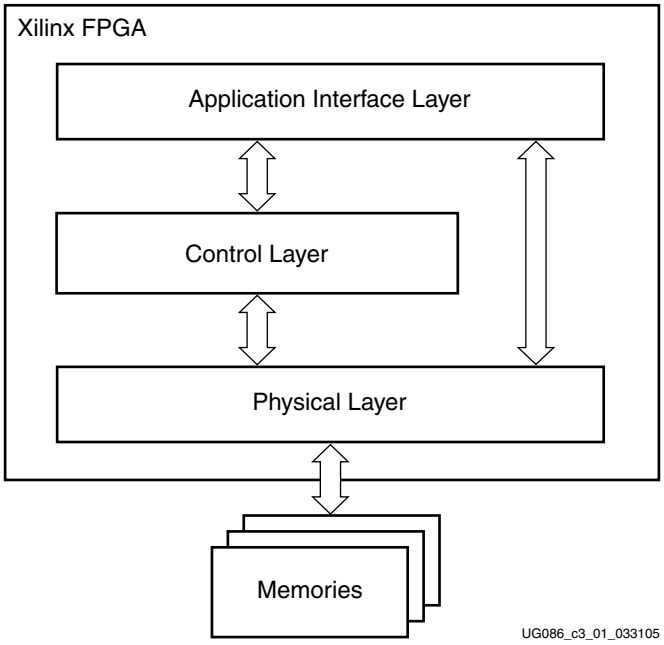


Figure 3-1: **Modular Memory Interface Representation**

Direct Clocking Interface

Feature Summary

This section summarizes the supported and unsupported features of the direct clocking DDR2 SDRAM controller design.

Supported Features

The DDR2 SDRAM controller design supports:

- Burst lengths of four and eight
- Sequential burst types
- CAS latencies of 3 and 4
- Additive latencies of 0, 1, and 2
- Differential DQS
- On-Die Termination (ODT)
- Up to four deep memories
- Registered DIMMs (up to 267 MHz)
- Unbuffered DIMMs (up to 200 MHz)
- Unbuffered SODIMMs (up to 200 MHz)
- Different memories (density/speed)
- ECC support
- Verilog and VHDL
- With or without a test bench
- With or without a DCM
- Multicontrollers (up to eight)

The supported features are described in more detail in "Architecture."

Unsupported Features

The DDR2 SDRAM controller design does not support:

- Interleaved burst types
- Additive latencies of 3 and 4
- Single-ended DQS
- Redundant DQS (RDQS)
- Unbuffered DIMMs (greater than 200 MHz)
- Unbuffered SODIMMs (greater than 200 MHz)

Architecture

Implemented Features

This section provides details on the supported features of the DDR2 SDRAM controller.

Burst Length

Bits M0:M3 of the Mode Register define the burst length. Read and write accesses to the DDR2 SDRAM are burst-oriented. The burst length is programmable to either four or eight. It determines the maximum number of column locations accessed for a given READ or WRITE command.

The DDR2 SDRAM ddr2_controller module implements burst lengths of four and eight.

CAS Latency

Bits M4:M6 of the Mode Register define the CAS latency (CL). CL is the delay in clock cycles between the registration of a READ command and the availability of the first bit of output data. CL can be set to three or four clocks.

CAS latency is implemented in the ddr2_controller module. During data write operations, the generation of the dqs_ena and dqs_rst signals varies according to the CL in the ddr2_controller module. During read data operations, the generation of the read_en signal varies according to the CL in the ddr2_controller module.

Additive Latency

DDR2 SDRAM devices support a feature called posted CAS additive latency (AL). This feature allows the READ command to be issued prior to t_{RCD} (minimum) by delaying the internal command to the DDR2 SDRAM by AL clocks. Posted CAS AL makes the command and data bus efficient for sustainable bandwidths in DDR2 SDRAM. Bits E3:E5 of the Extended Mode Register define the value of AL. Bits E3:E5 allow the user to program the DDR2 SDRAM with a CAS additive latency of 0, 1, 2, 3, or 4 clocks.

Additive latency is implemented in the DDR2 SDRAM ddr2_controller module. The ddr2_controller module issues READ/WRITE commands prior to t_{RCD} (minimum) depending on the user-selected AL value in the Extended Mode Register. Currently 0, 1, and 2 clocks are supported.

Registered DIMMs

DDR2 SDRAM supports registered DIMMs. This feature is implemented in the ddr2_controller module. For registered DIMMs, the read and write commands have one additional clock latency than unbuffered DIMMs.

Different Memories (Density/Speed)

This feature supports different memory components and DIMMs. The component densities can vary from 256 Mb to 1 Gb, and the DIMM densities can vary from 128 MB to 4 GB. To support this feature, the design can decode write and read addresses from the user in the DDR2 SDRAM controller module. The user address consists of row, column, bank, chip addresses, and user command.

Apart from the address decoding, timing parameters vary according to the density and speed grade. Table 3-1 lists the timing parameters for components, and Table 3-2 lists the timing parameters for DIMMs.

Table 3-1: **Timing Parameters for Components**

Parameter	Description	Micron 256 Mb		Micron 512 Mb		Micron 1 Gb	
		-37E	-5E	-37E	-5E	-37E	-5E
T_{MRD}	LOAD MODE command cycle time	2	2	2	2	2	2
T_{RP}	PRECHARGE command period	15	15	15	15	15	15
T_{RFC}	REFRESH to ACTIVE or REFRESH to REFRESH command interval	75	75	105	105	127.5	127.5
T_{RCD}	ACTIVE to READ or WRITE delay	15	15	15	15	15	15
T_{RAS}	ACTIVE to PRECHARGE command	40	40	40	40	40	40
T_{RC}	ACTIVE to ACTIVE (same bank) command	55	55	55	55	55	55
T_{RTP}	READ to PRECHARGE command delay	7.5	7.5	7.5	7.5	7.5	7.5
T_{WTR}	WRITE to READ command delay	7.5	10	7.5	10	7.5	10
T_{WR}	WRITE Recovery time	15	15	15	15	15	15

Table 3-2: **Timing Parameters for DIMMs**

Parameter	Description	MT4HTF		MT8HTF		MT16HTF		MT9HTF		MT18HTF	
		-53E	-40E	-53E	-40E	-53E	-40E	-53E	-40E	-53E	-40E
T_{MRD}	LOAD MODE command cycle time	2	2	2	2	2	2	2	2	2	2
T_{RP}	PRECHARGE command period	15	15	15	15	15	15	15	15	15	15
T_{RFC}	REFRESH to ACTIVE or REFRESH to REFRESH command interval	128 MB 75	75	256 MB 75	75	512 MB 75	75	256 MB 75	75	512 MB 75	75
		256 MB 105	105	512 MB 105	105	1 GB 105	105	512 MB 105	105	1 GB 105	105
		512 MB 127.5	127.5	1 GB 127.5	127.5	2 GB 127.5	127.5	1 GB 127.5	127.5	2 GB 127.5	127.5
T_{RCD}	ACTIVE to READ or WRITE delay	15	15	15	15	15	15	15	15	15	15
T_{RAS}	ACTIVE to PRECHARGE command	40	40	40	40	40	40	40	40	40	40
T_{RC}	ACTIVE to ACTIVE (same bank) command	55	55	55	55	55	55	55	55	55	55
T_{RTP}	READ to PRECHARGE command delay	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5	7.5
T_{WTR}	WRITE to READ command delay	7.5	10	7.5	10	7.5	10	7.5	10	7.5	10
T_{WR}	WRITE Recovery time	15	15	15	15	15	15	15	15	15	15

On-Die Termination

On-die termination (ODT) improves signal integrity in the system. An ODT pin is added to the SDRAM so the system can turn the termination on and off as needed. In a simple

system with one DRAM load per DQ signal, the termination is turned on for WRITE commands and disabled for READ commands.

In the DDR2 design, a maximum of four ODTs is supported (deep memory = 4). Four examples are given below:

1. If the user selects deep memory = 4, the memory component sequence is 0, 1, 2, and 3. During write operations, the ODT is enabled for component 3 when writing into 0, 1, or 2, otherwise it is enabled for component 2 when writing into component 3. During read operations, the ODT is enabled for component 3 when reading from 0, 1, or 2, otherwise it is enabled for component 2 for reading from component 3.
2. If the user selects deep memory = 3, the memory component sequence is 0, 1, and 2. During write operations, the ODT is enabled for component 2 when writing into 0 or 1, otherwise it is enabled for component 1 when writing into component 2. During read operations, the ODT is enabled for component 2 when reading from 0 or 1, otherwise it is enabled for component 1 for reading from component 2.
3. If the user selects deep memory = 2, the memory component sequence is 0 and 1. During write operations, the ODT is enabled for component 1 when writing into 0, otherwise it is enabled for component 0 when writing into component 1. During read operations, the ODT is enabled for component 1 when reading from 0, otherwise it is enabled for component 0 for reading from component 1.
4. If the user selects deep memory = 1, the memory component sequence is 0. During write operations, the ODT is enabled for component 0 when writing into 0. During read operations, the ODT is disabled.

Deep Memories

This feature increases the depth of the memory. For example, if the user selects from the GUI a 256 Mb component and deep memory = 4, then the tool generates a memory interface for a 1 Gb design. Deep memory logic is implemented in the DDR2 SDRAM ddr2_ controller module.

With deep memories, the DDR2 SDRAMs are initialized one after the other to avoid loading the address and control buses. Apart from initialization, the DDR2 SDRAM controller module also demultiplexes the row, column, and bank addresses from the user address. It also decodes the chip selects and rank addresses for components and DIMMs.

For deep memory implementations, the MIG 1.5 tool generates chip selects for each memory. During writes, the user sends write data and a write address. During reads, the user sends a read address. Both the write and read addresses consist of a row address, a column address, a bank address, a rank address, chip selects, and user command. The formats of user read/write addresses for a 256 Mb component and 2 GB and 4 GB DIMMs are given in "Deep Memory Configurations."

ECC Support

Error correction code (ECC) is based on XAPP645 [Ref 9]. The design can detect and correct all single bit errors, and it can detect double bit errors in the data. This design utilizes hamming code for the ECC operations.

Supported data widths are 40-bit (32-bit data and 7-bit parity), 72-bit (64-bit data and 8-bit parity), and 144-bit (128-bit data and 16-bit parity). The user can enable and disable the ECC from the GUI.

Hierarchy

Figure 3-2 shows the hierarchical structure of the modular design. The physical and control layers are clearly separated in this figure. The MIG 1.5 tool generates the entire DDR2 SDRAM controller as shown in this hierarchy, including the test bench.

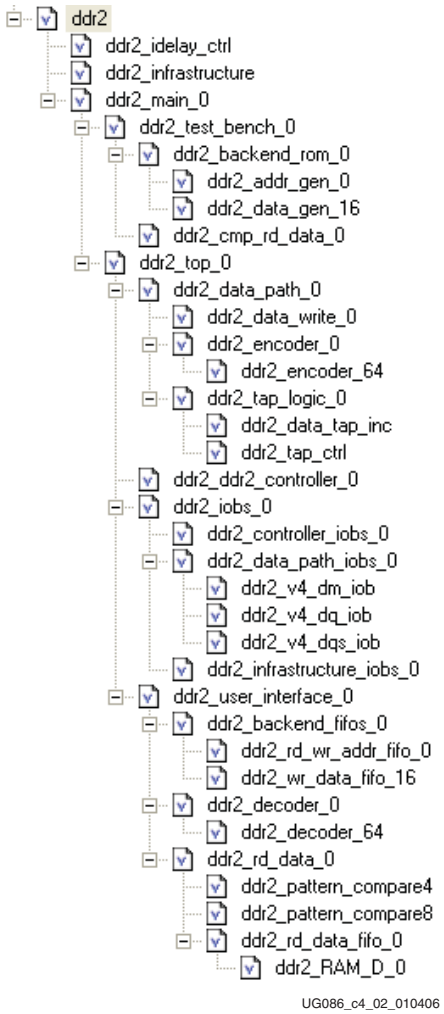


Figure 3-2: **Hierarchical Structure of the DDR2 Design (Direct Clocking)**

DDR2 Controller Submodules

Figure 3-3 is a detailed block diagram of the DDR2 SDRAM controller. The five blocks shown are the sub-blocks of the top module. The functions of these blocks are explained in the subsections following the figure.

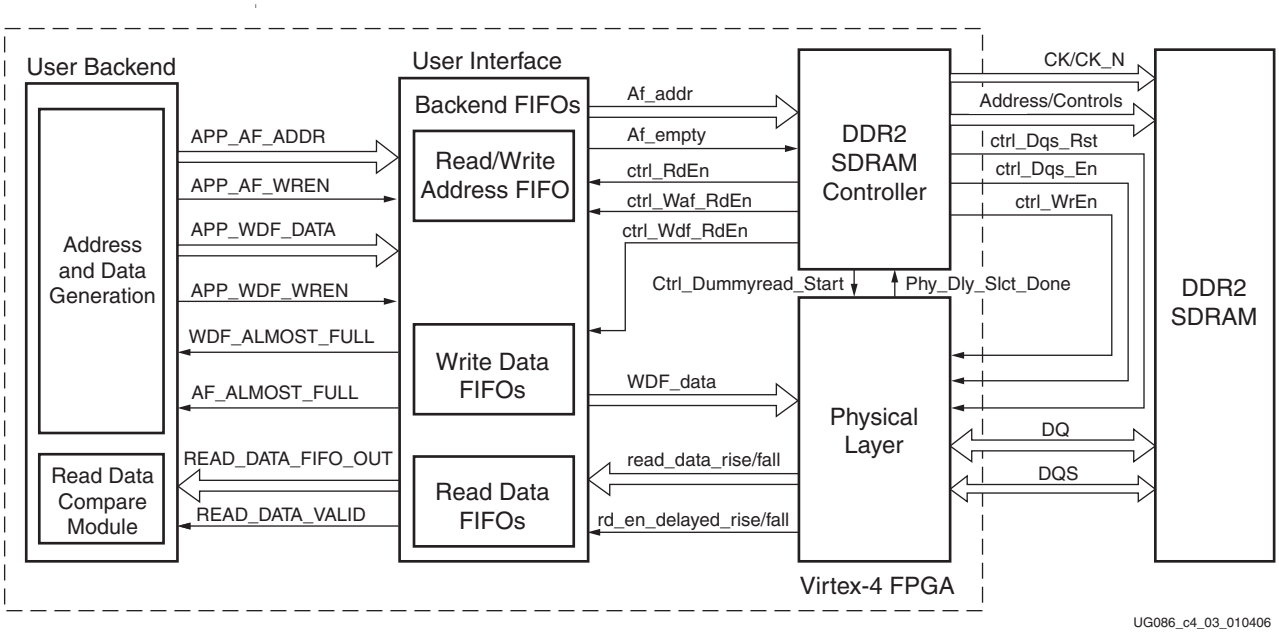


Figure 3-3: **DDR2 Memory Controller Block Diagram**

Controller

The DDR2 SDRAM ddr2_controller accepts and decodes user commands and generates read, write, and refresh commands. The DDR2 SDRAM controller also generates signals for other modules. The memory is initialized and powered-up using a defined process. The controller state machine handles the initialization process upon power-up. After memory initialization, the controller issues dummy read commands. During dummy reads, the tap_logic module calibrates the data and strobe, and delays the data to center-align with the FPGA clock.

Datapath

This module transmits data to the memories. Its major functions include storing the write data and calculating the tap value for the read datapath. The data_write and data_path_IOBs modules do the actual write functions. The Idelay_ctrl, tap_ctrl, and data_tap_inc modules do the calibration.

User Interface

This module stores write data, write addresses, and read addresses in FIFOs and receives read data from the memory. The rd_data and rd_data_fifos modules capture the data in the LUT-based RAMs. The rd_wr_addr_fifo and wr_data_fifo modules store the data and address in block RAMs.

Once the calibration is done, the controller issues a pattern_write command with a known pattern (0xAA559966) to the memory. Then the controller issues a pattern_read command from the same location and compares the read data with the known pattern in the pattern_compare8 module. During the pattern_read command, the controller generates the ctrl_rden signal, which is delayed in the pattern_compare8 module to synchronize with the read data. This delay is applied to the ctrl_rden signal generated from the ddr2_controller module during a normal read to register the valid data in the internal FIFOs.

The FIRST_RISING logic is implemented in the pattern_compare8 module. FIRST_RISING is asserted when the first data is captured with respect to the falling edge of FPGA clock. This signal is used in rd_data_fifo to swap rise and fall data.

Infrastructure Module

The infrastructure module generates the FPGA clocks and reset signals. A DCM generates the phase-shifted clocks (clk0, clk90), refresh clock, and calibration clock.

IOBS Module

All FPGA input and output signals are implemented in the IOBS module.

DDR2 SDRAM User Interface Signals

Table 3-3 describes the DDR2 SDRAM user interface signals.

Table 3-3: **DDR2 SDRAM Controller User Interface Signals**

Signal Name	Direction	Description
SYS_CLK_P, SYS_CLK_N	Input	Differential input clock to the DCM. The DDR2 controller and memory operate at this frequency.
CLK200_P, CLK200_N	Input	Differential clock used in the idelay_ctrl logic.
SYS_RESET	Input	Active-Low reset to the DDR2 controller.
CLK_TB	Output	All user interface signals must be synchronized with respect to CLK_TB.
RESET_TB	Output	Reset signal for the User Interface.
BURST_LENGTH[2:0]	Output	This signal determines the data burst length for each write address. 0b010: burst length = 4 0b011: burst length = 8
WDF_ALMOST_FULL	Input	This signal indicates the ALMOST_FULL status of the Write Data FIFO. When this signal is asserted, the user stops writing data into the Write Data FIFO and addresses into the Write Address FIFO.
APP_WDF_DATA[2n-1:0]	Input	User write data to the memory, where *n* indicates the data width of the memory. The user data is twice the data width of the memory. The most-significant bits contain the rising-edge data and the least-significant bits contain the falling-edge data.
APP_MASK_DATA[2m-1:0]	Input	User mask data to the memory, where *m* indicates the data mask width of the memory. The mask data is twice the mask width of the memory. The most-significant bits contain the rising-edge mask data and the least-significant bits contain the falling-edge mask data.
APP_WDF_WREN	Input	Write Enable signal to the Write Data FIFO.
AF_ALMOST_FULL	Output	This signal indicate the FULL status of the Address FIFO. When this signal is asserted, the user stops writing addresses into the Address FIFO.

Table 3-3: **DDR2 SDRAM Controller User Interface Signals *(Continued)***

Signal Name	Direction	Description
APP_AF_ADDR[35:0]	Input	User address consists of a memory address and dynamic commands. The address width [31:0] is the memory read/write address including the column, row, bank, and chip address. APP_AF_ADDR35 is fixed at zero. The address width [34:32] represents the following dynamic commands: 000 – Load Mode Register 001 – Auto Refresh 010 – Precharge All 011 – Active 100 – Write 101 – Read
APP_AF_WREN	Input	Write Enable signal to the Address FIFO.
READ_DATA_FIFO_OUT[2n-1:0]	Output	Read data from the memory, where *n* indicates the data width of the memory. The most-significant bits of the read data consist of the rising-edge data and the least-significant bits consist of the falling-edge data.
READ_DATA_VALID	Output	This signal is asserted to indicate the read data is available to the user.

Notes:

1. All user interface signal names are prepended with a controller number. DDR2 SDRAM devices support multicontroller operation, where a maximum of eight controllers can be selected by the user from the MIG 1.5 tool. For example, when the user selects eight controllers, the signal names have the following format: cntrl0_user_signal, cntrl1_user_signal, cntrl2_user_signal, cntrl3_user_signal, cntrl4_user_signal, cntrl5_user_signal, cntrl6_user_signal, and cntrl7_user_signal.

DDR2 SDRAM Writes

Figure 3-4 shows the timing diagram for a user write burst to the DDR2 SDRAM with burst lengths set to four. The user interface signals consist of WDF_ALMOST_FULL, AF_ALMOST_FULL, APP_AF_WREN, APP_AF_ADDR, APP_WDF_WREN, APP_MASK_DATA, and APP_WDF_DATA. The user interface consists of an Address FIFO and a Write Data FIFO.

The user initiates a write to memory by writing to the User Write Data FIFO and User Write Address FIFO when WDF_ALMOST_FULL and AF_ALMOST_FULL are deasserted. The Write Data FIFO is full when WDF_ALMOST_FULL is asserted. The APP_AF_WREN signal is the write-enable signal for the Address FIFO, and APP_AF_ADDR is the write address to the Address FIFO. The APP_WDF_WREN signal is the write-enable signal for the Write Data FIFO, and APP_WDF_DATA is the user write data to the memory. APP_WDF_DATA is twice the width of the memory width, consisting of rise and fall data.

For burst lengths of four, each write address consists of two write data cycles. Figure 3-4 shows a write burst with four consecutive bursts.

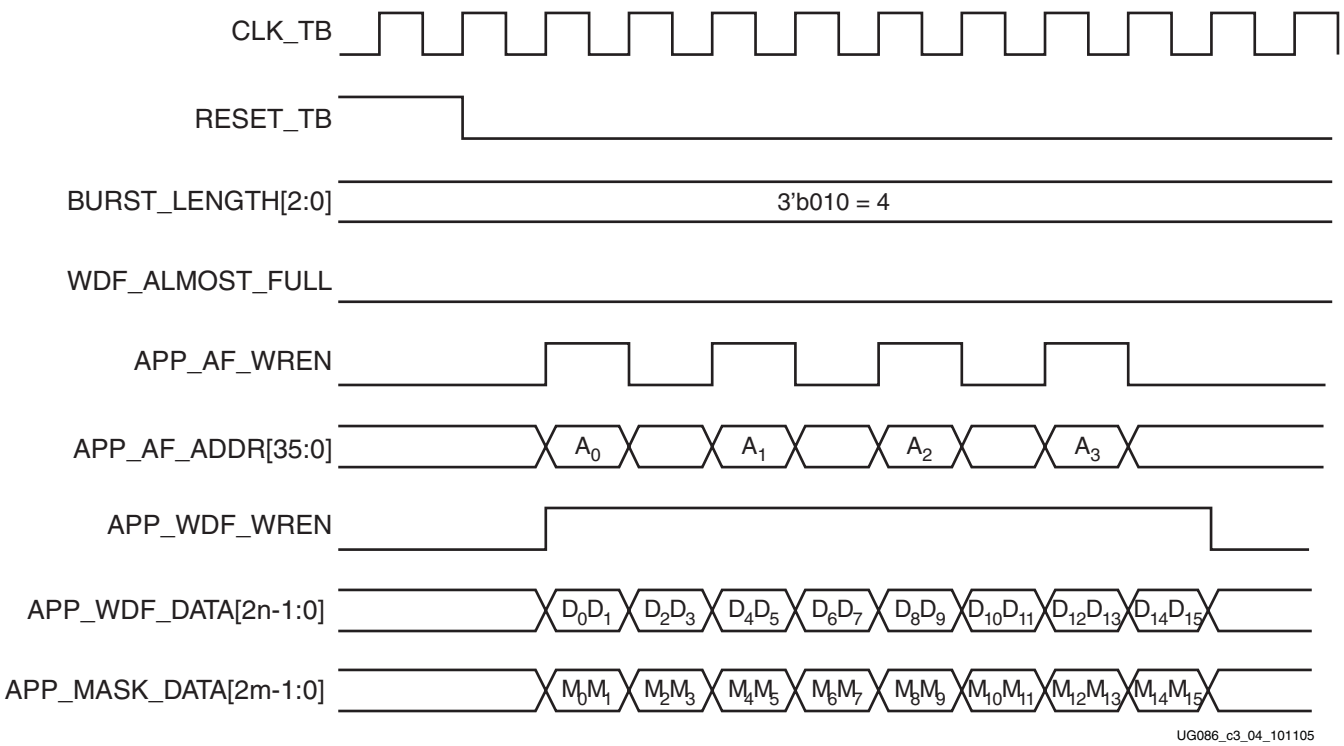


Figure 3-4: **DDR2 SDRAM Write Burst (BL = 4) for Four Bursts**

Figure 3-5 shows the timing diagram for a write burst with burst length set to eight. The user interface signals are the same for burst lengths of four and eight. When the burst length is eight, the user writes four write data cycles for each address.

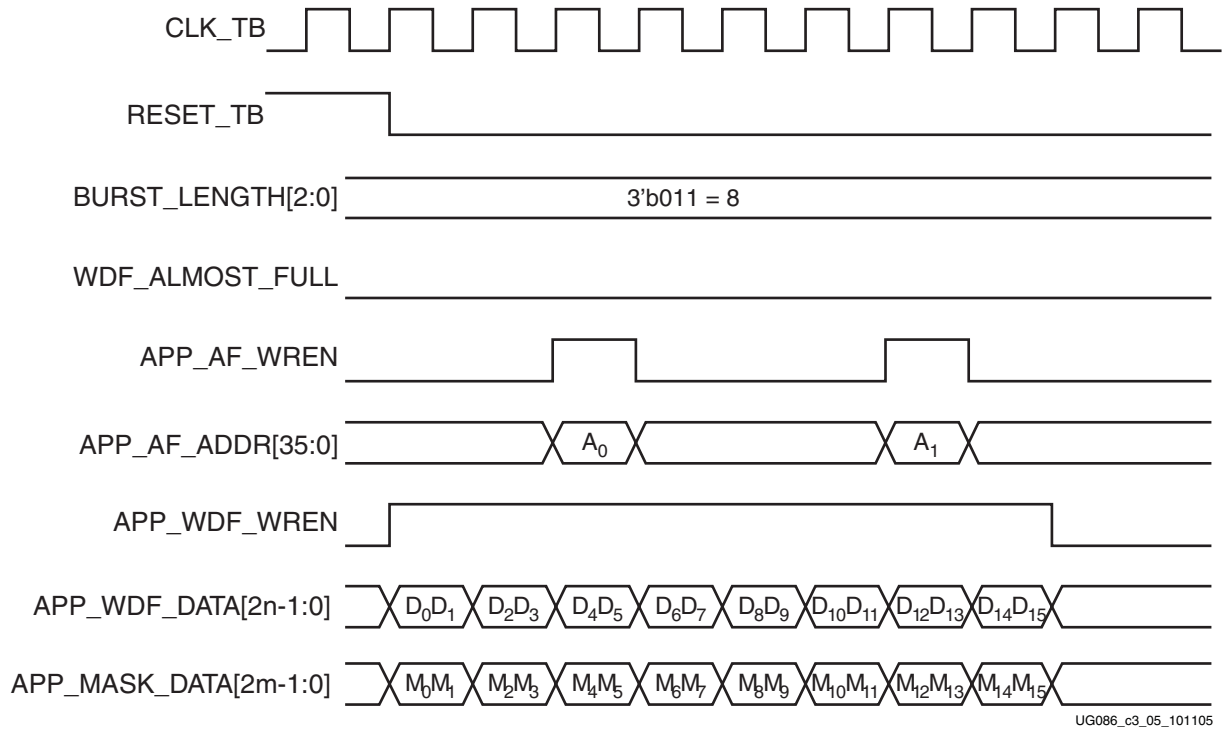


Figure 3-5: **DDR2 SDRAM Write Burst (BL = 8) for Two Bursts**

DDR2 SDRAM Reads

Figure 3-6 shows the timing diagram for a user read burst with burst length set to four. The user interface signals consist of AF_ALMOST_FULL, APP_AF_WREN, APP_AF_ADDR, READ_DATA_VALID, and READ_DATA_FIFO_OUT.

The user initiates the memory read by writing the read address to the Address FIFO when AF_ALMOST_FULL is deasserted (not full). The APP_AF_WREN signal is the write-enable signal for the Address FIFO, and APP_AF_ADDR is the memory read address. The READ_DATA_VALID signal is asserted when read data is available to the user. READ_DATA_FIFO_OUT is the read data from the memory to the user.

DDR2 SDRAM devices do not provide a read valid or read-enable signal along with read data. Consequently, the controller generates signals based on the CAS latency and the burst length.

The MIG tool adds one pattern_compare8 module per bank. The pattern_compare8 module delays the ctrl_rden signal and generates READ_EN_DELAYED_RISE and READ_EN_DELAYED_FALL signals, which are used as write enables for the Read Data FIFOs.

The MIG tool ensures that a DQS and its corresponding DQ signals do not cross a bank boundary.

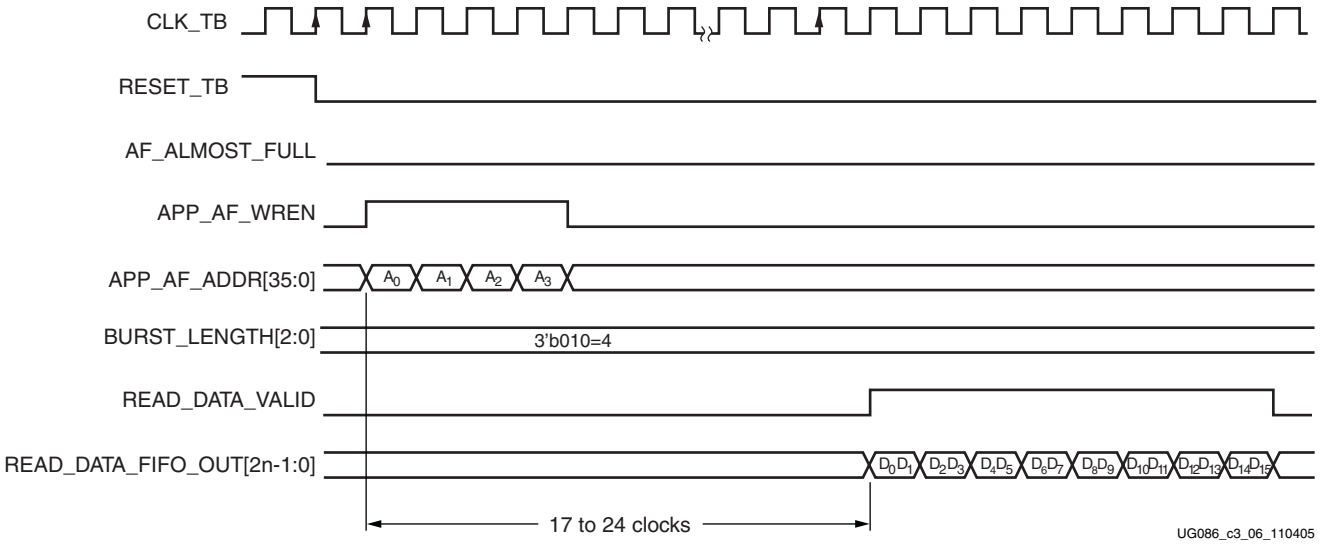


Figure 3-6: **DDR2 SDRAM Read Burst (BL = 4) for Four Bursts**

Figure 3-7 shows the timing diagram for a user read burst of two consecutive bursts with the burst length set to eight. The user interface signals are the same for burst lengths of four and eight.

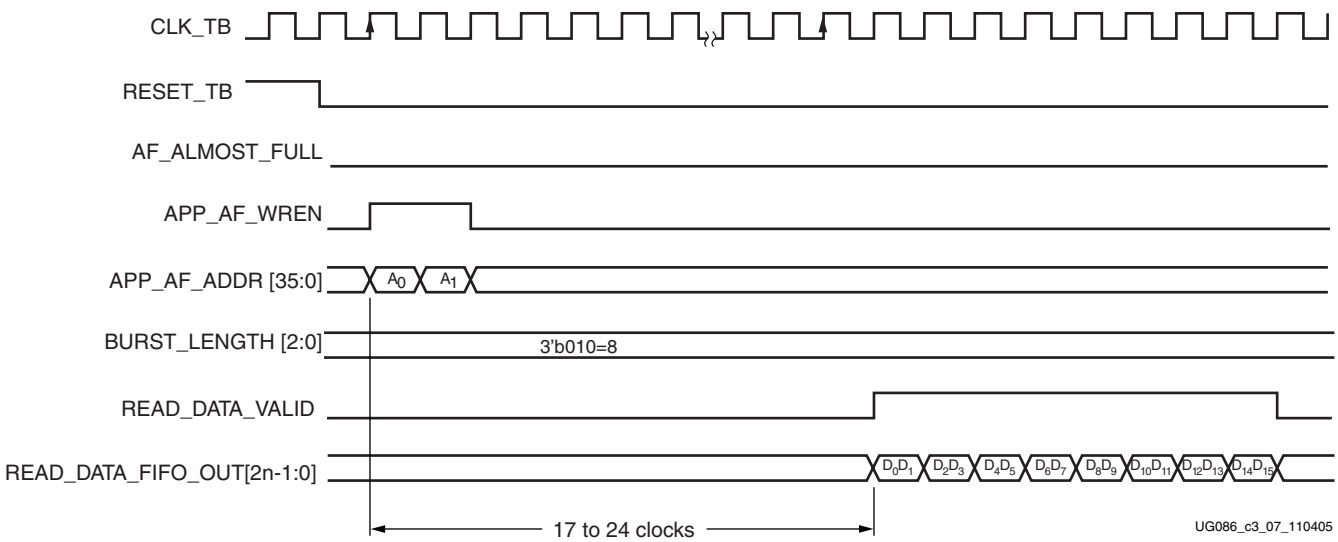


Figure 3-7: **DDR2 SDRAM Read Burst (BL = 8) for Two Bursts**

The 24 clocks from the read command to the read data are broken up as indicated in Table 3-4.

Table 3-4: **Read Command to Read Data Clock Cycles**

Parameter	Number of Clocks
Read Address to Empty Deassert	4 clocks
Empty to Active Command	5 clocks
Active to Read Command	4 clocks
Read Command to Read Data (the CAS latency is 3)	3 clocks
Read Data to IOB Output	2 clocks
IOB Output to Data Valid	6 clocks
Total:	**24 clocks**

User to Controller Interface

Table 3-5 lists the signals between the user interface and the controller.

Table 3-5: **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 3-8 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 4 clock cycles for a burst length of 8. Figure 3-8 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.

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

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]
```

Note: col_ap_width includes an autoprecharge (A10) bit 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 for autoprecharge. The col_ap_width parameter is used internally for prepending the A10 bit during the Precharge command. For a column address of 10, col_ap_width is defined as 11. For an 11-bit column address, col_ap_width is defined as 12.

Dynamic Command Request

Table 3-6 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 3-6: **Optional Commands**

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

Figure 3-8 describes four consecutive writes followed by four consecutive reads with a burst length of 8. Table 3-7 lists the state signal values for Figure 3-8.

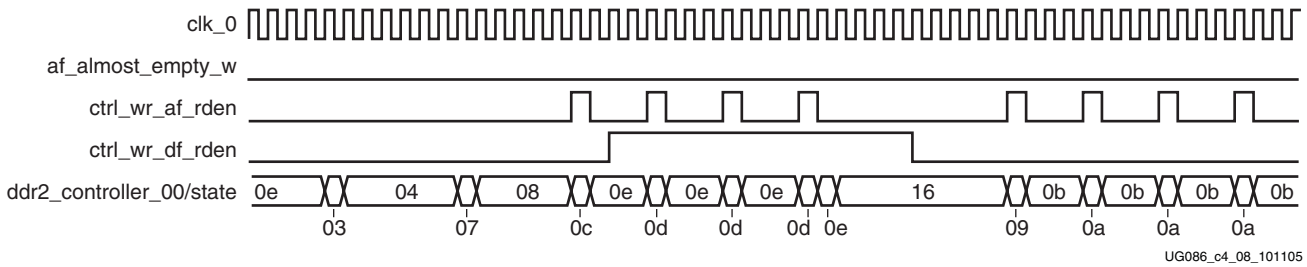


Figure 3-8: **Consecutive Writes Followed by Consecutive Reads with Burst Length of 8**

Table 3-7: **State Signal Values for Figure 3-8**

State Signal Value	Description
0c	First write
0d	Burst write
09	First read
0a	Burst read

Controller to Physical Layer Interface

Table 3-8 lists the signals between the controller and the physical layer.

Table 3-8: **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 3-9 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.

Table 3-8: **Signals Between the Controller and Physical Layer *(Continued)***

Port Name	Port Width	Port Description	Notes
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 3-9 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 3-9 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 3-9 describes the timing waveform for control signals from the controller to the physical layer. Table 3-9 lists the state signal values for Figure 3-9.

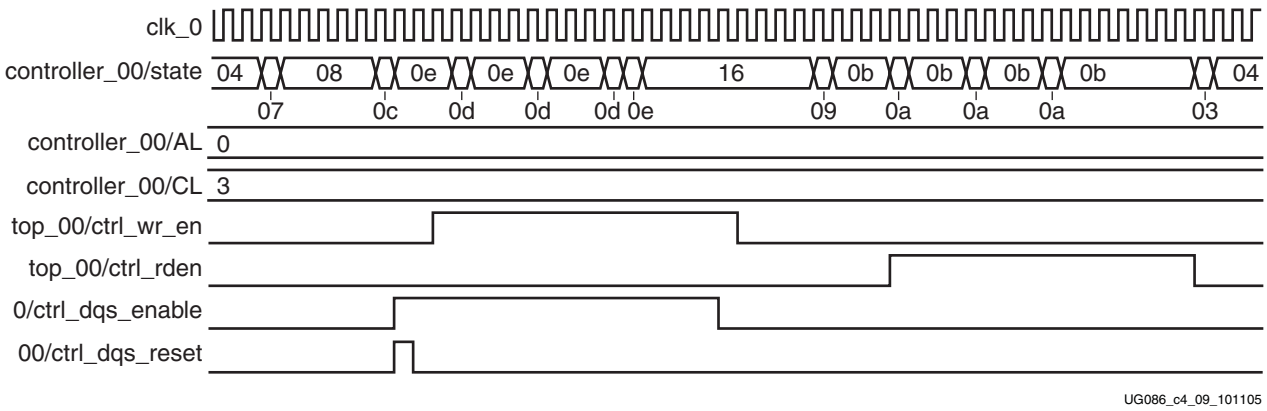


Figure 3-9: **Timing Waveform for Control Signals from the Controller to the Physical Layer**

Table 3-9: **State Signal Values for Figure 3-9**

State Signal Value	Description
0c	First write
0d	Burst write
09	First read
0a	Burst read

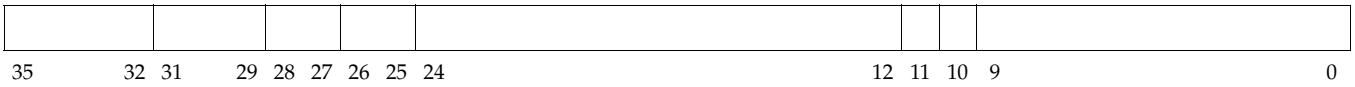
Deep Memory Configurations

Components

Case 1: 256 Mb (x4 component)

Density	256 Mb (256 Mb x 4 = 1 Gb)
Depth	4
Row address	13
Column address	11
Bank address	2
Rank/chip + deep address	2

Write Address/Read Address:

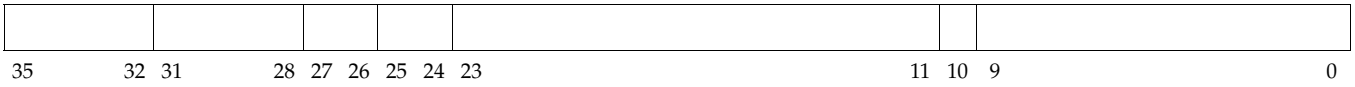


A11, A9-A0	Column address
A10	Auto precharge
A24-A12	Row address
A26-A25	Bank address
A28 -A27	Rank + deep address
A31-A29	Assigned to all zeros
A35-A32	Dynamic commands

Case 2: 256 Mb (x8 component)

Density	256 Mb (256 Mb x 3 = 768 Mb)
Depth	3
Row address	13
Column address	10
Bank address	2
Rank/chip + deep address	2

Write Address/Read Address:



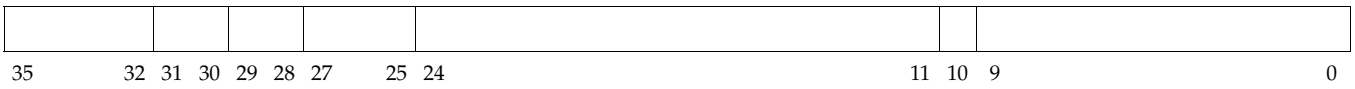
A9-A0	Column address
A10	Auto precharge
A23-A11	Row address
A25-A24	Bank address

A27-A26	Rank + deep address
A31-A28	Assigned to all zeros
A35-A32	Dynamic commands

Case 3: 256 Mb (x16 component)

Density	256 Mb (256 Mb x 2 = 512 Mb)
Depth	2
Row address	13
Column address	9
Bank address	2
Rank/chip + deep address	1

Write Address/Read Address:



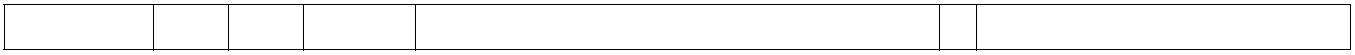
A8-A0	Column address
A10	Auto precharge
A23-A11	Row address
A25-A24	Bank address
A26	Rank + deep address
A9, A31-A27	Assigned to all zeros
A35-A32	Dynamic commands

DIMMs

Case 1: 4 GB

Density	2 GB (2 x 2 = 4 GB)
Depth	2
Row address	14
Column address	10
Bank address	3
Rank/chip + deep address	2

Write Address/Read Address:

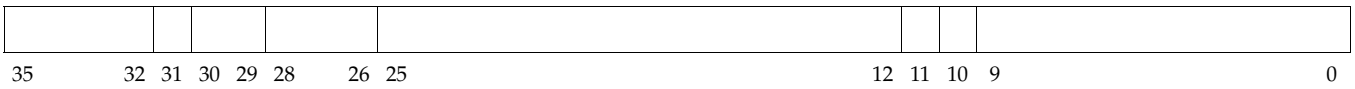


A9-A0	Column address
A10	Auto precharge
A24-A11	Row address
A27-A25	Bank address
A39-A28	Rank + deep address
A31-A30	Assigned to all zeros
A35-A32	Dynamic commands

Case 2: (8 GB)

Density	4 GB (4 x 2 = 8 GB)
Depth	2
Row address	14
Column address	11
Bank address	3
Rank/chip + deep address	2

Write Address/Read Address:



A11, A9-A0	Column address
A10	Auto precharge
A25-A12	Row address
A28-A26	Bank address
A30 - A29	Rank + deep address
A35 - A31	Dynamic commands

Table 3-10 shows the pin mapping for x4 registered DIMMs between the memory data sheet and the UCF file.

Table 3-10: **Pin Mapping for x4 DIMMs**

Memory Data Sheet	MIG1.x UCF
DQ[63:0]	DQ[63:0]
CB3 - CB0	DQ[67:64]
CB7 - CB4	DQ[71:68]
DQS0, $\overline{\text{DQS0}}$	DQS[0], DQS_N[0]
DQS1, $\overline{\text{DQS1}}$	DQS[2], DQS_N[2]
DQS2, $\overline{\text{DQS2}}$	DQS[4], DQS_N[4]
DQS3, $\overline{\text{DQS3}}$	DQS[6], DQS_N[6]
DQS4, $\overline{\text{DQS4}}$	DQS[8], DQS_N[8]
DQS5, $\overline{\text{DQS5}}$	DQS[10], DQS_N[10]
DQS6, $\overline{\text{DQS6}}$	DQS[12], DQS_N[12]
DQS7, $\overline{\text{DQS7}}$	DQS[14], DQS_N[14]
DQS8, $\overline{\text{DQS8}}$	DQS[16], DQS_N[16]
DQS9, $\overline{\text{DQS9}}$	DQS[1], DQS_N[1]
DQS10, $\overline{\text{DQS10}}$	DQS[3], DQS_N[3]
DQS11, $\overline{\text{DQS11}}$	DQS[5], DQS_N[5]
DQS12, $\overline{\text{DQS12}}$	DQS[7], DQS_N[7]
DQS13, $\overline{\text{DQS13}}$	DQS[9], DQS_N[9]
DQS14, $\overline{\text{DQS14}}$	DQS[11], DQS_N[11]
DQS15, $\overline{\text{DQS15}}$	DQS[13], DQS_N[13]
DQS16, $\overline{\text{DQS16}}$	DQS[15], DQS_N[15]
DQS17, $\overline{\text{DQS17}}$	DQS[17], DQS_N[17]

SerDes Clocking Interface

This technique uses the Input Serializer/Deserializer (ISERDES) and Output Serializer/Deserializer (OSERDES) features available in every Virtex-4 I/O. A DDR2 SDRAM interface is source-synchronous, where the read data and read data strobe are transmitted edge-aligned. To capture this transmitted data using Virtex-4 FPGAs, either the strobe or the data can be delayed. In this design, the read data is captured in the delayed strobe domain and recaptured in the FPGA clock domain in the ISERDES. The received signal, double data rate (DDR) read data, is converted to 4-bit parallel single data rate (SDR) data at half the frequency of the interface using the ISERDES. The write data and strobe transmitted by the FPGA use the OSERDES. The OSERDEDS converts 4-bit parallel data at half the frequency of the interface to DDR data at the interface frequency.

Feature Summary

This section summarizes the supported and unsupported features of the SerDes clocking DDR2 SDRAM controller design.

Supported Features

The DDR2 SDRAM controller design supports:

- Burst lengths of four and eight
- Sequential burst types
- CAS latencies of 4 and 5
- Differential DQS
- Different memories (density/speed)
- Components
- Additive latencies 0, 1, and 2

The supported features are described in more detail in "Architecture."

Unsupported Features

The DDR2 SDRAM controller design does not support:

- Interleaved burst types
- Single-ended DQS
- Redundant DQS (RDQS)
- Auto precharge
- On-die termination (ODT)
- DIMMs
- Deep memories (density/speed)
- ECC support

Architecture

Implemented Features

This section provides details on the supported features of the DDR2 SDRAM controller.

Burst Length

Bits M0:M3 of the Mode Register define the burst length. Read and write accesses to the DDR2 SDRAM are burst-oriented. The burst length is programmable to either four or eight. It determines the maximum number of column locations accessed for a given READ or WRITE command.

The DDR2 SDRAM ddr2_controller module implements burst lengths of four and eight.

CAS Latency

Bits M4:M6 of the Mode Register define the CAS latency (CL). CL is the delay in clock cycles between the registration of a READ command and the availability of the first bit of output data. CL can be set to four or five clocks.

CAS latency is implemented in the ddr2_controller module. During data write operations, the generation of the ctrl_wr_en, ctrl_wr_dis, ctrl_wr_en_90, ctrl_wr_dis_90, and ctrl_Odd_Latency signals varies according to the CL in the ddr2_controller module. During read data operations, the ctrl_RdEn_div0 signal is generated with each read command and aligned with read data.

Components

Supported memory components have type Micron, 256 Mb, 512 Mb, and 1 Gb densities, and speed grades of -3 and -37E. The different speed grades have different timing parameters. Table 3-11 lists the timing parameters for the components.

Table 3-11: **Timing Parameters for Components**

Parameter	Description	Micron 256 Mb		Micron 512 Mb		Micron 1 Gb		Units
		37E	3	37E	3	37E	3	
T_{MRD}	LOAD MODE Command Cycle Time	2	2	2	2	2	2	T_{CK}
T_{RP}	PRECHARGE Command Period	15	15	15	15	15	15	ns
T_{RFC}	REFRESH to ACTIVE or REFRESH to REFRESH Command Interval	75	75	105	105	127.5	127.5	ns
T_{RCD}	ACTIVE to READ or WRITE Delay	15	15	15	15	15	15	ns
T_{RAS}	ACTIVE to PRECHARGE Command	40	40	40	40	40	40	ns
T_{RC}	ACTIVE to ACTIVE (Same Bank) Command	55	55	55	55	55	55	ns
T_{RTP}	READ to PRECHARGE Command Delay	7.5	7.5	7.5	7.5	7.5	7.5	ns
T_{WTR}	WRITE to READ Command Delay	7.5	7.5	7.5	7.5	7.5	7.5	ns
T_{WR}	WRITE Recovery Time	15	15	15	15	15	15	ns

Hierarchy

Figure 3-10 shows the hierarchical structure of the modular design. The physical and control layers are clearly separated in this figure. The MIG 1.5 tool generates the entire DDR2 SDRAM controller as shown in this hierarchy, including the test bench.

Figure 3-10: **Hierarchical Structure of the DDR2 SDRAM Design (SerDes Clocking)**

DDR2 Controller Submodules

Figure 3-11 is a detailed block diagram of the DDR2 SDRAM controller. The five blocks shown are the sub-blocks of the top module. The functions of these blocks are explained in the subsections following Figure 3-11.

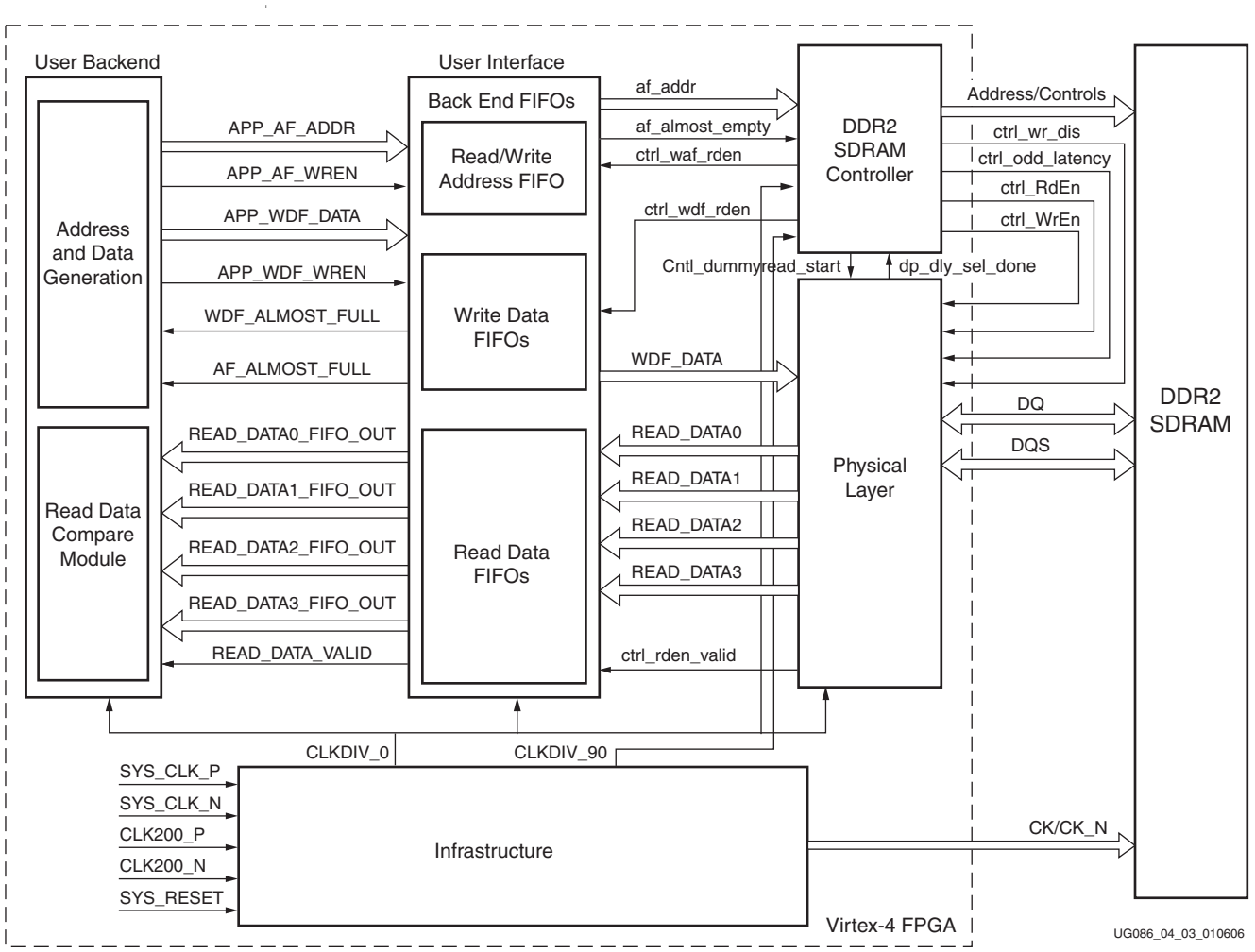


Figure 3-11: **DDR2 Memory Controller Block Diagram (SerDes Clocking)**

ddr2_controller

The DDR2 SDRAM ddr2_controller accepts and decodes user commands and generates read, write, and refresh commands. The DDR2 SDRAM controller also generates signals for other modules. The memory is initialized and powered up using a defined process. The controller state machine handles the initialization process upon power-up. When the initialization is over, the controller starts doing a dummy write and continuous dummy reads until the dp_dly_slct_done signal is asserted. After this signal is asserted, the controller starts writing to and reading from the memory.

The ddr2_controller is clocked at half the frequency of the interface using CLKDIV_0 and CLKDIV_90. Therefore the address and bank address are driven and the command signals (RAS_L, CAS_L, and WE_L) are asserted for two clock cycles of the fast memory interface clock. The control signals (CS_L, CKE, and ODT) are DDR of the half frequency clock CLKDIV_0, ensuring that the control signals are asserted for just one clock cycle of the fast memory interface clock. Figure 3-12 shows the command and control timing diagram.

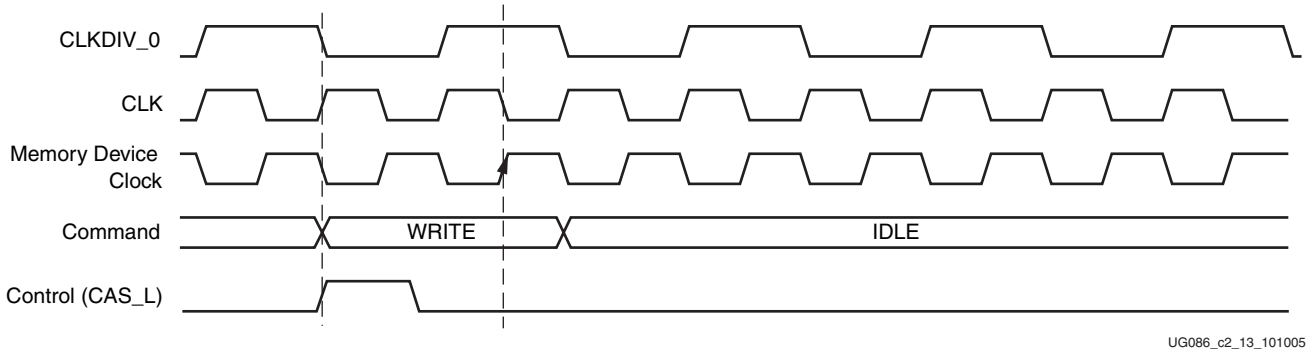


Figure 3-12: **Command and Control Timing from Controller to DDR2 Memory**

Datapath

This module transmits data to and receives data from the memories. Its major functions include processing the data in the write datapath, and calibrating the data in the read datapath. The write datapath function is implemented in the data_write module and the read datapath function is implemented in the tap_ctrl, data_tap_inc, Idelay_ctrl, and idelay_rd_en_io modules.

To start calibration in the read datapath, the write datapath first generates the training pattern (known data) and writes it to the memory during dummy writes. Calibration is done during the dummy reads. The read datapath expects the training pattern. When the received training pattern is correct, then DQ and DQS are aligned with the FPGA clock to capture the data without errors during actual writes and reads. After this calibration is finished, dp_dly_slct_done is asserted, which initiates writes and reads to the memory.

User Interface

This module stores write data and write addresses, writes the data into a location specified by the write address, stores read addresses used to read from a specific location, and also stores data read from the memory in FIFOs. The rd_data and rd_data_fifos modules store the data in LUT-based RAMs. The rd_wr_addr_fifo and wr_data_fifo modules store the data and address in block RAMs.

The width of the wr_data_fifo module is four times the data width, because the data corresponding to four edges is given in one clock cycle.

Infrastructure Module

The infrastructure module generates the necessary FPGA clocks and reset signals. The clocking scheme used for this design includes one digital clock manager (DCM) and two phase-matched clock drivers (PMCDs) as shown in Figure 3-13.

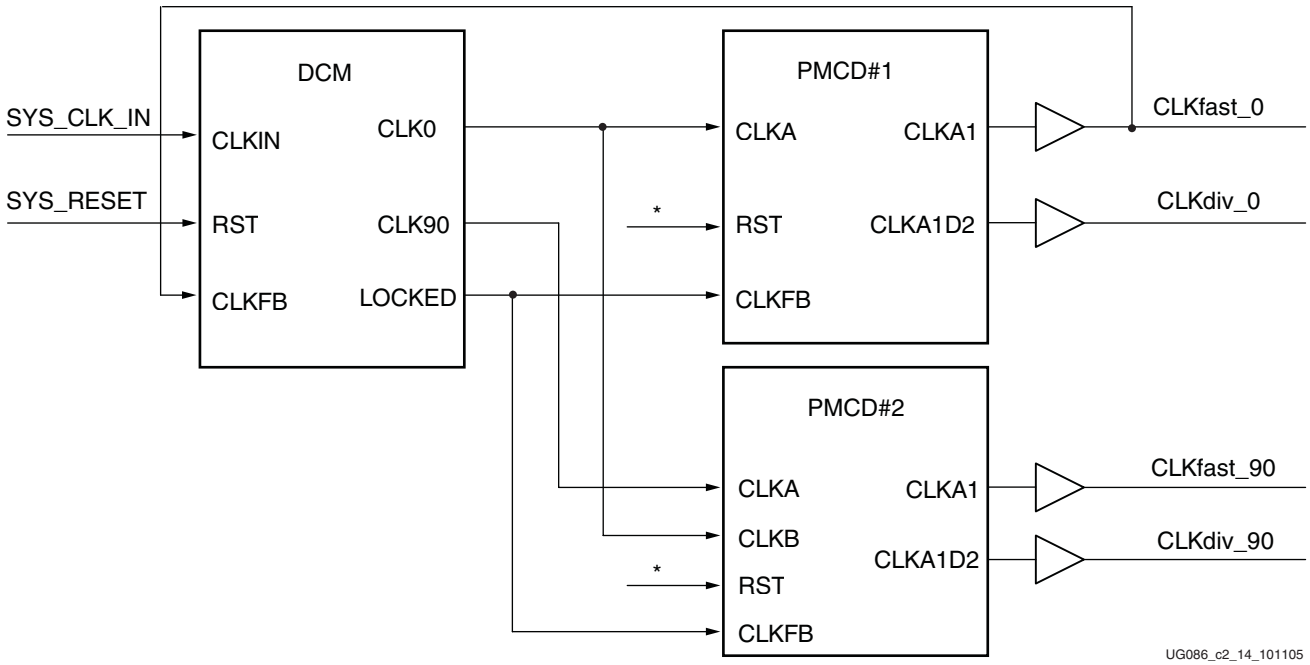


Figure 3-13: **Clocking Scheme for the High-Performance Memory Interface Design**

IOBS Module

All FPGA input and output signals that go to and come from memory are implemented in the IOBS module. These signals correspond to address, control, data, and clocks. This module comprises three submodules: controller_iobs, infrastructure_iobs, and data_path_ios. The controller_iobs module contains all the IOBs for all address and control signals. The infrastructure_iobs module contains all the IOBs for clock signals to memory. The data_path_ios module contains all the IOBs for the data, dqs, mask, and read enable signals.

DDR2 SDRAM System and User Interface Signals

Table 3-12 lists the system signals that are required for the design.

Table 3-12: **DDR2 SDRAM System Signals**

Signal Name	Direction	Description
SYS_CLK_P, SYS_CLK_N	Input	These differential input clocks generate the single-ended clock to the input of the DCM. Memory operates at this frequency, but the ddr2_controller, data_path, and user_interface modules, and all other FPGA slice logic are clocked at half of this frequency.
CLK200_P, CLK200_N	Input	Differential clock used in the idelay_ctrl logic.
SYS_RESET	Input	Active-Low reset to the design.

Table 3-13 describes the DDR2 SDRAM user interface signals.

Table 3-13: **DDR2 SDRAM Controller User Interface Signals**

Signal Name	Direction	Description
CLKDIV_0	Output	All user interface signals must be synchronized with respect to the negative edge of CLKDIV_0.
RESET0	Output	Reset signal for the User Interface.
BURST_LENGTH[2:0]	Output	This signal determines the data burst length for each write address. 0b010 for burst length = 4 0b100 for burst length = 8
WDF_ALMOST_FULL	Input	This signal indicates the ALMOST_FULL status of the Write Data FIFO. When this signal is asserted, the user stops writing data into the Write Data FIFO and addresses into the Write Address FIFO.
APP_WDF_DATA[4n-1:0]	Input	User write data to the memory, where *n* indicates the data width of the memory. The user data is four times the data width of the memory. This bus has the data for two rising edges and two falling edges. The most-significant bits contain the second falling-edge data, and the least-significant bits contain the first rising-edge data.
APP_MASK_DATA[4m-1:0]	Input	User mask data to the memory, where *m* indicates the data mask width of the memory. The mask data is four times the mask width of the memory. This bus also has the mask data for four edges. The most-significant bits contain the mask data for the second falling edge, and the least-significant bits contain the mask data for the first rising edge.
APP_WDF_WREN	Input	Write Enable signal to the Write Data FIFO.
AF_ALMOST_FULL	Output	This signal indicates the FULL status of the Address FIFO. When this signal is asserted, the user stops writing addresses into the Address FIFO.
APP_AF_ADDR[35:0]	Input	The user address consists of a memory address and dynamic commands. The address width [31:0] is the memory read/write address, which includes the column, row, bank, and chip address. The address width [35:32] represents dynamic commands. 000 – Load Mode Register 001 – Auto Refresh 010 – Precharge All 011 – Active 100 – Write 101 – Read
APP_AF_WREN	Input	Write Enable signal to the Address FIFO.

Table 3-13: **DDR2 SDRAM Controller User Interface Signals *(Continued)***

Signal Name	Direction	Description
READ_DATA0_FIFO_OUT[n-1:0] READ_DATA1_FIFO_OUT[n-1:0] READ_DATA2_FIFO_OUT[n-1:0] READ_DATA3_FIFO_OUT[n-1:0]	Output	The read data captured from the memory is four parallel *n*-bit data buses, each at half the frequency of the interface, where *n* indicates the data width of the memory. READ_DATA0_FIFO_OUT is the first rising-edge data, READ_DATA1_FIFO_OUT is the second rising-edge data, READ_DATA2_FIFO_OUT is the first falling-edge data, and READ_DATA3_FIFO_OUT is the second falling-edge data.
READ_DATA_VALID	Output	This signal is asserted to indicate the read data is available to the user.

Notes:

1. All user interface signal names are prepended with a controller number. DDR2 SDRAM devices support multicontroller operation, where a maximum of eight controllers can be selected by the user from the MIG 1.5 tool. For example, when the user selects eight controllers, the signal names have the following format: cntrl0_user_signal, cntrl1_user_signal, cntrl2_user_signal, cntrl3_user_signal, cntrl4_user_signal, cntrl5_user_signal, cntrl6_user_signal, and cntrl7_user_signal.

DDR2 SDRAM Memory Writes

Figure 3-14 shows the timing diagram for a user write burst to the DDR2 SDRAM with BURST_LENGTH set to 3'b010 for a burst length of four. The user interface signals consist of WDF_ALMOST_FULL, AF_ALMOST_FULL, APP_AF_WREN, APP_AF_ADDR, APP_WDF_WREN, APP_MASK_DATA, and APP_WDF_DATA. The user interface consists of an Address FIFO and a Write Data FIFO.

The user initiates a write to memory by writing to the User Write Data FIFO and User Write Address FIFO when WDF_ALMOST_FULL and AF_ALMOST_FULL are deasserted. The Write Data FIFO is full when WDF_ALMOST_FULL is asserted. The APP_AF_WREN signal is the write-enable signal for the Address FIFO, and APP_AF_ADDR is the write address to the Address FIFO. The APP_WDF_WREN signal is the write-enable signal for the Write Data FIFO, and APP_WDF_DATA is the user write data to the memory. APP_WDF_DATA is four times the data width of the memory, consisting four edges of data (two rising edges and two falling edges).

For burst lengths of four, each write address consists of one write data cycle. Figure 3-14 shows a write burst with four consecutive bursts.

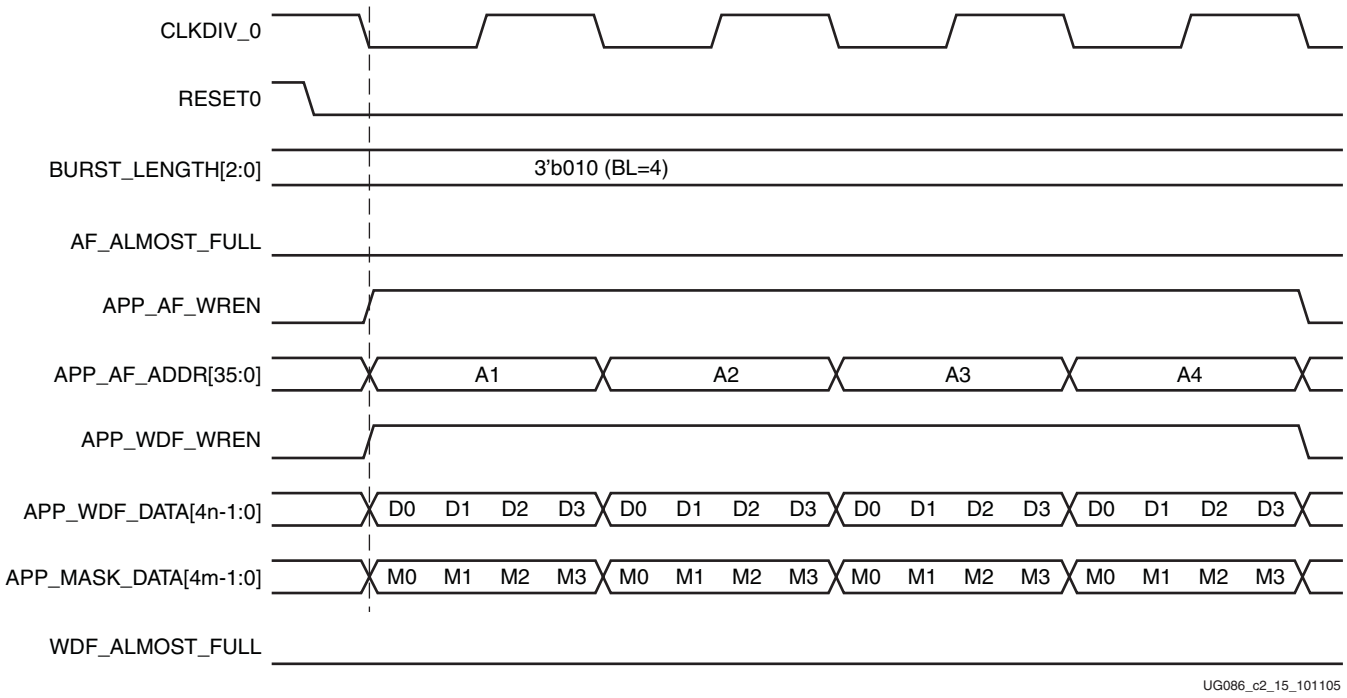


Figure 3-14: **DDR2 SDRAM Write Burst (BL = 4) for Four Bursts**

Figure 3-15 shows the timing diagram for a write burst with burst length set to eight. The user interface signals are the same for burst lengths of four and eight. When the burst length is eight, the user writes two write data cycles for each address. The value of BURST_LENGTH is 0b100 for a burst length of eight.

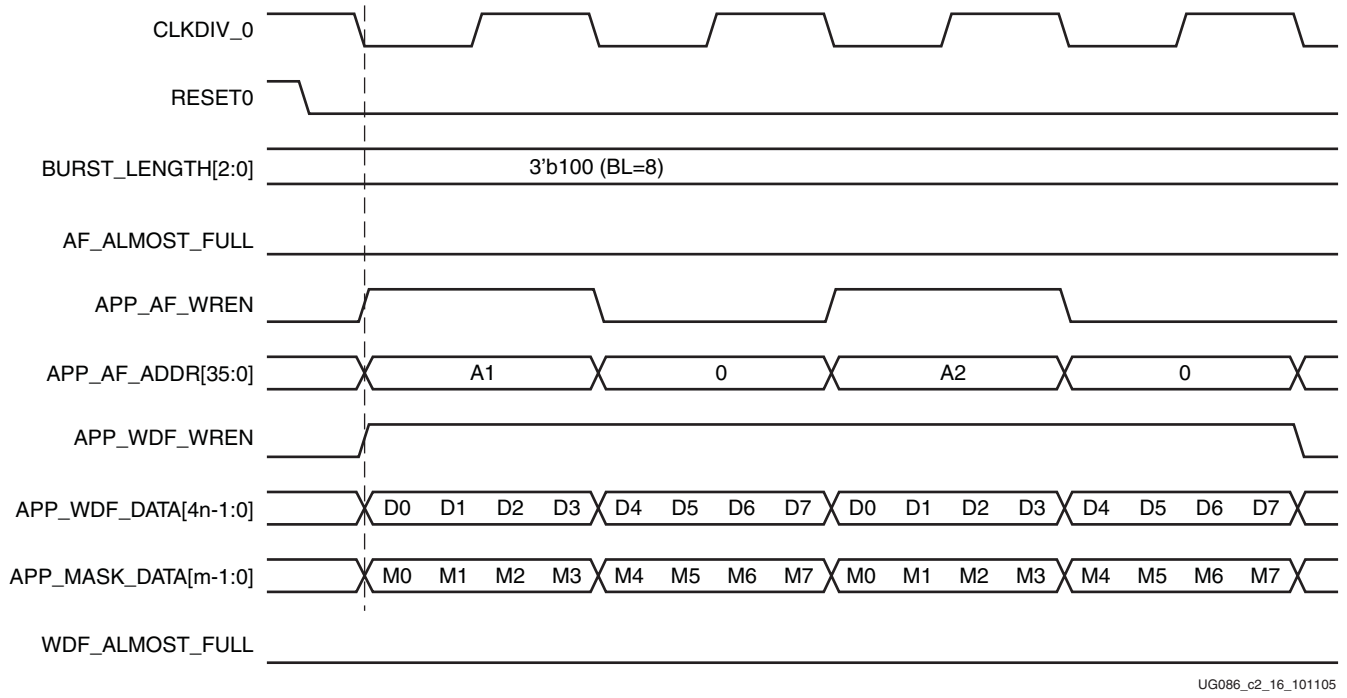


Figure 3-15: **DDR2 SDRAM Write Burst (BL = 8) for Two Bursts**

DDR2 SDRAM Memory Reads

Figure 3-16 shows the timing diagram for a user read burst with burst length set to four. The user interface signals consist of AF_ALMOST_FULL, APP_AF_WREN, APP_AF_ADDR, READ_DATA_VALID, READ_DATA0_FIFO_OUT, READ_DATA1_FIFO_OUT, READ_DATA2_FIFO_OUT, and READ_DATA3_FIFO_OUT.

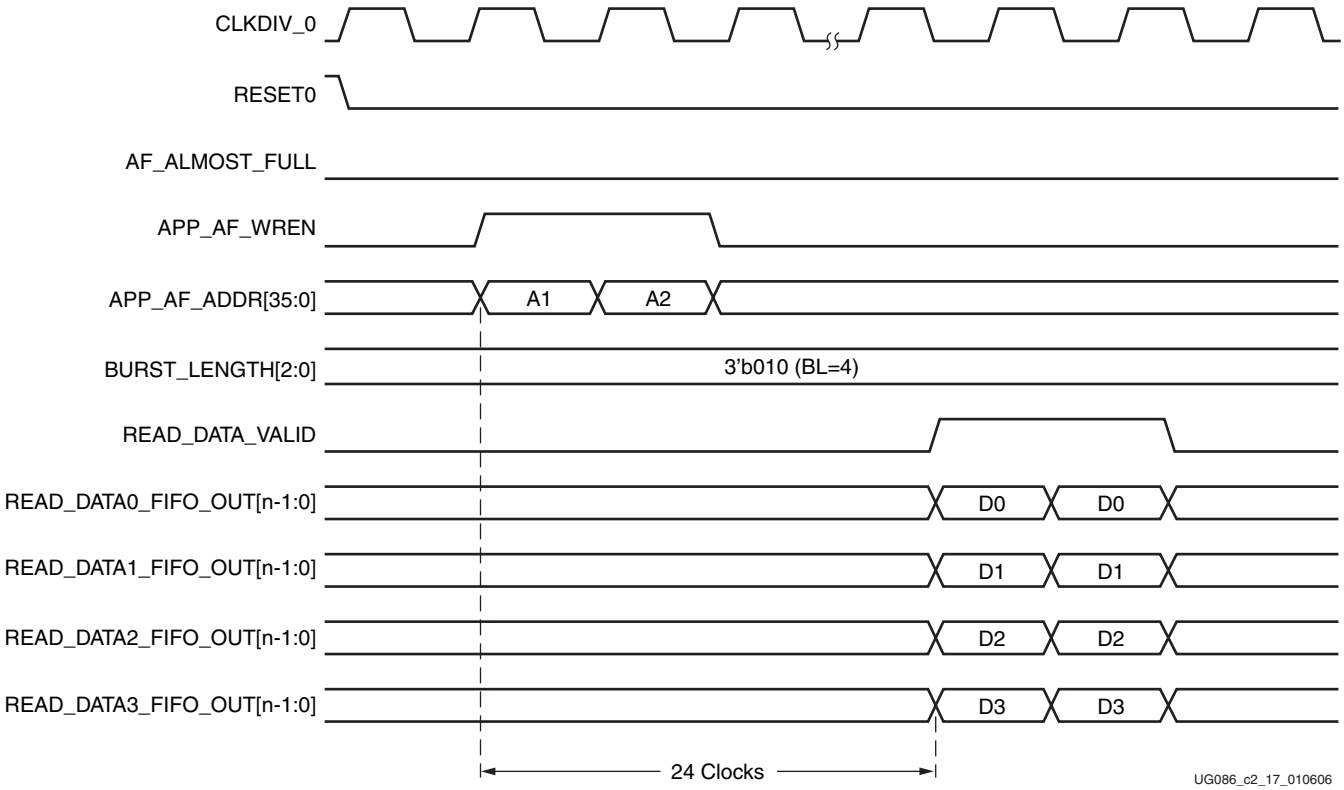


Figure 3-16: **DDR2 SDRAM Read Burst (BL = 4) for Two Bursts**

The user initiates the memory read by writing the read address to the Address FIFO when AF_ALMOST_FULL is deasserted (not full). The APP_AF_WREN signal is the write-enable signal for the Address FIFO, and APP_AF_ADDR is the memory read address. The READ_DATA_VALID signal is asserted when read data is available to the user. This read data is four parallel data buses (READ_DATA0_FIFO_OUT, READ_DATA1_FIFO_OUT, READ_DATA2_FIFO_OUT, READ_DATA3_FIFO_OUT), where the width of each is equal to the memory data width and each data corresponds to one edge.

DDR2 SDRAM devices do not provide a read valid or read-enable signal along with read data. Consequently, after the power-up calibration is done, dummy reads are done to set up the delay between the read command and read data from the memory. While these dummy reads are in progress, the read enable is generated with each read command and is delayed until the read data matches the write data. This delay includes CAS latency, trace delay, and path delay. This precalculated delay is used for asserting the read-enable signals that latch the data into the Read Data FIFOs. The delays are calculated on a per DQS basis. For example, if a bank has two DQS signals, there are two read enables used to latch the read data to the FIFOs.

The strobe (DQS), data (DQ), and clock (CK/$\overline{\text{CK}}$) signals should be matched in trace length from the FPGA to the memory device.

The MIG tool ensures that a DQS and its corresponding DQ signals do not cross a bank boundary.

Figure 3-17 shows the timing diagram for a user read burst of two consecutive bursts with the burst length set to eight. The user interface signals are the same for burst lengths of four and eight.

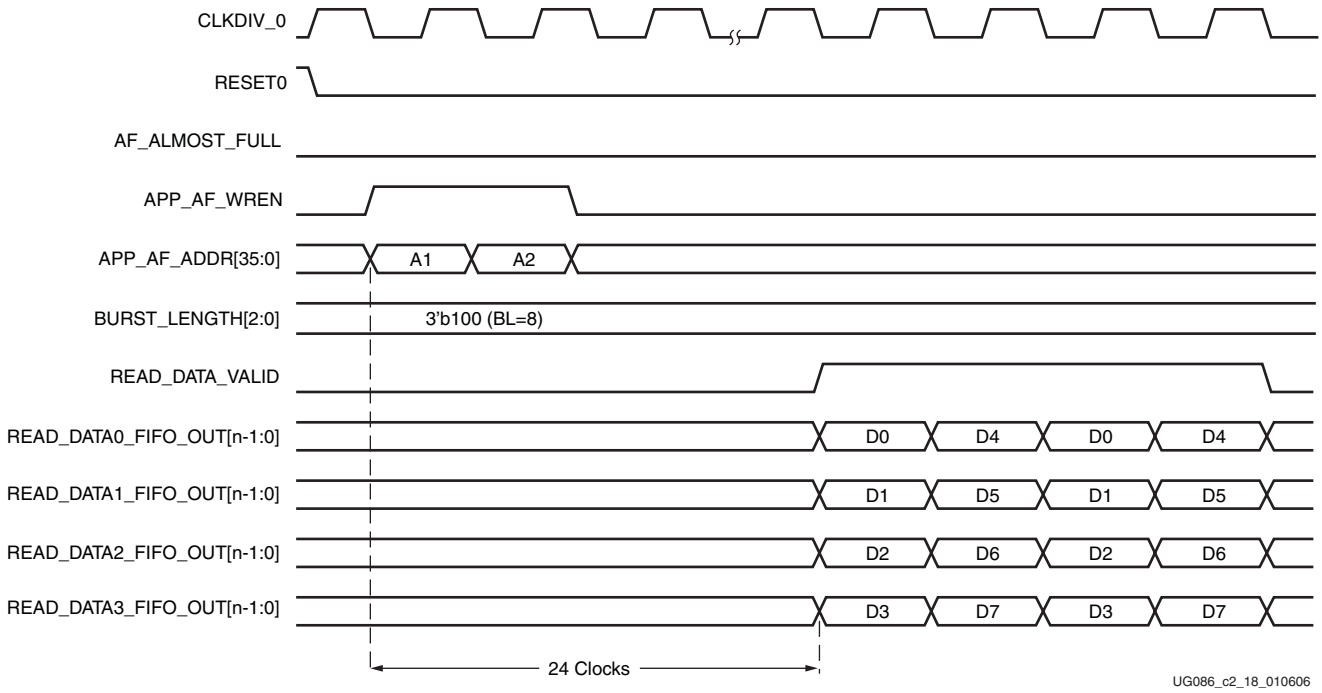


Figure 3-17: **DDR2 SDRAM Read Burst (BL = 8) for Two Bursts**

Table 3-14 shows how the 24 clocks from the read command to the read data are broken up.

Table 3-14: **Read Command to Read Data Clock Cycles**

Parameter	Number of Clocks (CLKDIV_0)
User Read Address to FIFO Input Read Address	2 Clocks
FIFO Input Read Address to Output Read Address	5 Clocks
FIFO Output Read Address to Active Command	6 Clocks
Active to Read Command	3 Clocks
Read Command to Read Data (the CAS Latency is 4)	2 Clocks
Read Data to IOB Output	2 Clocks
IOB Output to Data Valid	4 Clocks
Total:	**24 Clocks**

User to Controller Interface

Table 3-15 lists the signals between the user interface and the controller.

Table 3-15: **List of Signals Between User Interface and Controller**

Port Name	Port Width	Port Description	Notes
waf_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: [32] Dynamic Command Request: [35:33]	Monitor FIFO-full status flag to write address into the Address FIFO
af_almost_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_waf_RdEn	1	Read Enable input to Address FIFO in the user interface	This signal is asserted for one CLKDIV_0 clock cycle when the controller state is write, read, Load Mode register, Precharge All, Auto Refresh, or Active resulting from dynamic command requests. Figure 3-18 shows the timing waveform for a burst length of 8 with two back-to-back writes followed by two back-to-back reads.
ctrl_wdf_Rden	1	Read Enable input to Write Data FIFO in the user interface	The controller asserts this signal one CLKDIV_0 clock cycle after the first write state. This signal remains asserted for one clock cycle for a burst length of 4 and two clock cycles for a burst length of 8. Figure 3-18 shows the timing waveform. Sufficient data must be available in the 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 four 64-bit data words in the Write Data FIFO for every write address before issuing the write command.

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

Column Address

```
['column_address - 1:0]
```

Row Address

```
[(row_address + 'column_address) - 1:'column_address]
```

Bank Address

```
[('bank_address + 'row_address + 'column_address) -
 1:('column_address + 'row_address)]
```

Chip Select

```
['cs_width + 'bank_address + 'row_address + 'column_address -
 1:'bank_address + 'row_address + 'column_address]
```

Dynamic Command Request

Table 3-16 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 3-16: **Optional Commands**

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

Figure 3-18 describes two consecutive writes followed by two consecutive reads with a burst length of 8. Table 3-17 lists the state signal values for Figure 3-18.

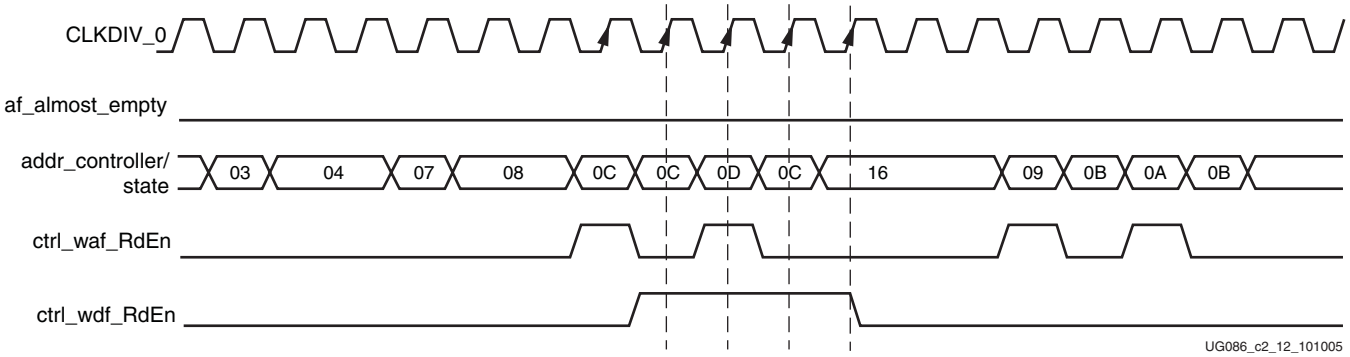


Figure 3-18: **Controller Read of Command and Data from User Interface FIFOs for a Burst Length of 8**

Table 3-17: **State Signal Values for Figure 3-18**

State Signal Value	Description
03	precharge
04	precharge_wait
07	active
08	active_wait
09	first_read
0a	burst_read
0b	read_wait
0c	first_write
0d	burst_write
0e	write_wait
16	write_read

Controller to Physical Layer Interface

Table 3-18 lists the signals between the controller and the physical layer.

Table 3-18: **Signals Between the Controller and Physical Layer**

Signal Name	Signal Width	Signal Description	Notes
ctrl_wren	1	Output from the controller to the write datapath. Write DQS and DQ generation begins when this signal is asserted.	Asserted for two CLKDIV_0 cycles for a burst length of 4 and three CLKDIV_0 cycles for burst length of 8. Asserted one CLKDIV_0 cycle earlier than the WRITE command for CAS latency values of 4 and 5.
ctrl_wr_dis	1	Output from the controller to the write datapath. Write DQS and DQ generation ends when this signal is asserted.	Asserted for one CLKDIV_0 cycle for a burst length of 4 and two CLKDIV_0 cycles for burst length of 8. Asserted one CLKDIV_0 cycle earlier than the WRITE command for CAS latency values of 4 and 5.
ctrl_odd_latency	1	Output from the controller to the write datapath. Asserted when the selected CAS latency is an odd number. Required for generation of write DQS and DQ after the correct latency (CAS latency – 1).	

Table 3-18: **Signals Between the Controller and Physical Layer *(Continued)***

Signal Name	Signal Width	Signal Description	Notes
ctrl_RdEn_div0	1	Output from the controller to the datapath generated with each read command. This is delayed by the precalculated amount and is used as a write enable to the read data capture FIFOs.	This signal is asserted for one CLKDIV_0 clock cycle for a burst length of 4 and two clock cycles for a burst length of 8.
ctrl_dummyread_start	1	Output from the controller to the write datapath. When this signal is asserted, the strobe and data calibration begin.	This signal must be asserted when valid read data is available on the read data bus. This signal is deasserted when the dp_dly_slct_done signal is asserted.
dp_dly_slct_done	1	Output from the read datapath to the controller indicating the strobe and data calibration are complete.	This signal is asserted when the data and strobe are calibrated. Normal operation begins after this signal is asserted.

Figure 3-19 describes the timing waveform for control signals from the controller to the physical layer with a CAS latency of 4 and an additive latency of 0.

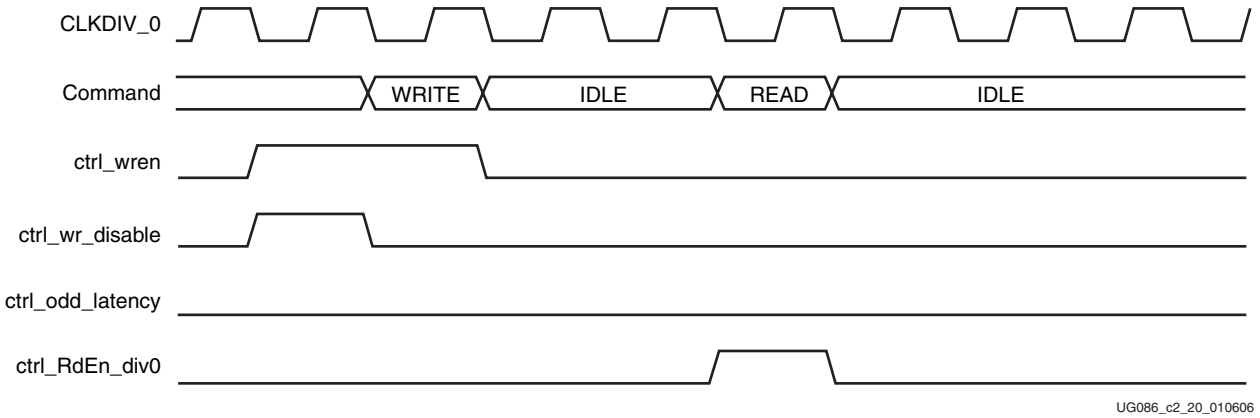


Figure 3-19: **Timing Waveform for Control Signals from the Controller to the Physical Layer**

Chapter 4

Implementing QDRII SRAM Controllers

This chapter describes how to implement QDRII SRAM interfaces for Virtex-4 FPGAs created with the MIG 1.5 design tool. This design is based on XAPP703 [Ref 12].

Feature Summary

This section summarizes the supported and unsupported features of the QDRII controller design.

Supported Features

The QDRII controller design supports:

- A maximum frequency of 250 MHz
- 9-bit, 18-bit, 36-bit, and 72-bit data widths
- Burst lengths of two and four
- Implementation using different Virtex-4 devices
- Operation with any 9-bit, 18-bit, and 36-bit memory component
- Verilog and VHDL

Architecture

Figure 4-1 shows the top-level block diagram of the QDRII memory controller. One side of the QDRII memory controller connects to the user interface denoted as Block Application. The other side of the controller interfaces to QDRII memory. The memory interface data width is selectable.

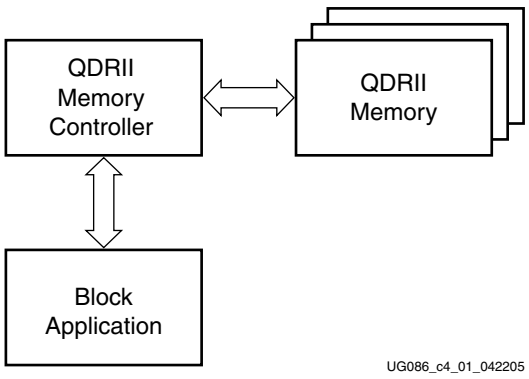


Figure 4-1: **QDRII Memory Controller**

Data is double-pumped to QDRII SRAM on both the positive and the negative clock edges. The HSTL_18 Class I I/O standard is used for the data, address, and control signals.

QDRII SRAM interfaces are source-synchronous and double data rate like DDR SDRAM interfaces.

The key advantage to QDRII devices is they have separate data buses for reads and writes to SRAM.

Interface Model

The memory interface is layered to simplify the design and make the design modular. Figure 4-2 shows the layered memory interface in the QDRII memory controller. The three layers are the application layer, the implementation layer, and the physical layer.

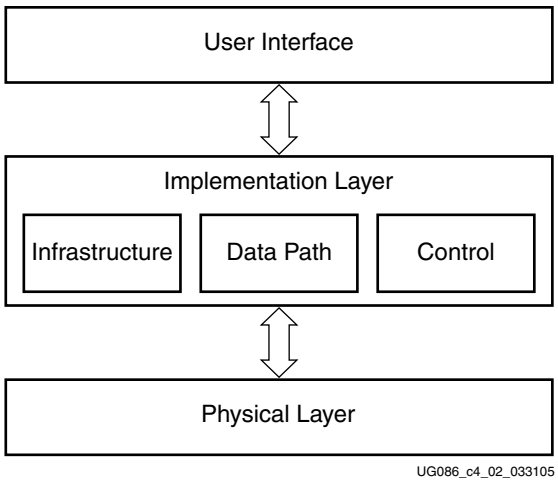


Figure 4-2: **Interface Layering Model**

The application layer creates the user interface, which initiates memory writes and reads by writing data and memory addresses to the User Interface FIFOs. The QDRII memory controller monitors the User Interface FIFOs and initiates data transfers with QDRII memory accordingly.

Hierarchy

Figure 4-3 shows the QDRII SRAM controller hierarchy.

Figure 4-3: **QDRII SRAM Controller Hierarchy**

QDRII Memory Controller Modules

Figure 4-4 shows a detailed block diagram of the QDRII memory controller. The four blocks shown are sub-blocks of the top module. The functionalities of these blocks are explained in the subsections following the figure.

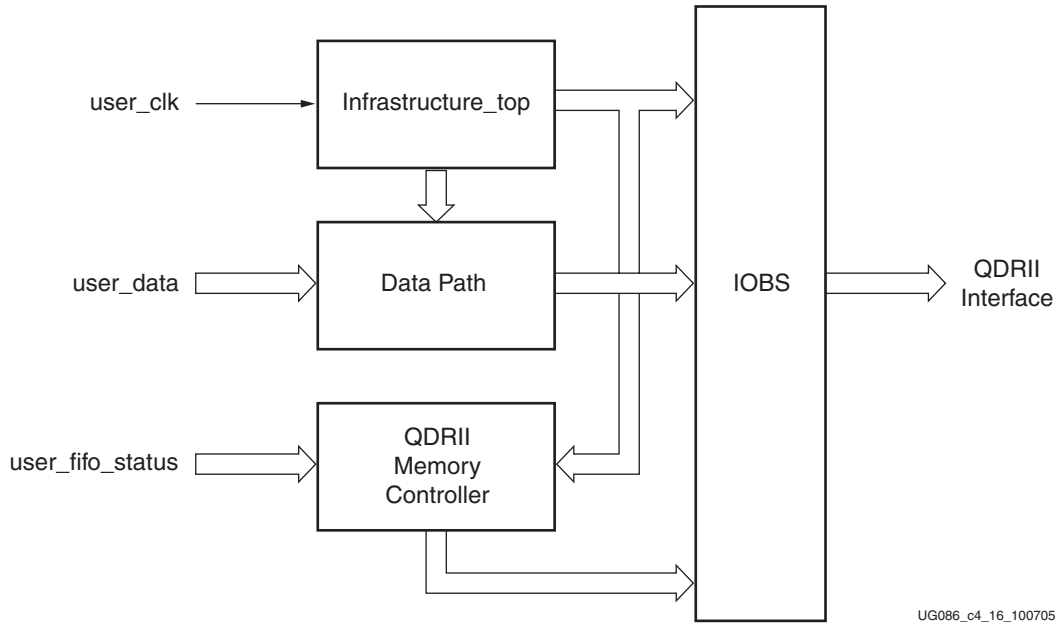


Figure 4-4: **QDRII Memory Controller Modules**

Figure 4-5 shows the QDRII memory controller modules with a 36-bit interface.

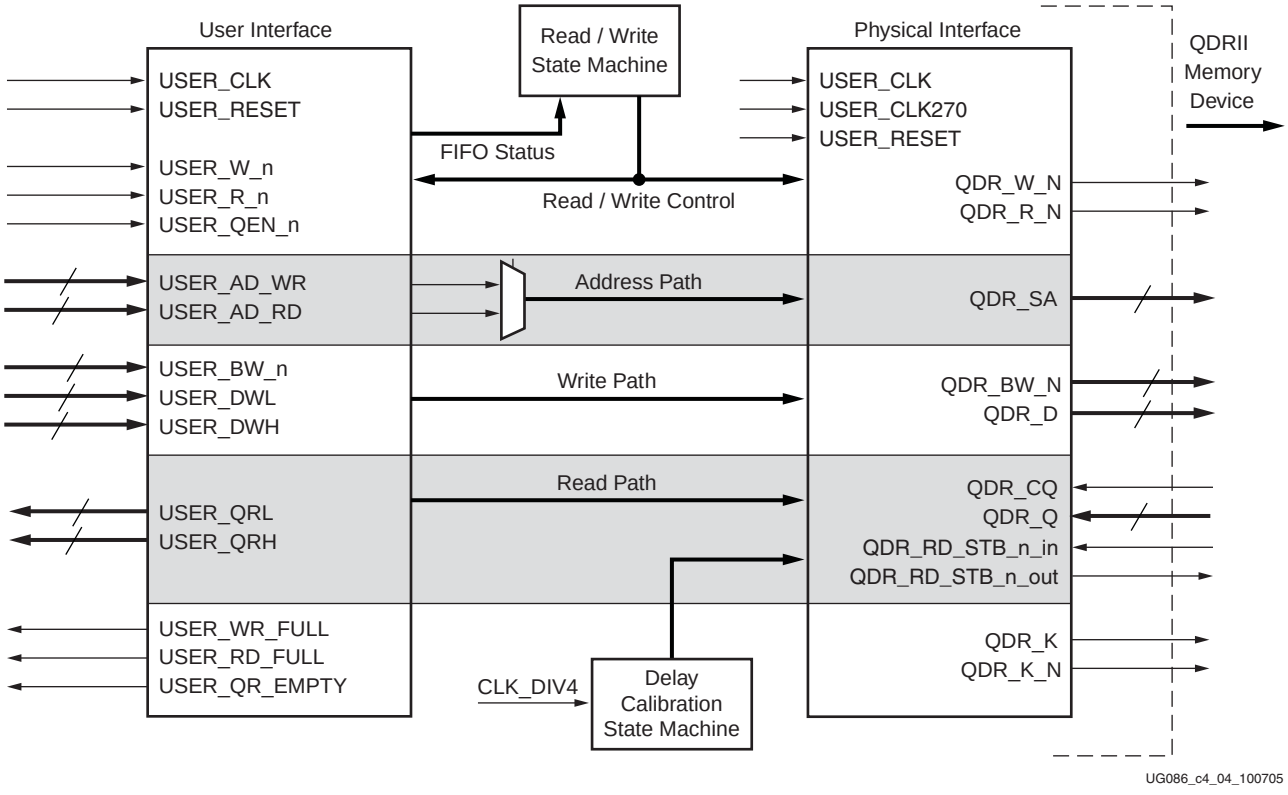


Figure 4-5: **QDRII Memory Controller Modules**

Controller

The QDRII memory controller initiates memory writes when the User Write Data FIFO and the User Write Address FIFO are not empty. The user writes the data and the address into the User Write Data FIFO and User Write Address FIFO, respectively. This data is written into memory.

The user initiates memory read commands when the User Read Data FIFOs are not full and the User Read Address FIFO is not empty. The memory address from which data is to be read is written into the User Read Address FIFO. The User Read Data FIFOs store the data captured from memory. The user can access the data read from memory by reading the User Read Data FIFOs.

When the User Write Data FIFO is not empty, the QDRII memory controller generates a write-enable signal for memory. When the write-enable signal is asserted, data is transferred to memory. The QDRII memory controller also generates a read-enable signal for memory when the User Address Data FIFO is not empty and the User Read Data FIFOs are not full. When the read-enable signal is asserted, data read from memory is captured and stored in the User Read Data FIFOs.

Figure 4-6 shows a state machine of the QDRII memory controller for burst lengths of four. When calibration is complete, the state machine is in the IDLE state. When the User Write Data FIFO and the User Write Address FIFO are not empty (that is, the user has written into the FIFOs), the state machine goes to the WRITE state, initiating a memory write of one burst.

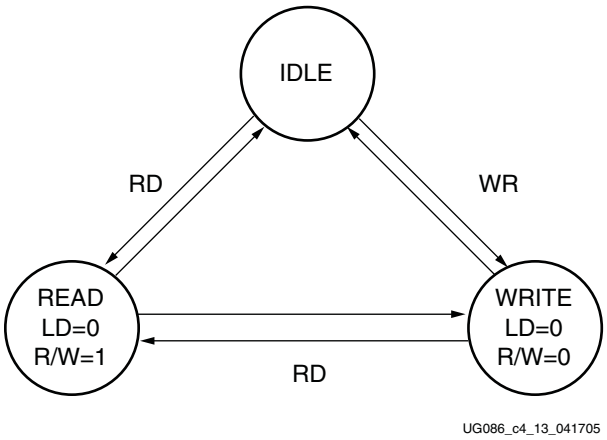


Figure 4-6: **QDRII Memory Controller State Machine with Burst Lengths of 4**

When the User Read Address FIFO is not empty (that is, the user has written into the User Read Address FIFO), the state machine goes to the READ state, initiating a memory read of one burst.

From the IDLE state, the QDRII memory controller can go to either the WRITE or the READ state depending on which FIFO is not empty. Writes are given priority. In the WRITE state, a memory write is initiated, and the User Read Address Not Empty status is checked to transfer into the READ state. When the User Read Address FIFO is empty, the state machine goes to the IDLE state.

In the READ state, a memory read is initiated, and the User Write Data and the User Write Address FIFO Not Empty status is checked before going to the WRITE state. If the FIFOs are empty, the state machine goes to the IDLE state.

Figure 4-7 shows a state machine of the QDR II memory controller for burst lengths of two. When calibration is complete, the state machine is in the IDLE state. When the User Write Data FIFO or Write Address FIFO is not empty (that is, the user has written into either FIFO), the state machine goes to the READ_WRITE state, initiating a memory write of one burst, or when the User Read Address FIFO is not empty (that is, the user has written into the User Read Address FIFO), the state machine goes to the READ_WRITE state, initiating a memory read of one burst.

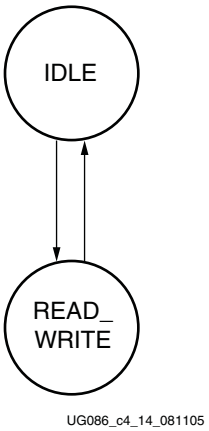


Figure 4-7: **QDRII Memory Controller State Machine with Burst Lengths of 2**

From the IDLE state, the QDR II memory controller goes to READ_WRITE state if either Read FIFO Address FIFO or the Read Write FIFO is not empty. In the READ_WRITE state, the User Read Address Not Empty status is checked to initiate a memory read. To initiate a memory write in the READ_WRITE state, the User Write Data/Write Address FIFOs Not Empty status is checked. If the Write FIFOs are empty and the Read Address FIFO is also empty, the state machine goes to the IDLE state. If either Write FIFO or the Read Address FIFO is NOT empty, the state machine remains in the READ_WRITE state to issue memory writes or reads.

Refer to XAPP703 [Ref 12] for detailed design and timing analysis of the QDRII memory controller module.

Datapath

The Datapath module transmits and receives data to and from the memories. Its major functions are listed below:

- Asserts a write-enable signal for memories with burst lengths of two or four
- Asserts a read-enable signal to memory and a write-enable signal to the User Read Data FIFO
- Generates increment/decrement signals (tap count) for IDELAY elements in the IOBS
- Center-aligns the data window to the FPGA clock

Refer to XAPP701 [Ref 10] for techniques on data writes to memory and data captures from memory. For burst lengths of two, the write-enable signal to memory is asserted at the same time that write data is driven. For burst lengths of four, the write-enable signal is asserted one clock before the write data is driven on the memory bus. The data is driven on both edges of the clock. The address to memory is driven for one full clock cycle.

Memory read data is edge-aligned with the source-synchronous clock, CQ. The QDRII memory clock, to which data is synchronized, is a free-running strobe. Data is captured

using this free-running strobe and is transferred to the FPGA domain. The input side of the data uses exactly the same resources as used by the input side of the strobe, which ensures matched delays on the data and the strobe signals until the strobe is delayed in the strobe delay circuit.

The RD_STB_n_out signal is generated during reads by the controller. This signal is brought out on the board to memory and is looped back to RD_STB_n_in. The RD_STB_n_in signal, which is delayed the same as the data, is used to delay the write enable for the Read Data FIFOs, where the captured data is stored. An RD_STB_n signal is generated for every memory component.

The delay calibration circuit generates the delay reset, delay select, and delay increment values for IDELAY elements used in delaying strobes and data read from memory. The strobe is center-aligned with the FPGA clock, which results in the data window falling to the center of the FPGA clock. Refer to XAPP703 [Ref 12] for details about the delay calibration. The delay calibration logic is the same for the QDRII SRAM controller and the DDRII SRAM controller.

Infrastructure

The Infrastructure (Infrastructure_top) module generates the FPGA clocks and the reset signals. A DCM generates the clock, 270-degree phase-shifted version of the clock, and the divide-by-four clock.

IOBS

All the input and output signals of the QDRII SRAM controller are implemented in the IOBS module. All address and byte enable signals are registered in the IOBs and driven out.

The IDELAY elements for the read strobe, the echo clock, and data read from memory are implemented in the IOBS. The IOBS also implements Inout buffers for write and read data. It registers the output data (ODDR) before driving it out and registers the input data (IDDR).

User Interface

The user interface consists of six FIFOs. The User Write interface has three FIFOs: one FIFO is used for the memory address, and the remaining two FIFOs contain positive-edge and negative-edge data for memory. The QDRII memory controller checks the not empty status of these FIFOs and initiates a memory write. The user interface is single data rate (SDR). The controller handles the conversion from the SDR user interface to the DDR Memory interface and vice versa.

The User Read interface has three FIFOs: one FIFO is used for the memory address, and the remaining two FIFOs contain positive-edge and negative-edge data read from memory. The user writes to the User Read Address FIFO the memory address from which data is to be read. The QDRII memory controller checks the not empty status of this FIFO and initiates a memory read burst. The data read is stored in the User Read Data FIFOs. The user reads these FIFOs to access the data read from memory.

Refer to the timing diagrams in "QDRII Controller User Interface Signals" for how the user can access these FIFOs.

QDRII Controller User Interface Signals

Table 4-1 describes the QDRII Controller user interface signals.

Table 4-1: **QDRII User Interface Signals**

Signal Name	Direction	Description
REFCLK_P and REFCLK_N	Input	Differential DCM input clock. This is the reference clock to the DCM input. Various clocks for the QDRII memory controller are derived from this clock. The clock for the QDRII memory controller is the output of the DCM. DCM outputs CLK270 and CLKDV (set to divide by 4) are also inputs to the QDRII memory controller. The design operates with respect to this clock frequency.
SYS_RST_N	Input	Reset signal for the QDRII memory controller.
DLY_CLK_200_P and DLY_CLK_200_N	Input	Differential 200 MHz clock. 200 MHz reference clock for IDELAY_CTRL.
USER_CLK	Output	All user interface signals are to be synchronized to this clock.
USER_RST	Output	This reset is active until the DCM is not locked.
USER_W_n	Input	This active-Low signal is the write enable for the User Write Data and User Write Address FIFO. The user asserts this signal to write new data to the FIFOs. The QDRII memory controller reads the data from the User Write Data FIFO and writes to memory at the address located in the User Write Address FIFO.
USER_AD_RD[(addr_width-1):0]	Input	QDRII memory address for read data. This is valid when USER_R_n is asserted.
USER_AD_WR[(addr_width-1):0]	Input	QDRII memory address for write data. This is valid when USER_W_n is asserted.
USER_BW_n[(BW_width-1):0]	Input	Byte enables for QDRII memory write data. The byte enables are valid when USER_W_n is asserted. Byte enables are stored with the write address.
USER_DWL[(data_width-1):0]	Input	Positive-edge data for memory writes. The data bus is valid when USER_W_ n is asserted.
USER_DWH[(data_width-1):0]	Input	Negative-edge data for memory writes. The data bus is valid when USER_W_ n is asserted.
USER_R_n	Input	This active-Low signal is the write enable for the User Read Address FIFO. The user asserts this signal to read new data from memory. The QDRII memory controller reads the address from the Read Address FIFO and does a memory read to the corresponding memory address.
USER_QEN_n	Input	This active-Low signal is the read enable for the User Read Data FIFOs. The QDRII memory controller captures the data read from memory and stores it in the Read Data FIFOs. The user can access these FIFOs to get the data read from memory.

Table 4-1: **QDRII User Interface Signals *(Continued)***

Signal Name	Direction	Description
USER_WR_FULL	Output	This signal indicates the User Write FIFO status. It is asserted when either the User Write Address FIFO or the User Write Data FIFO is full. When this signal is asserted, any writes to the User Write Address FIFO and the User Write Data FIFO are invalid, possibly leading to controller malfunction.
USER_RD_FULL	Output	This signal indicates the User Read Address FIFO status. It is asserted when the User Read Address FIFO is full. When this signal is asserted, any write to the User Read Address FIFO is ignored.
USER_QRL[(data_width-1):0]	Output	Positive-edge data read from memory. This data is output when USER_QEN_n is asserted.
USER_QRH[(data_width-1):0]	Output	Negative-edge data read from memory. This data is output when USER_QEN_n is asserted.
USER_QR_EMPTY	Output	This signal indicates the User Read Data FIFO status. This signal is asserted when the User Read Data FIFO is empty. When this signal is asserted, any read to the User Read Data FIFO is not valid.
USER_WR_ERR	Output	This signal is asserted when an error occurs while writing to the User Write Data FIFO or the User Write Address FIFO.
USER_RD_ERR	Output	This signal is asserted when an error occurs while writing to the User Read Address FIFO.
USER_QR_ERR	Output	This signal is asserted when an error occurs while reading the User Read Data FIFO.
DLY_CAL_DONE	Output	This signal is asserted to indicate that the calibration is done.
TAP_CNT[5:0]	Output	This signal outputs the IDELAY tap value, which can be used for monitoring purposes.

Notes:

1. All signal directions are with respect to the QDRII memory controller.
2. The MIG 1.5 tool prepends *cntrl0_* to all user interface signal names (except system signals REFCLK_P, REFCLK_N, SYS_RST_N, DLY_CLK_200_P, and DLY_CLK_200_N). This naming convention supports multicontrollers in future releases.

Figure 4-8 shows the user interface when the burst length is four for the selected memory. The DLY_CAL_DONE signal is asserted after calibration completes. The controller initiates a memory write or a memory read only after calibration is complete. It is recommended that no writes occur to the User FIFOs until the DLY_CAL_DONE signal is asserted.

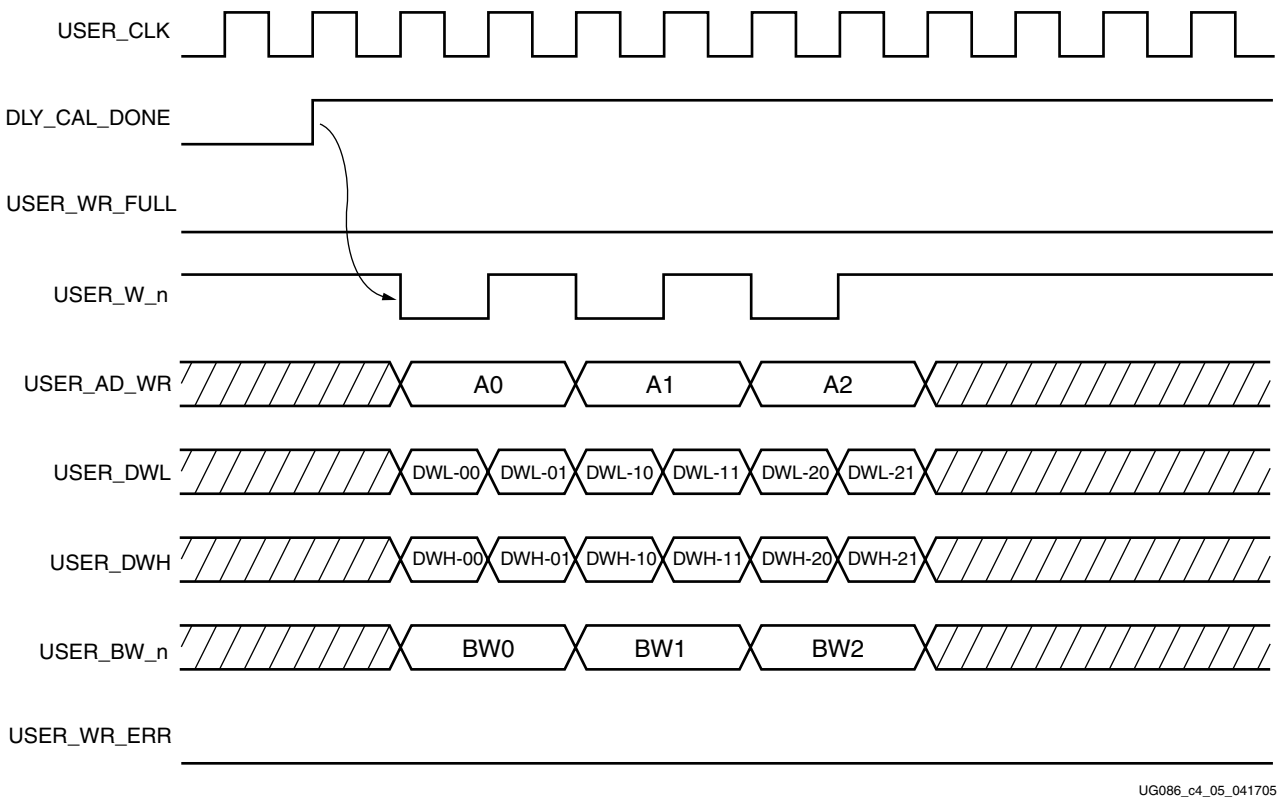


Figure 4-8: **Write to User Write Data and User Write Address FIFOs (BL = 4)**

To initiate writes to memory, the user writes to the User Write Data FIFO and the User Write Address FIFO. To write into these FIFOs, the user asserts the USER_W_n signal Low, which is the write enable for those FIFOs. When the burst length is four (as shown in Figure 4-8), one memory address and two data words are to be written into the User Write FIFOs with every write enable. Data is to be driven on both the LSB bus (USER_DWL) and the MSB bus (USER_DWH). The QDRII memory controller drives the LSB bus data to memory on the positive clock edge and drives the MSB bus data on the negative clock edge. The controller handles the conversion from the SDR user interface to the DDR Memory interface and vice versa.

When the burst length is two, one memory address and one data word are to be written into the User Write FIFOs with every write enable. Figure 4-9 shows the user interface to the User Write Data FIFOs and User Write Address FIFO when the memory burst length is two.

USER_WR_FULL is asserted when either the User Write Data FIFOs or the User Write Address FIFO is full. This assertion is an indicator to the User to stop writing into these FIFOs. USER_WR_ERR is asserted to indicate an error occurred when writing into any of these FIFOs.

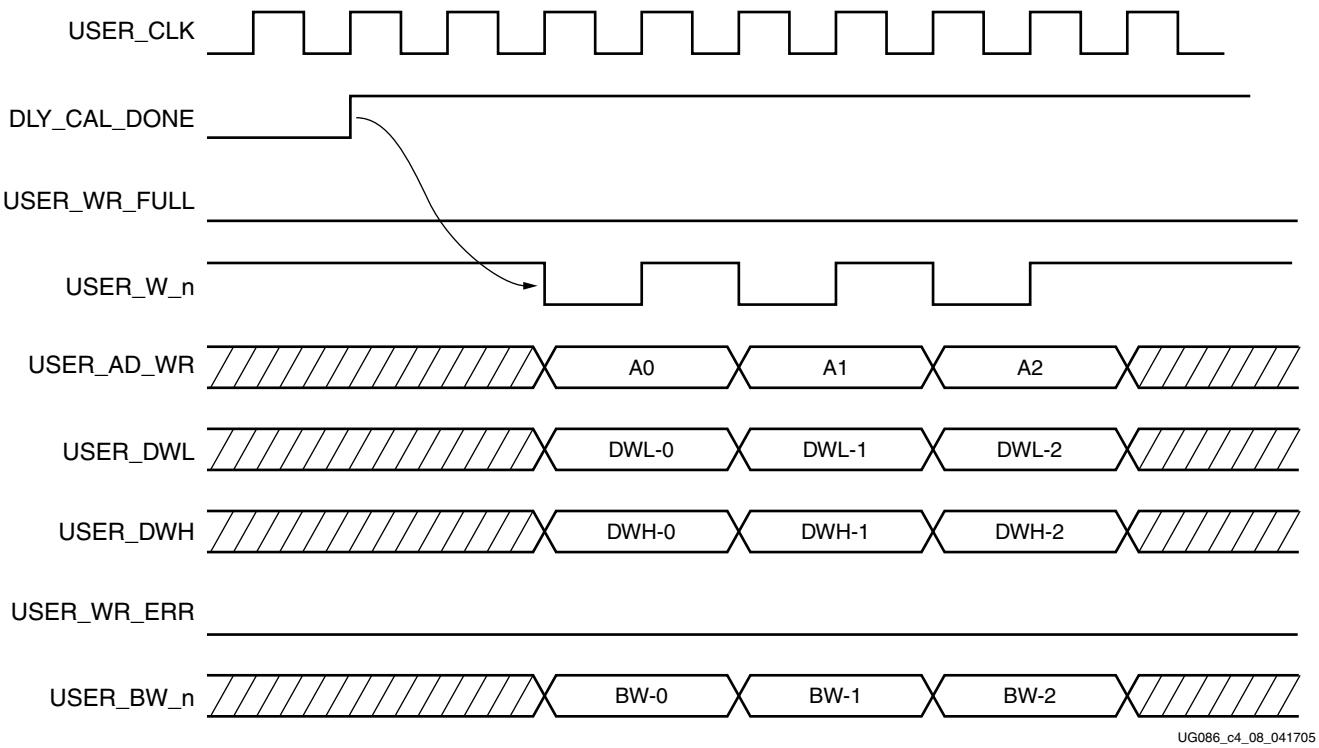


Figure 4-9: **Write to User Write Data and User Write Address FIFOs (BL = 2)**

Figure 4-10 shows the user interface to the User Read Address FIFO when the memory burst length is two or four. The user interface to the User Read Address FIFO is the same for burst lengths of two and four. This interface is similar to the User Write Address FIFO interface. To write into the User Read Address FIFO, the user asserts the USER_R_n signal Low, which is the write enable for this FIFO. The QDRII memory controller reads the memory address from the FIFO and places it on the memory address bus.

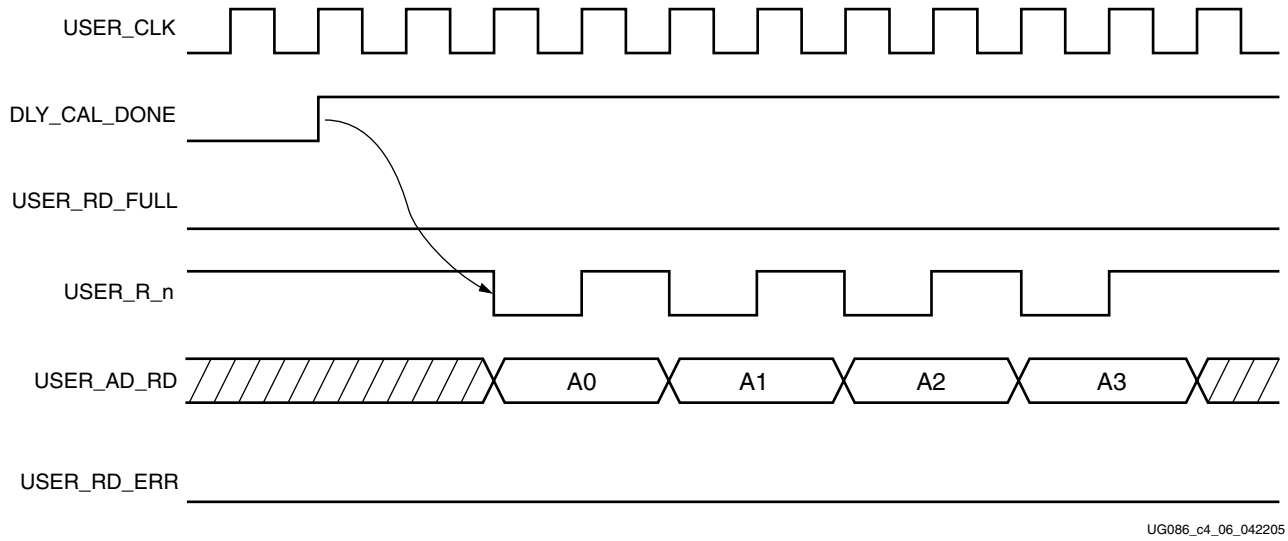


Figure 4-10: **Write to User Read Address FIFO (BL = 2 or BL = 4)**

USER_RD_FULL is asserted when the User Read Address FIFO is full. This assertion is an indicator to the user to stop writing into the FIFO. The USER_RD_ERR signal is asserted to indicate an error occurred when writing to this FIFO.

Figure 4-11 shows the user interface to the Read Data FIFOs when the memory burst length is two or four. The user interface to the User Read Data FIFOs is the same for burst lengths of two and four. The user can read from the FIFOs only after the USER_QR_EMPTY signal is deasserted. The QDRII memory controller reads data from the memory and stores it in the Read Data FIFOs. To read these FIFOs, the user asserts USER_QEN_n Low, which is the read-enable signal to the Read Data FIFOs. When USER_QEN_n is asserted, a data word is driven on the LSB bus (USER_QRL) and on the MSB bus (USER_QRH).

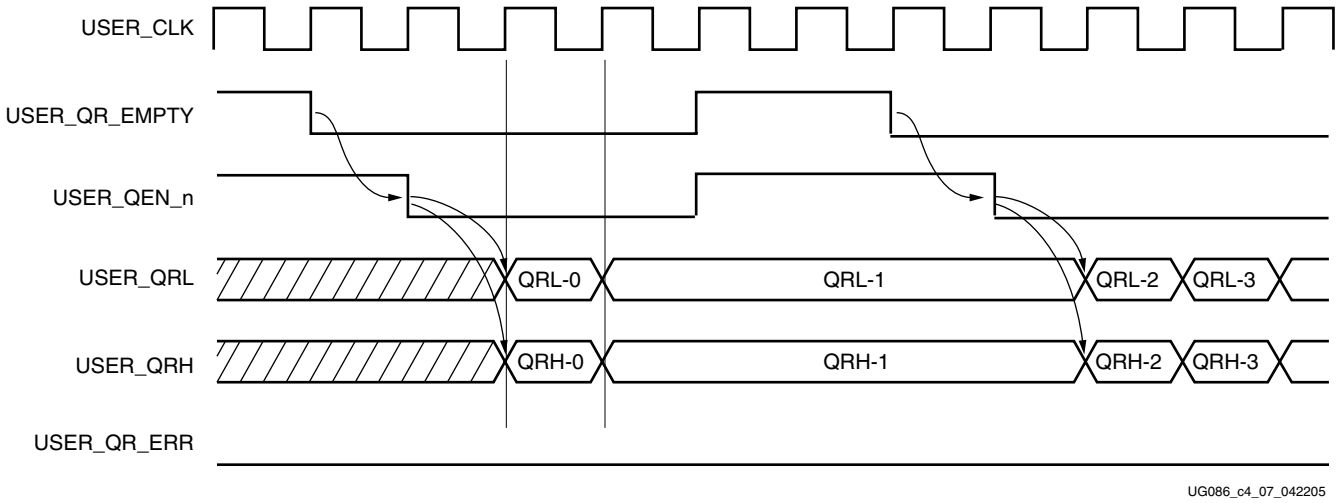


Figure 4-11: **Read from User Read Data FIFOs (BL = 2 or BL = 4)**

USER_QR_EMPTY is asserted when either User Read Data FIFO is empty. This assertion is an indicator to the user to stop writing into the FIFOs. USER_QR_ERR is asserted to indicate an error occurred when reading these FIFOs.

Chapter 5

Implementing DDRII SRAM Controllers

This chapter describes how to implement DDRII SRAM interfaces for Virtex-4 FPGAs created with the MIG 1.5 design tool.

Feature Summary

This section summarizes the supported and unsupported features of the DDRII SRAM controller design.

Supported Features

The DDRII SRAM controller design supports:

- A maximum frequency of 250 MHz
- Data widths of 9, 18, 36, and 72 bits
- Burst lengths of two and four
- Implementation using different Virtex-4 devices
- Operation with any 9-bit, 18-bit, and 36-bit memory component
- Verilog and VHDL

Unsupported Features

The DDRII SRAM controller design does not support:

- DDR SIO memory

Architecture

Figure 5-1 shows the top-level block diagram of the DDRII SRAM controller interface. One side of the DDRII SRAM controller connects to the user interface denoted as *Block Application*. The other side of the controller interfaces to DDRII memory. The memory interface data width is selectable.

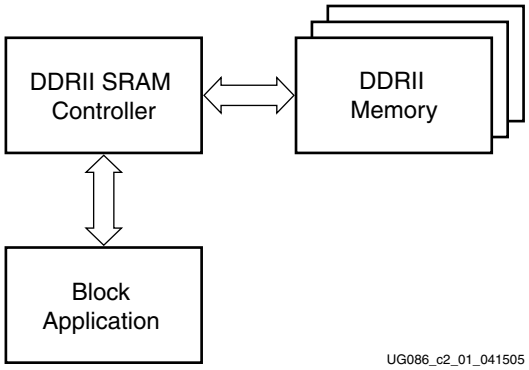


Figure 5-1: **DDRII SRAM Controller Interface**

Data is double-pumped to DDRII memory on both the positive and the negative edges of the clock. The HSTL_18 Class I I/O standard is used for the data, address, and control signals.

DDRII memory interfaces are source-synchronous and double data rate like DDR SDRAM interfaces.

Interface Model

The Memory interface is layered to simplify the design and make the design modular. Figure 5-2 shows the layered memory interface used in the DDRII SRAM controller. The three layers are the application layer, the implementation layer, and the physical layer.

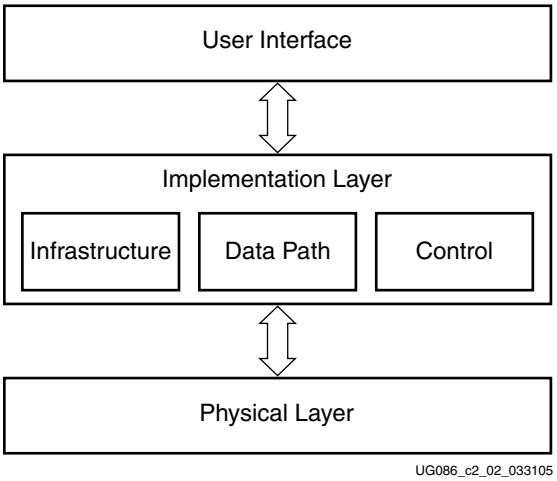


Figure 5-2: **Interface Layering Model**

The application layer creates the user interface, which initiates memory writes and reads by writing data and memory addresses to the User Interface FIFOs. The DDRII SRAM controller monitors the User Interface FIFOs and initiates data transfers with DDRII memory accordingly.

Hierarchy

Figure 5-3 shows the DDRII SRAM controller hierarchy.

Figure 5-3: **DDRII SRAM Controller Hierarchy**

DDRII SRAM Controller Modules

Figure 5-4 shows a detailed block diagram of the DDRII SRAM controller. The four blocks shown are sub-blocks of the top module. The functionalities of these blocks are explained in the subsections following the figure.

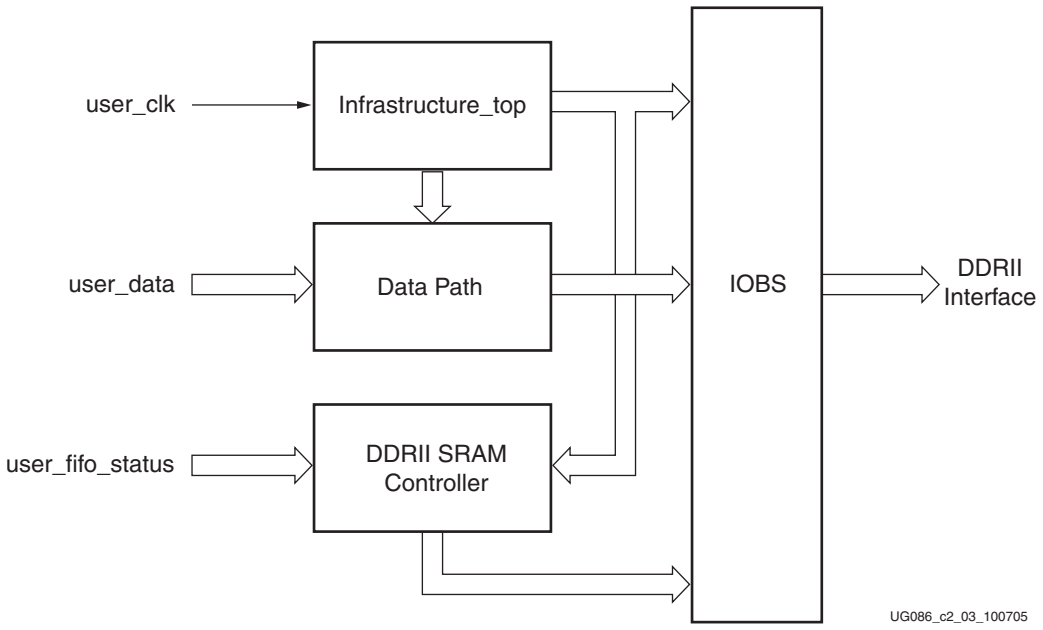


Figure 5-4: **DDRII SRAM Controller Modules**

Figure 5-5 shows the DDRII SRAM controller modules with a 36-bit interface.

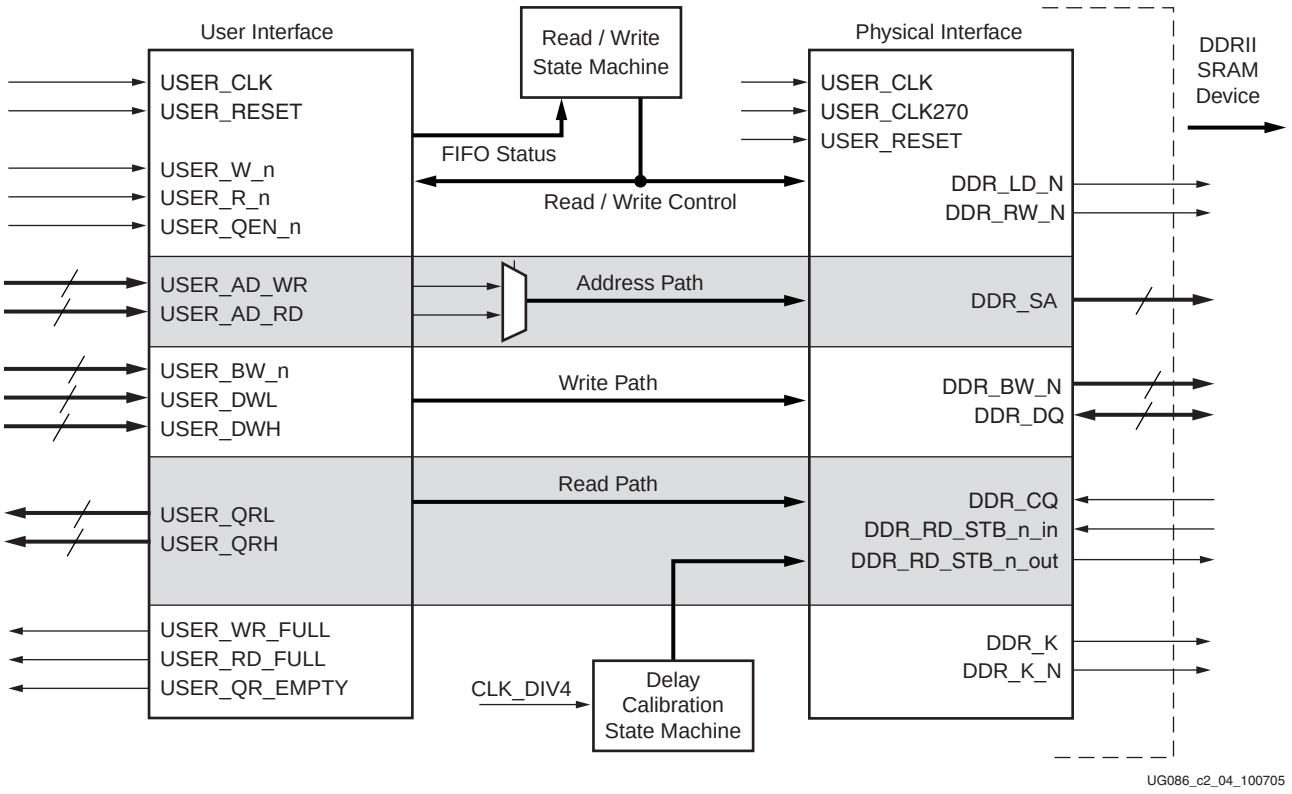


Figure 5-5: **DDRII SRAM Controller Modules with Interface Signals**

Controller

The DDRII SRAM controller initiates memory writes when the User Write Data FIFO and the User Write Address FIFO are not empty. The user writes the data and the address into the User Write Data FIFO and the User Write Address FIFO, respectively. This data is written into memory.

The user initiates memory read commands when the User Read Data FIFO are not full and the User Read Address FIFO is not empty. The memory address from which data is to be read is written into the User Read Address FIFO. The User Read Data FIFOs store the data captured from memory. The user can access the data read from memory by reading the User Read Data FIFOs.

When the User Write Data FIFO is not empty, the DDRII SRAM controller generates a write-enable signal for memory. When the write-enable signal is asserted, data is transferred to memory. The DDRII SRAM controller also generates a read-enable signal for memory when the User Read Address FIFO is not empty and the User Read Data FIFOs are not full. When the read-enable signal is asserted, data read from memory is captured and stored in the User Read Data FIFOs.

Figure 5-6 shows a state machine of the DDRII SRAM controller with memory burst length set to two. When calibration is complete, the state machine is in the IDLE state. When the User Write Data FIFO and the User Write Address FIFO are not empty (that is, the user has written into the FIFOs), the state machine goes to the WRITE state, initiating a memory write of one burst.

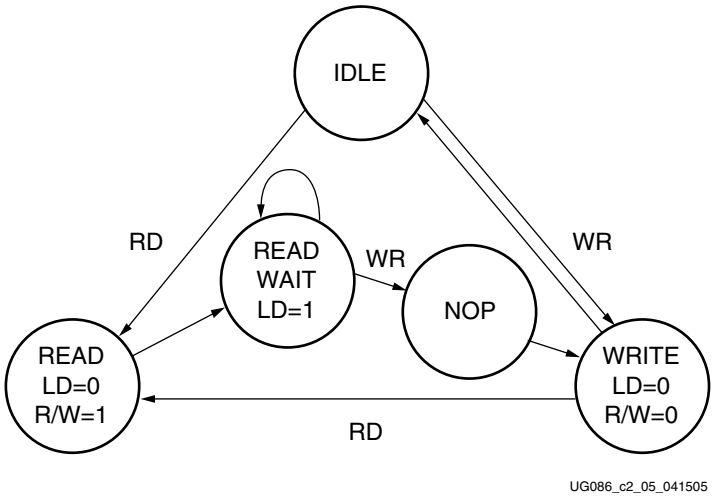


Figure 5-6: **DDRII SRAM Controller State Machine (BL = 2)**

When the User Read Address FIFO is not empty (that is, the user has written into the User Read Address FIFO), the state machine goes to the READ state, initiating a memory read of one burst.

From the IDLE state, the DDRII SRAM controller can go to either the WRITE or the READ state depending on which FIFO is not empty. Writes are given priority. In the WRITE state, a memory write is initiated and the User Read Address Not Empty status is checked to transfer into the READ state. When the User Read Address FIFO is empty, the state machine goes to the IDLE state.

In the READ state, a memory read is initiated. From the READ to the WRITE states, two wait states are present to allow for memory bus turnaround time. At high frequencies, two NOP states instead of one might be required between memory reads and writes.

In the READ WAIT state, the User Write Data and User Write Address FIFO Not Empty status is checked before going to the WRITE state. If the FIFOs are empty, the state machine waits in the same state for the user to write into these FIFOs.

Figure 5-7 shows a state machine of the DDRII SRAM controller with memory burst length set to four. This state machine is similar to that with memory burst length set to two with a few modifications. Because the burst length is four, to allow four data words to be written into memory, a wait state is added between the WRITE and READ states. In addition, three wait states are added between the READ and the WRITE states.

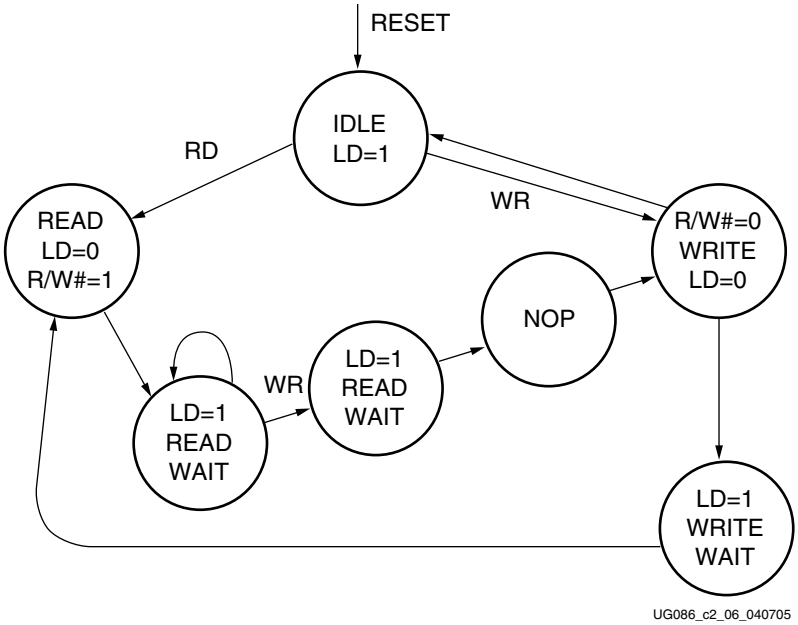


Figure 5-7: **DDRII SRAM Controller State Machine (BL = 4)**

Datapath

The Datapath module transmits and receives data to and from the memories. Its major functions are listed below:

- Asserts a write-enable signal for memories with burst lengths of two or four
- Asserts a read-enable signal to memory and a write-enable signal to the User Read Data FIFO
- Generates increment/decrement signals (tap count) for IDELAY elements in the IOBS
- Center-aligns the data window to the FPGA clock

Refer to XAPP701 [Ref 10] for techniques on data writes to memory and data captures from memory. For burst lengths of four and two, the write-enable signal is asserted one clock before the write data is driven on the memory bus. The data is driven on both edges of the clock. The address to memory is driven for one full clock cycle.

Memory read data is edge-aligned with the source-synchronous clock, CQ. The DDRII clock, to which data is synchronized, is a free-running strobe. Data is captured using this free-running strobe and is transferred to the FPGA domain. The input side of the data uses exactly the same resources as used by the input side of the strobe, which ensures matched delays on the data and the strobe signals until the strobe is delayed in the strobe delay circuit.

The RD_STB_n_out signal is generated during reads by the controller. This signal is brought out on the board to memory and is looped back to RD_STB_n_in. The RD_STB_n_in signal, which is delayed the same as the data, is used to delay the write enable for the Read Data FIFOs, where the data captured is stored. An RD_STB_n signal is generated for every memory component.

The delay calibration circuit generates the delay reset, delay select, and delay increment values for IDELAY elements used in delaying strobes and data read from memory. The strobe is center-aligned with the FPGA clock, which results in the data window falling to the center of the FPGA clock. Refer to XAPP703 [Ref 12] for details about the delay calibration. The delay calibration logic is the same for the DDRII SRAM controller and the QDRII SRAM controller.

Infrastructure

The Infrastructure module (Infrastructure_top) generates the FPGA clocks and the reset signals. A DCM generates the clock, the 270-degree phase-shifted version of the clock, and the divide-by-four clock.

IOBS

All the input and output signals of the DDRII SRAM controller are implemented in the IOBS module. All address and byte enable signals are registered in the IOBs and driven out.

The IDELAY elements for the read strobe, the echo clock, and data read from memory are implemented in the IOBS. The IOBS also implements Inout buffers for write and read data. The IOBS registers the output data (ODDR) before driving it out and also registers the input data (IDDR).

User Interface

The user interface consists of six FIFOs. The User Write interface has three FIFOs: one FIFO is used for the memory address, and the remaining two FIFOs contain positive-edge and negative-edge data for memory. The DDRII SRAM controller checks the not empty status of these FIFOs and initiates a memory write. The user interface is single data rate (SDR). The controller handles the conversion from the SDR user interface to the DDR Memory interface and vice versa.

The User Read interface has three FIFOs, where one FIFO is used for the memory address and the remaining two FIFOs contain positive-edge and negative-edge data read from memory. The user writes to the User Read Address FIFO the memory address from which data is to be read. The DDRII SRAM controller checks the status of this FIFO and initiates a memory read burst. The data read is stored in the User Read Data FIFOs. The user reads these FIFOs to access the data read from memory.

Refer to the timing diagrams in "DDRII SRAM Controller Interface Signals" for how the user can access these FIFOs.

DDRII SRAM Controller Interface Signals

Table 5-1 describes the DDRII SRAM Controller interface signals.

Table 5-1: **DDRII SRAM Controller Interface Descriptions**

Signal Name	Direction	Description
REFCLK_P and REFCLK_N	Input	Differential DCM input clock. This is the reference clock to the DCM input. Various clocks for the DDRII SRAM controller are derived from this clock. The clock for the DDRII SRAM controller is the output of the DCM. DCM outputs CLK270 and CLKDV (set to divide by 4) are also inputs to the DDRII SRAM controller. The design operates with respect to this clock frequency.
SYS_RST_N	Input	Reset signal for the DDRII SRAM controller.
DLY_CLK_200_P and DLY_CLK_200_N	Input	Differential 200 MHz clock. 200 MHz reference clock for IDELAY_CTRL.
USER_CLK	Output	All user interface signals are to be synchronized to this clock.
USER_RST	Output	This reset is active until the DCM is not locked.
USER_W_n	Input	This active-Low signal is the write enable for the User Write Data and User Write Address FIFO. The user asserts this signal to write new data to the FIFOs. The DDRII SRAM controller reads the data from the User Write Data FIFO and writes to memory at the address located in the User Write Address FIFO.
USER_AD_RD[(addr_width-1):0]	Input	DDRII memory address for read data. This bus is valid when USER_R_n is asserted.
USER_AD_WR[(addr_width-1):0]	Input	DDRII memory address for write data. This bus is valid when USER_W_n is asserted.
USER_BW_n[(BW_width-1):0]	Input	Byte enables for DDR memory write data. The byte enables are valid when USER_W_n is asserted. Byte enables are stored with the memory write address.
USER_DWL[(data_width-1):0]	Input	Positive-edge data for memory writes. The data bus is valid when USER_W_ n is asserted.
USER_DWH[(data_width-1):0]	Input	Negative-edge data for memory writes. The data bus is valid when USER_W_ n is asserted.
USER_R_n	Input	This active-Low signal is the write enable for the User Read Address FIFO. The user asserts this signal to read new data from memory. The DDRII SRAM controller reads the address from the User Read Address FIFO and does a memory read to the corresponding memory address.
USER_QEN_n	Input	This active-Low signal is the read enable for the User Read Data FIFOs. The DDRII SRAM controller captures the data read from memory and stores it in the User Read Data FIFOs. The user can access these FIFOs to get the data read from memory.

Table 5-1: **DDRII SRAM Controller Interface Descriptions *(Continued)***

Signal Name	Direction	Description
USER_WR_FULL	Output	This signal indicates the User Write FIFO status. It is asserted when either the User Write Address FIFO or the User Write Data FIFO is full. When this signal is asserted, any writes to the User Write Address FIFO and the User Write Data FIFO are invalid, possibly leading to controller malfunction.
USER_RD_FULL	Output	This signal indicates the User Read Address FIFO status. It is asserted when the User Read Address FIFO is full. When this signal is asserted, any write to the User Read Address FIFO is ignored.
USER_QRL[(data_width-1):0]	Output	Positive-edge data read from memory. This data is output when USER_QEN_n is asserted.
USER_QRH[(data_width-1):0]	Output	Negative-edge data read from memory. This data is output when USER_QEN_n is asserted.
USER_QR_EMPTY	Output	This signal indicates the User Read Data FIFO status. This signal is asserted when the User Read Data FIFO is empty. When this signal is asserted, any read to the User Read Data FIFO is not valid.
USER_WR_ERR	Output	This signal is asserted when an error occurs while writing to the User Write Data FIFO or the User Write Address FIFO.
USER_RD_ERR	Output	This signal is asserted when an error occurs while writing to the User Read Address FIFO.
USER_QR_ERR	Output	This signal is asserted when an error occurs while reading the User Read Data FIFO.
DLY_CAL_DONE	Output	This signal is asserted to indicate that the calibration is done.
TAP_CNT[5:0]	Output	This signal outputs the IDELAY tap value, which can be used for monitoring purposes.

Notes:

1. All the signal directions are with respect to the DDRII SRAM controller.
2. The MIG 1.5 tool prepends *cntrl0_* to all user interface signal names (except system signals REFCLK_P, REFCLK_N, SYS_RST_N, DLY_CLK_200_P, and DLY_CLK_200_N). This naming convention supports multicontrollers in future releases.

Figure 5-8 shows the user interface when the burst length is four for the selected memory. The DLY_CAL_DONE signal is asserted after calibration completes. The controller initiates a memory write or a memory read only after calibration is complete. It is recommended that no writes occur to the User FIFOs until the DLY_CAL_DONE signal is asserted.

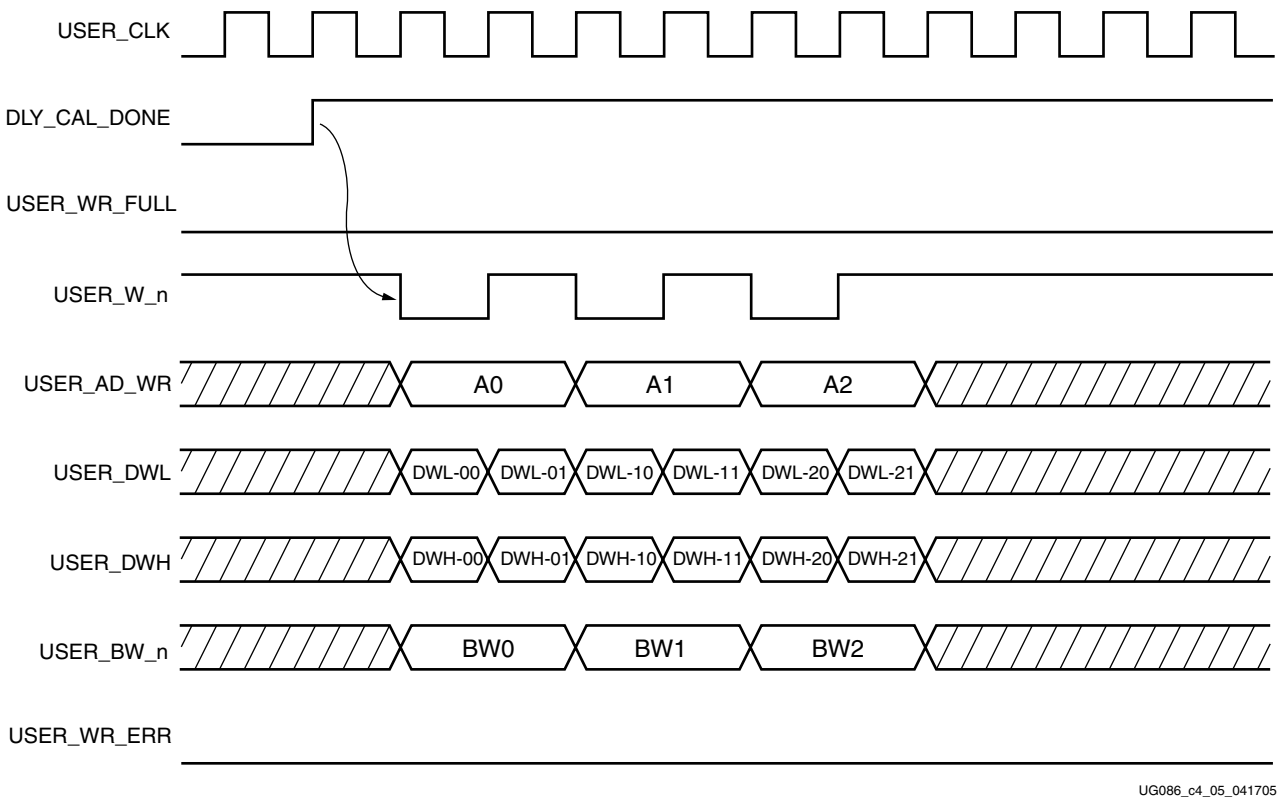


Figure 5-8: **Write to User Write Data and User Write Address FIFOs (BL = 4)**

To initiate writes to memory, the user writes to the User Write Data FIFO and the User Write Address FIFO. To write into these FIFOs, the user asserts the USER_W_n signal Low, which is the write enable for those FIFOs. When the burst length is four (as shown in Figure 5-8), one memory address and two data words are to be written into the User Write FIFOs with every write enable. Data is to be driven on both the LSB bus (USER_DWL) and the MSB bus (USER_DWH). The DDRII SRAM controller drives the LSB bus data to memory on the positive clock edge and drives the MSB bus data on the negative clock edge. The controller handles the conversion from the SDR user interface to the DDR Memory interface and vice versa.

When the burst length is two, one memory address and one data word are to be written into the User Write FIFOs with every write enable. Figure 5-9 shows the user interface to the User Write Data FIFOs and User Write Address FIFO when the memory burst length is two.

USER_WR_FULL is asserted when either the User Write Data FIFOs or the User Write Address FIFO is full. This assertion is an indicator to the user to stop writing into these FIFOs. USER_WR_ERR is asserted to indicate an error occurred when writing into any of these FIFOs.

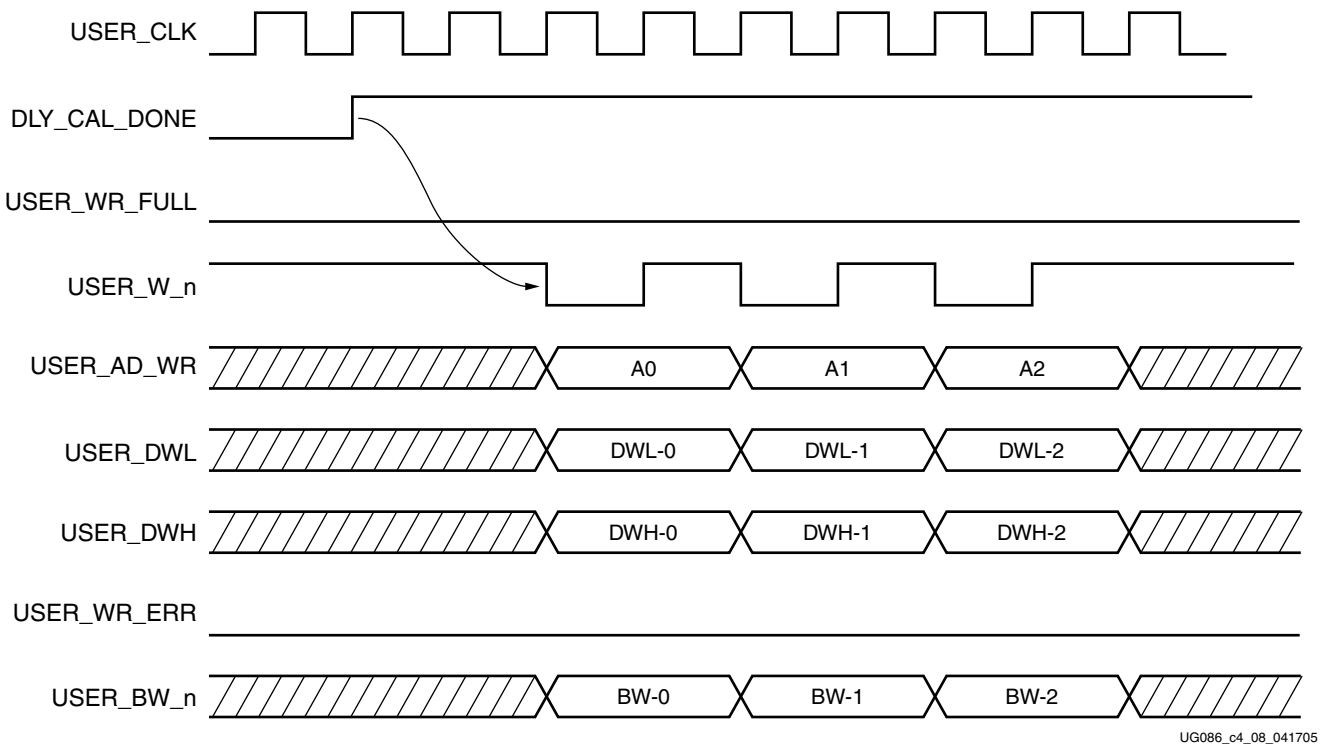


Figure 5-9: **Write to User Write Data and User Write Address FIFOs (BL = 2)**

Figure 5-10 shows the user interface to the User Read Address FIFO when the memory burst length is two or four. The user interface to the User Read Address FIFO is the same for burst lengths of two and four. This interface is similar to the User Write Address FIFO interface. To write into the User Read Address FIFO, the user asserts the USER_R_n signal Low, which is the write enable for this FIFO. The DDRII SRAM controller reads the memory address from the FIFO and places it on the memory address bus.

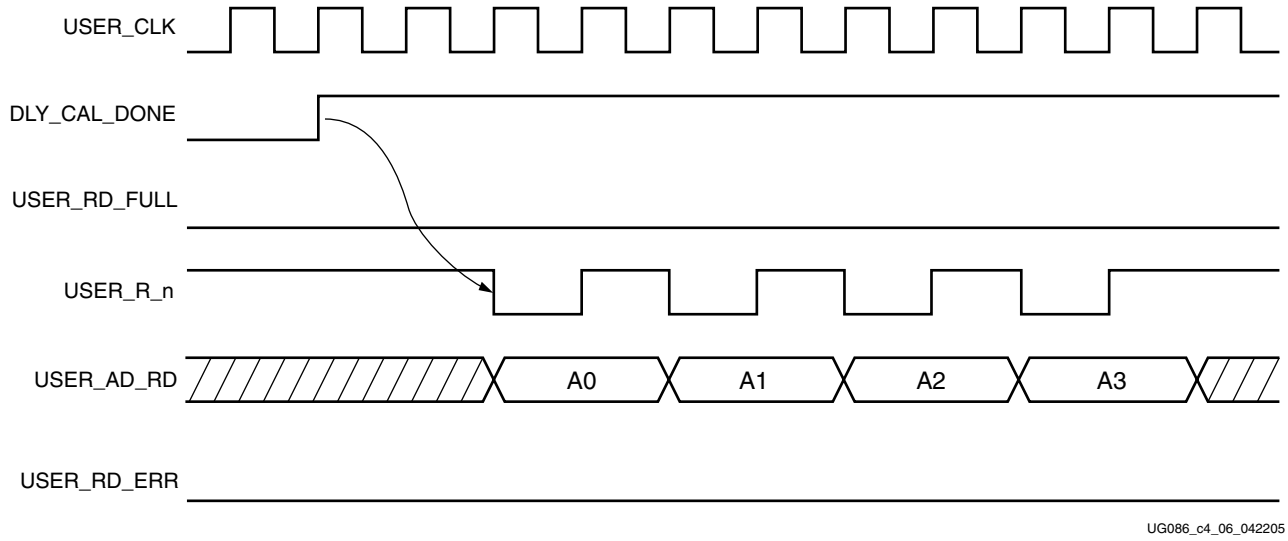


Figure 5-10: **Write to User Read Address FIFO (BL = 2 or BL = 4)**

USER_RD_FULL is asserted when the User Read Address FIFO is full. This assertion is an indicator to the user to stop writing into the FIFO. The USER_RD_ERR signal is asserted to indicate an error occurred when writing to this FIFO.

Figure 5-11 shows the user interface to the Read Data FIFOs when the memory burst length is two or four. The user interface to the User Read Data FIFOs is the same for burst lengths of two and four. The user can read from the FIFOs only after the USER_QR_EMPTY signal is deasserted. The DDRII SRAM controller reads data from the memory and stores it in the User Read Data FIFOs. To read these FIFOs, the user asserts USER_QEN_n Low, which is the read-enable signal to the User Read Data FIFOs. When USER_QEN_n is asserted, a data word is driven on the LSB bus (USER_QRL) and on the MSB bus (USER_QRH).

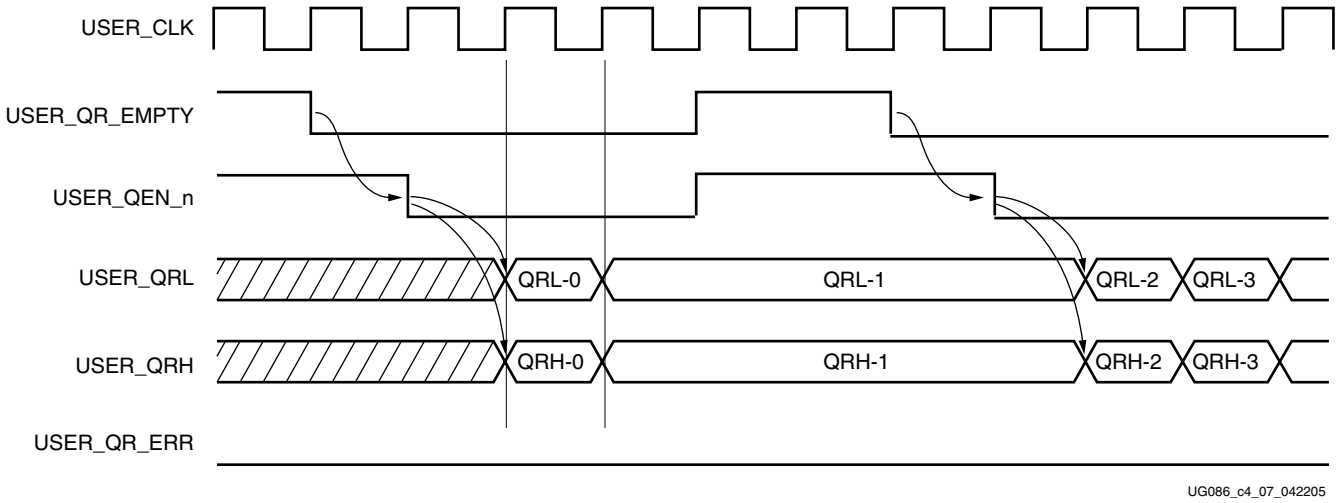


Figure 5-11: **Read from User Read Data FIFOs (BL = 2 or BL = 4)**

USER_QR_EMPTY is asserted when either User Read Data FIFO is empty. This assertion is an indicator to the user to stop writing into the FIFOs. USER_QR_ERR is asserted to indicate an error occurred when reading these FIFOs.

Chapter 6

Implementing RLDRAM II Controllers

Reduced Latency DRAM (RLDRAM II) devices address high bandwidth memory requirements. The RLDRAM II utilizes an eight-bank architecture optimized for high-speed operation and a double data rate I/O for increased bandwidth. This chapter describes how to implement RLDRAM II interfaces for Virtex-4 FPGAs created with the MIG 1.5 design tool. This design is based on XAPP710 [Ref 14].

Feature Summary

This section summarizes the supported and unsupported features of the RLDRAM II controller design.

Supported Features

The RLDRAM II memory controller design supports:

- A maximum frequency of 270 MHz
- Both SIO and CIO memories
- Multiplexed and non-multiplexed addresses
- All configurations (Config1, Config2, and Config3)
- x9, x18, and x36 components
- Data widths of 9, 18, 36, and 72 bits
- Back-to-back read and write operations
- Write followed by read operations
- Read followed by write operations
- All combinations of the Mode Register
- XST and Synplicity synthesis tools
- Verilog and VHDL

Unsupported Features

The RLDRAM II memory controller design does not support:

- Commands in successive clocks. The controller processes these commands with one extra clock latency.

Supported RLDRAM II Devices

The RLDRAM II memory controller design supports all RLDRAM II devices from Micron as indicated in Table 6-1.

Table 6-1: **Supported RLDRAM II Devices**

Device	Make	CIO/SIO	Configuration	Speed Grade	Supported Data Widths (in bits)
MT49H32M9FM	Micron	CIO	x9	(-5), (-25), (-33)	9, 18, 36, 72
MT49H16M18FM	Micron	CIO	x18	(-5), (-25), (-33)	18, 36, 72
MT49H8M36FM	Micron	CIO	x36	(-5), (-25), (-33)	36, 72
MT49H32M9CFM	Micron	SIO	x9	(-5), (-25), (-33)	9, 18, 36, 72
MT49H16M18CFM	Micron	SIO	x18	(-5), (-25), (-33)	18, 36, 72

Architecture

Figure 6-1 shows the top-level block diagram of the RLDRAM II memory controller.

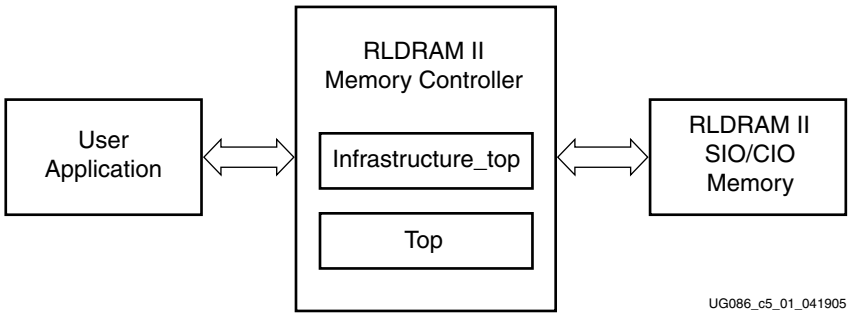


Figure 6-1: **RLDRAM II Memory Controller Block Diagram**

Figure 6-2 shows the hierarchical structure of the controller.

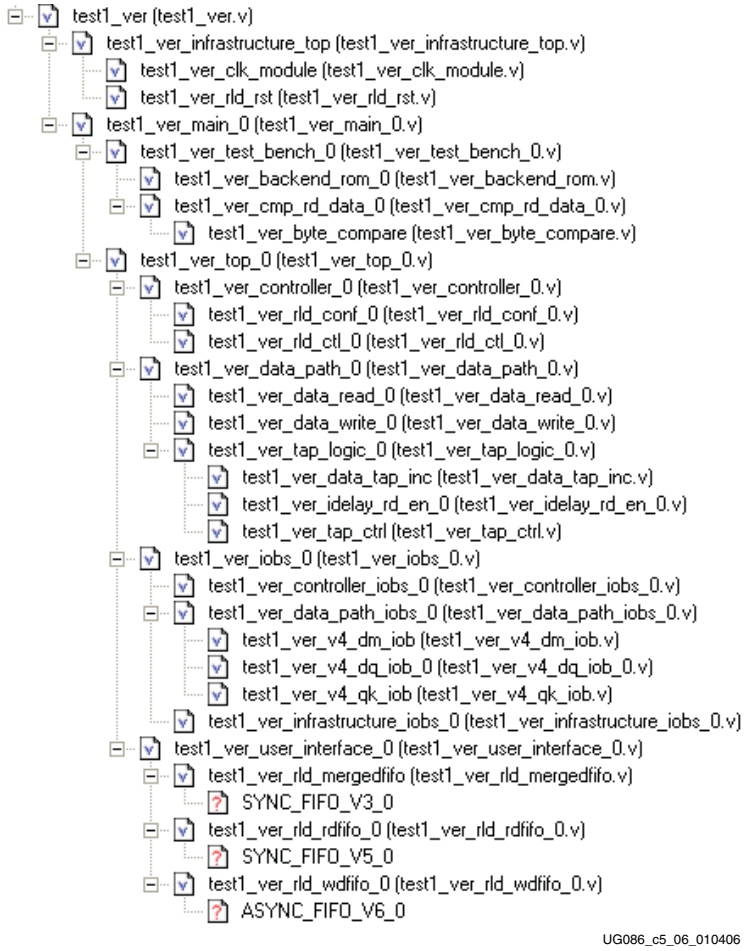


Figure 6-2: **RLDRAM II Memory Controller Hierarchy**

The RLDRAM II memory controller processes the user commands to generate the RLDRAM II interface signals. The RLDRAM II memory controller has a built-in synthesizable test bench to generate all the RLDRAM commands. The built-in test bench enables simulation and validation of the design in hardware. To interface with the user application, the RLDRAM II memory controller must be separated from the built-in test bench. The MIG 1.5 design tool provides an option to generate the design without the test bench. The following parameters are selectable through the GUI: the type of the RLDRAM (SIO or CIO), the data width, the burst length, multiplexed or non-multiplexed address, memory component, and other configuration values.

The design can use any selected banks of the Virtex-4 FPGAs. It can use different banks or the same banks for data, address, and control signals.

The HSTL_II_18 I/O standard is used for address, data, and control signals. The DIFF_HSTL_II_DCI_18 I/O standard is used for clock signals.

Similar to other DRAM architectures, the RLDRAM II requires its entire content to be refreshed periodically. The AREF command initiates a refresh for the device and must be used each time a refresh is required. The RLDRAM II memory controller has an option to enable the execution of auto-refresh commands periodically. If this option is OFF, the user has to provide the auto-refresh commands at regular intervals.

Implemented Features

This section provides details on the supported features of the RLDRAM II controller.

Address Multiplexing

The RLDRAM II memory controller supports multiplexed and non-multiplexed address modes. Bit A5 of the Mode Register determines whether the address mode is multiplexed (A5 = 1) or non-multiplexed (A5 = 0). In multiplexed address mode, the address is provided to the RLDRAM II memory in two cycles, which are latched into the memory on two consecutive rising clock edges. The advantage of this approach is a maximum of 11 address bits are required to control the RLDRAM II memory.

In multiplexed address mode, the controller outputs an 11-bit address. The user has to properly connect the addresses to the RLDRAM II devices. Table 6-2 provides the address mapping between the controller and the RLDRAM II devices for the multiplexed address mode.

Table 6-2: **Address Mapping in Multiplexed Address Mode**

Address	Address Mapping										
Output Address	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
RLDRAM II Address	A0	A3	A4	A5	A8	A9	A10	A13	A14	A17	A18

CIO/SIO

The RLDRAM II memory controller supports both CIO and SIO memory components. The GUI provides an option to select the required memory components. The separate RLDRAM I/O interface transfers two 18-bit or 9-bit data words per clock cycle at the I/O balls. The read port has dedicated data outputs to support read operations, while the write port has dedicated input balls to support write operations. Output data is referenced to the free-running output data clock. This architecture eliminates the need for high-speed bus turnarounds.

Data Capture Using the Direct Clocking Technique

The read data from the RLDRAM II is captured using the direct clocking technique. In this technique, data is delayed and center-aligned with respect to the internal FPGA clock. In this scheme, the internal FPGA clock captures the read data. The clock/strobe transmitted from the memory determines the delay value for the associated data bits. As a result, there are no restrictions on the number of data bits associated with a strobe. Because the strobe does not need to be distributed to the associated data bits, no additional clocking resources are required. Refer to XAPP701 [Ref 10] for details on this technique.

Memory Initialization

The RLDRAM II device must be powered up and initialized in a predefined manner. The controller handles the initialization sequence as described in this section.

After all power supply and reference voltages are stable and the master clock (RLD_CK and RLD_CK_N) is stable, the RLDRAM II device requires a 200 µs (minimum) delay prior to applying an executable command. After the 200 µs (minimum) delay has passed, three MODE REGISTER SET (MRS) commands are issued. For non-multiplexed addressing, two

dummy commands and one valid MRS command are issued. For multiplexed addressing, four MODE REGISTER SET (MRS) commands are issued, consisting of two dummy commands and two valid MRS commands.

Six clock cycles (t_{MRSC}) after the valid MRS commands, eight AUTO REFRESH commands are issued, one on each bank, separated by 2048 cycles.

Initialization is complete after t_{RC}. The number of clock cycles (t_{RC}) after auto refresh depends on the Mode Register configuration parameter. The RLDRAM II memory controller takes care of the t_{RC} value for different configurations. The device is ready for normal operation as indicated by the init_Done outputs to the application.

Block Diagram Description

Figure 6-3 shows a detailed block diagram of the RLDRAM II memory controller. The major blocks of the controller are described following the figure.

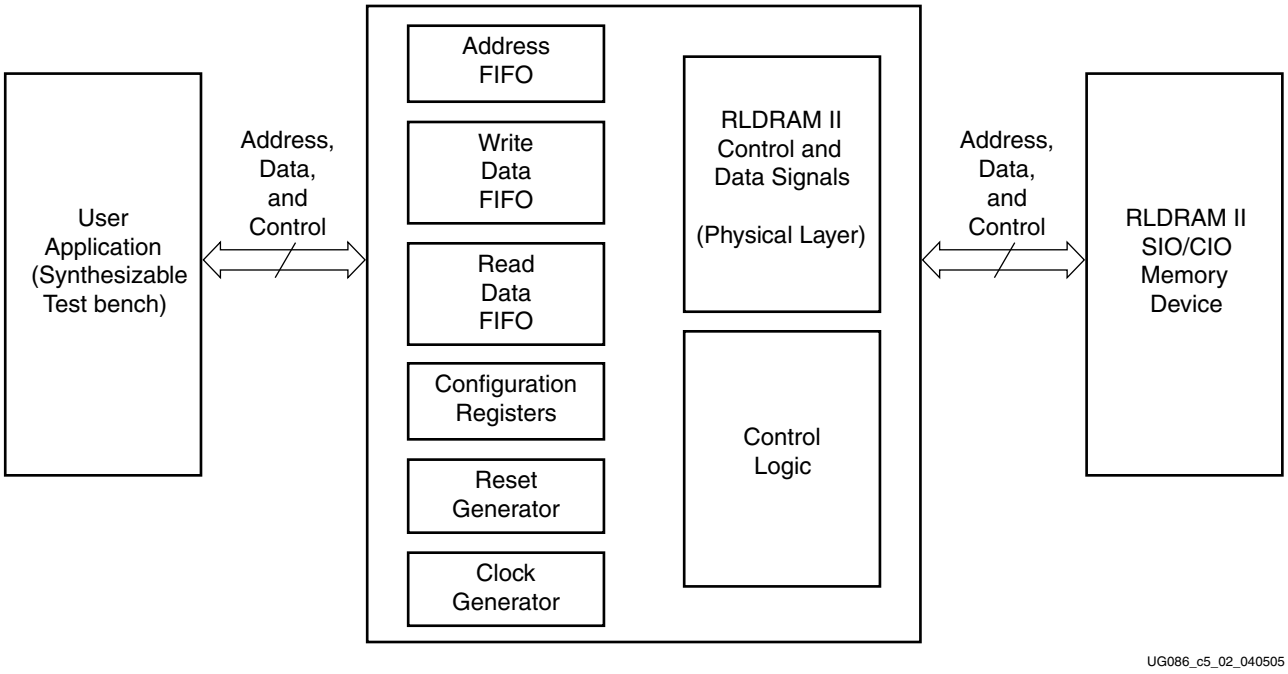


Figure 6-3: **Detailed Block Diagram of the RLDRAM II Memory Controller**

User Interface

The user interface of the RLDRAM II memory controller is a FIFO-based implementation. Three FIFOs are used: an Address FIFO, a Write Data FIFO, and a Read Data FIFO. The user interface also provides a configuration register and additional control signals.

Address FIFO

This FIFO serves as the buffer for the user interface to store addresses corresponding to the read and write data as well as the user-controlled refreshes. All reads, writes, and user refreshes are scheduled in this FIFO. This synchronous FIFO is 26 bits wide and 16 words deep. Table 6-3 defines the configuration of the 26 bits.

Table 6-3: **Address FIFO Bit Configuration**

Bit Configuration	Description
25	User Refresh
24	Read/$\overline{\text{Write}}$
23	Reserved
[22:3]	Memory Address bits A[19:0]
[2:0]	Memory Bank Address bits BA[2:0]

Write Data FIFO

The Write Data FIFO serves as a buffer for the user interface to store data to be written into memory. This asynchronous FIFO is two times the memory data width plus the data mask (DM) width and is 15 words deep. For a burst length of two, each location in the Write Data FIFO comprises the required data. For a burst length of four, two locations in the Write Data FIFO comprise the required data. For a burst length of eight, four locations in the Write Data FIFO comprise the required data.

Table 6-4 defines the FIFO configuration for 36-bit data width using x36 memory components.

Table 6-4: **Write Data FIFO Bit Configuration for 36-bit Data Width**

Bit Configuration	Description
[73:72]	Write Data Mask
[71:0]	Write Data

Read Data FIFO

The Read Data FIFO serves as a buffer for the RLDRAM II memory controller to store data it has read from the memory. This synchronous FIFO is two times the width of the memory data width and 16 words deep. For x18 memory components, an 18-bit wide Base FIFO is used, and for x36 memory components, a 36-bit wide Base FIFO is used. Multiple Base FIFO instances are used to match the two times memory data width. For x18 components with a 36-bit data width, the Base Read FIFO width is 18 bits. Four Read FIFO instances are used to get two times the memory data width. For a burst length of two, each location in the Read Data FIFO constitutes the data read from the memory. For a burst length of four, two locations in the Read Data FIFO constitute the data read from the memory. For a burst length of eight, four locations in the Read Data FIFO constitute the data read from the memory.

Table 6-5 defines the configuration of the Read Data FIFO for the selected memory data width of 36 bits.

Table 6-5: **Read Data FIFO Bit Configuration for a 36-bit Data Width**

Bit Configuration	Description
[71:0]	Read Data

Configuration Registers

This block provides an interface for the application to read from and write to the Configuration Registers. Table 6-6 shows the internal configuration register read and write details from the user interface. A four-bit address from the user interface selects the internal controller register that is to be read or written. Eight bits can be read or written at a time to the selected register.

Table 6-6: **Configuration Read/Write Details from the User Interface**

ApConfA[3:0] (Address)	Register Selected	ApConfWr	ApConfRd	Description
0000	confMReg[7:0]	High	Low	ApConfWrD[7:0] from the user interface is loaded into confMReg[7:0].
0000	confMReg[7:0]	Low	High	confMReg[7:0] data is read into bits ApConfRdD[7:0].
0011	confRcCnt0[6:0]	High	Low	ApConfWrD[6:0] from the user interface is loaded into register confRcCnt0[6:0].
0011	confRcCnt0[6:0]	Low	High	confRcCnt0[6:0] data is read into bits ApConfRdD[6:0]
1010	confMReg[9:8]	High	Low	ApConfWrD[1:0] from the user interface is loaded into confMReg[9:8].
1011	confCycRef	High	Low	apConfWrD[0] from the user interface is loaded into register confCycRef.

Auto refresh is ON by default, making the RLDRAM II memory controller send AREF commands to the memories at the required intervals. The user can turn auto refresh OFF via the confCycRef bit (an internal configuration bit that the user can update and read through the configuration read/write access port). In this case, the user is responsible for issuing USER REFRESH commands at required intervals.

The burst length can be changed from the GUI through the Mode Register settings or programmed from the user interface.

Clock Generator

This block generates all the required clocks for the RLDRAM II memory controller by using a DCM. The two clock phases output are 0 degrees and 90 degrees. The 200 MHz reference clock buffer is included in this module. This clock goes to all IDELAYCTRL primitives.

Reset Generator

This block generates different reset signals. It also performs the initialization and configuration (MRS) of the RLDRAM II memories.

Control Logic

The logic in this block controls NOP, READ, WRITE, and USER REFRESH operations from and to the memories. The RLDRAM II memory controller is triggered with data in the Address FIFO. Bit 24 of the Address FIFO discriminates between read and write commands. Bit 25 is the USER REFRESH command. If the auto refresh bit is ON, the

controller generates the AUTO REFRESH command periodically. The controller issues a read or a write grant only when there is no user refresh request command or no pending internal refresh request. If there is a pending refresh request, the RLDRAM II memory controller issues the read or the write grant after the refresh is done.

RLDRAM II Control Signal Physical Layer

This block has the pads that interface with the RLDRAM II data signals. A calibration circuit samples the QK/$\overline{QK}$ signals using the Virtex-4 ChipSync™ feature. The FPGA clock samples both the data and clock (for calibration) and the data itself to capture it in the same clock domain. Refer to XAPP701 [Ref 10] for more details.

RLDRAM II Interface Signals

Table 6-7 defines the RLDRAM II interface signals.

Table 6-7: **RLDRAM II Signal Descriptions**

Pin Name	Pin Direction	Description
System Signals		
sysClk_p sysClk_n	In	System clock input made up of differential clock pairs. This clock pair goes to a differential input buffer. The differential buffer output goes to the DCM input. The DCM generates the required clocks for the design.
CLK200	In	200 MHz reference clock for IDELAYCTRL (taps)
CLK50	In	50 MHz clock, which goes to DELAY elements.
clkGlob	Out	Clock output from the DCM (main FPGA clock)
clk90	Out	CLK90 output from the DCM
sysReset	In	Active-High system reset
rstHard	Out	Active-High reset until the DCM is locked. This signal is synchronous with clkGlob.
rstHard_90_o	Out	Active-High reset until DCM is locked. This signal is synchronous with clk90.
Init_Done	Out	When asserted, this signal indicates that memory initialization is complete. This signal is synchronous with clkGlob.
Init_Done_270	Out	This signal is init_done. It is synchronous with clk270.
dcmLocked	Out	When asserted, this signal indicates that the DCM is locked. This signal is synchronous with clkGlob.
issueMRS	In	A pulse on this input makes the controller program the Mode Register into the memory. This signal is synchronous with clkGlob. (At power up, MRS is done as part of the initialization.)
Interface to User Application		
ApAdd[25:0]	In	Address FIFO data input. This bus is synchronous with clkGlob (the FPGA clock).

Table 6-7: **RLDRAM II Signal Descriptions *(Continued)***

Pin Name	Pin Direction	Description
ApConfA[3:0]	In	Configuration register address bus. This bus is synchronous with clkGlob.
apConfRd	In	Configuration register read enable. This signal is synchronous with clkGlob.
ApConfRdD[7:0]	Out	Configuration register read data. This bus is synchronous with clkGlob.
ApConfWr	In	Configuration register write data valid. This signal is synchronous with clkGlob.
apConfWrD [7:0]	In	Configuration register write data. This signal is synchronous with clkGlob.
apRdfRdEn	In	Read Data FIFO read enable. This signal is synchronous with clk90.
apValid	In	Address FIFO data valid input. This signal is synchronous with clkGlob.
apWriteData [(2*datawidth)-1:0]	In	Write Data FIFO data input. This bus is synchronous with clkGlob.
apWriteDM [(2*Dmwidth)-1:0]	In	Write Data FIFO data mask (DM) input. This signal is synchronous with clkGlob.
apWriteDValid	In	Write Data FIFO data valid input. This signal is synchronous with clkGlob.
BurstLength[1:0]	Out	These signals indicate the selected burst length in bits A4:A3 of the Configuration Mode Register.
rlafEmpty	Out	Address FIFO empty flag. This signal is synchronous with clkGlob.
rlaFull	Out	Address FIFO full flag. This signal is synchronous with clkGlob.
rldReadData [(2*datawidth)-1:0]	Out	Read Data FIFO data output. This bus is synchronous with clkGlob. The width of the FIFO is twice the width of the selected data.
rldWriteDone	Out	When asserted, this signal indicates that a write to the memory is complete. This signal is synchronous with clkGlob. This signal is asserted when all memory write commands loaded into the Address FIFO have completed.
rlRdfEmpty	Out	Read Data FIFO empty flag. This signal is synchronous with clkGlob.
rlRdfFull	Out	Read Data FIFO full flag. This signal is synchronous with clkGlob.
rlWdfEmpty	Out	Write Data FIFO empty flag. This signal is synchronous with clkGlob.

Table 6-7: **RLDRAM II Signal Descriptions *(Continued)***

Pin Name	Pin Direction	Description
rlWdfFull	Out	Write Data FIFO full flag. This signal is synchronous with clkGlob.
Interface to CIO/SIO RLDRAM II Devices		
RLD_DQ[35:0] (CIO)	In/Out	Data input/outputs. During READ commands, the data is sampled on both edges of QK. During WRITE commands, the data is sampled on both edges of DK.
RLD_D[17:0] (SIO)	Out	Write data.
RLD_Q[17:0] (SIO)	In	Read data.
RLD_ADDRESS[21:0]	Out	Row and column addresses for READ and WRITE operations. During a MODE REGISTER SET command, the address inputs define the register settings.
RLD_BA[2:0]	Out	These bank addresses select the internal bank to which to apply commands
RLD_CK[1:0] RLD_CK_N[1:0]	Out	Master differential clocks for addresses and commands.
RLD_CS_N[1:0]	Out	Chip select command.
RLD_DK[1:0] RLD_DK_N[1:0]	Out	Differential write data clocks.
RLD_DM[1:0]	Out	Data mask signals for write data.
RLD_REF_N	Out	Refresh command.
RLD_WE_N	Out	Write-enable command.
RLD_QK [3:0] RLD_QK_N [3:0]	In	Differential read data clocks. These clocks are transmitted by the RLDRAM II devices and are edge-aligned with read data.
RLD_QVLD[1:0]	In	Data valid signals transmitted by the RLDRAM II devices. They indicate valid read data.

User Command Interface

The current implementation supports commands that come in successive clocks with one extra clock latency.

Write Commands

Figure 6-4 is a timing diagram of a write burst with burst length set to four.

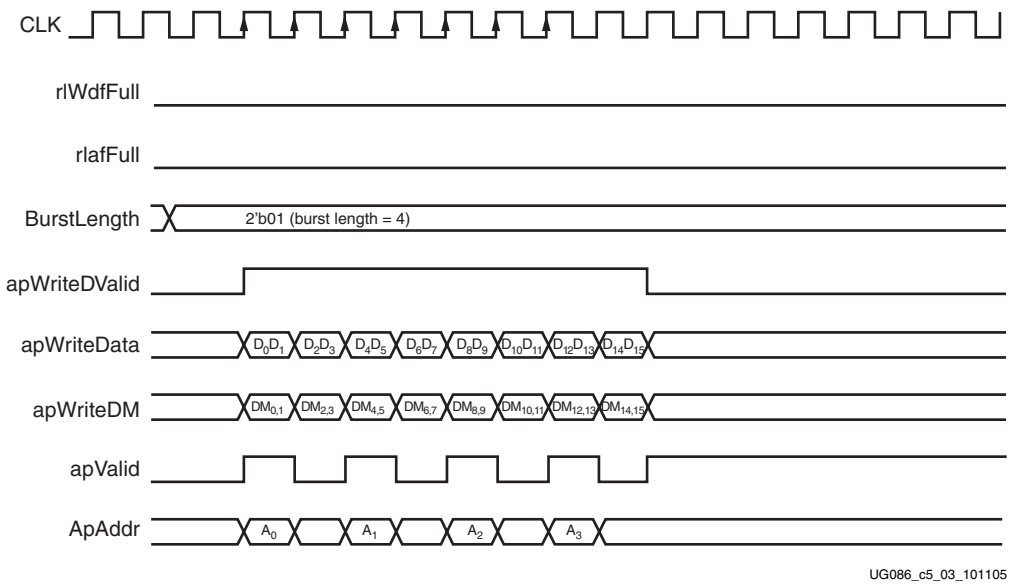


Figure 6-4: **RLDRAM II Write Burst Timing Diagram (BL = 4), Four Bursts**

The rlWdfFull signal is asserted to indicate the Write Data FIFO is full, and the rlafFull signal is asserted to indicate the Address FIFO is full. The BurstLength signal indicates the selected burst length.

The apWriteDValid signal is the write-enable signal, and apWriteData is the data input for the Write Data FIFO. The apValid signal is the write-enable signal for the Address FIFO, and apAddr is the data input (burst write address for RLDRAM II memory) for the Address FIFO.

The user loads the Write Data FIFO and Write Address FIFO with memory data and addresses as shown in Figure 6-5. The Address FIFO is loaded with the burst address when the Write Data FIFO contains enough data. For a burst length of two, one location in the Write Data FIFO contains two data words (the width of the Write Data FIFO is two times the width of the memory). For a burst length of four, two locations in the Write Data FIFO contain the required data. For a burst length of eight, four locations in the Write Data FIFO contain the required data. When the rlWdfFull signal is asserted, the user stops writing data on the next clock. The RLDRAM II memory controller asserts the rlWdfFull signal four locations before the FIFO is full (this is the default setting). The user can reset this parameter to a required value.

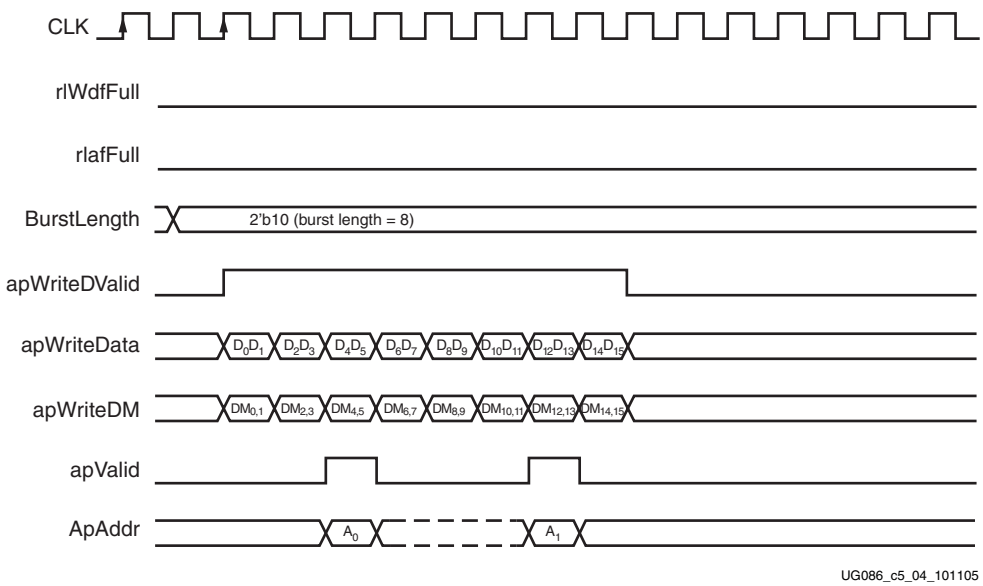


Figure 6-5: **RLDRAM II Write Burst Timing Diagram (BL = 8), Two Bursts**

Bit 24 of the Address FIFO is the command bit. A zero on bit 24 indicates a WRITE command, and a one indicates a READ command. Bit 25 of the Address FIFO indicates a REFRESH command.

The RLDRAM II memory controller reads the Address FIFO when the empty status of the Address FIFO goes Low. When there is no pending internal refresh request and the command is not a REFRESH command, the controller initiates the read or write operation. After a WRITE command is initiated to RLDRAM II memory, the RLDRAM II memory controller enables reading of the Write Data FIFO after a write latency period. The number of words read from the Write Data FIFO depends upon the burst length. The RLDRAM II memory controller reads the address FIFO periodically depending on the burst length. For burst lengths of two and four, the controller reads in alternate clocks. For a burst length of eight, the controller reads the Address FIFO once every four clocks. The RLDRAM II memory controller checks for the t_{RC} before issuing the commands to memory. It waits until the t_{RC} time is met, if there is a t_{RC} violation.

Read Commands

Figure 6-6 is a READ command timing diagram. The READ command is loaded into the Address FIFO from the user interface when the Address FIFO full status signal is Low. The RLDRAM II memory controller initiates the READ command to RLDRAM II memory if the controller is in the IDLE condition. The controller's IDLE condition indicates there are no pending internal refreshes, no external user refresh commands, no ongoing write commands, and no previous read commands. Read data from RLDRAM II memory is written into the Read Data FIFO. When the Read Data FIFO empty status signal goes Low, data is read from the user interface when the apRdfRdEn signal is asserted. Data is first available in the Read Data FIFO approximately 22 clock cycles after the command is loaded into the Address FIFO.

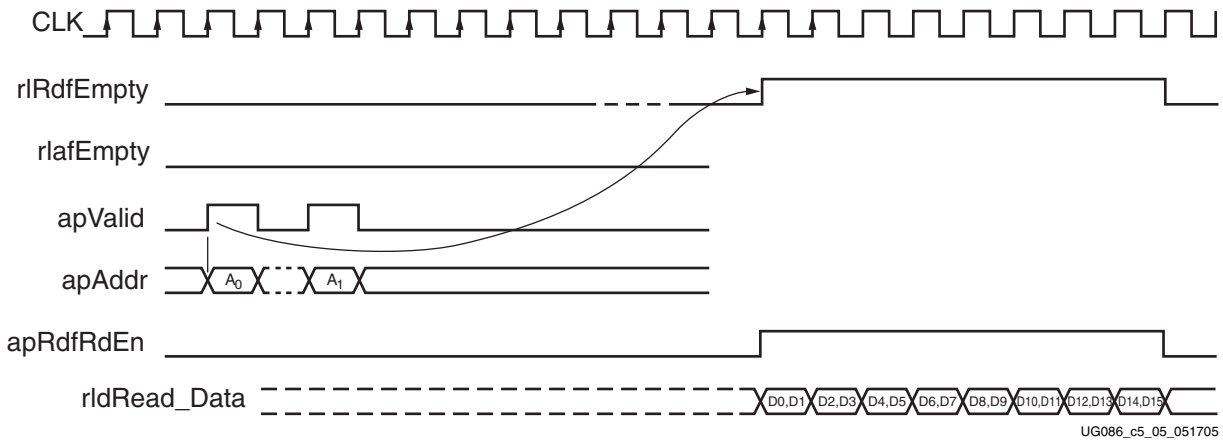


Figure 6-6: **RLDRAM II Read Burst Timing Diagram (BL = 8), Two Bursts**

Refresh Commands

The confCycRef bit controls the auto refresh functionality. The user can update or read this bit through the configuration read/write access port. If the confCycRef bit is set to one, auto refresh is ON, making the controller send AREF commands to the memories at the required intervals. To turn auto refresh OFF, the user clears the confCycRef bit. In this case, the user is responsible for issuing refresh commands.

Section 2: Spartan-3 FPGA to Memory Interfaces

XILINX®

Chapter 7

Implementing DDR SDRAM Controllers

This chapter describes how to implement DDR SDRAM interfaces for Spartan-3 and Spartan-3E FPGAs created with the MIG 1.5 tool. This design is based on XAPP768c [Ref 16], which is available only under NDA.

Feature Summary

The DDR SDRAM controller design supports:

- Burst lengths of two, four, and eight
- CAS latencies of 2, 2.5, and 3
- Auto refresh
- Spartan-3 maximum frequency:
  - 133 MHz with a -4 speed grade device
  - 166 MHz with a -5 speed grade device
- Spartan-3E maximum frequency:
  - 90 MHz with a Stepping 0 device
  - 133 MHz with a Stepping 1 device
- Components, unbuffered DIMMs, registered DIMMs, and SODIMMs
- With and without test bench
- With and without DCM
- All Spartan-3 and Spartan-3E FPGAs
- Verilog and VHDL
- XST, Synplicity, and Precision synthesis tools

Controller Architecture

DDR SDRAM Interface

High-speed memory interfaces are source-synchronous and double data rate. They transfer data on both edges of the clock cycle. A memory interface can be modularly represented as shown in Figure 7-1. Creating a modular interface has many advantages. It allows designs to be ported easily, and it also makes sharing parts of the design across different types of memory interfaces possible.

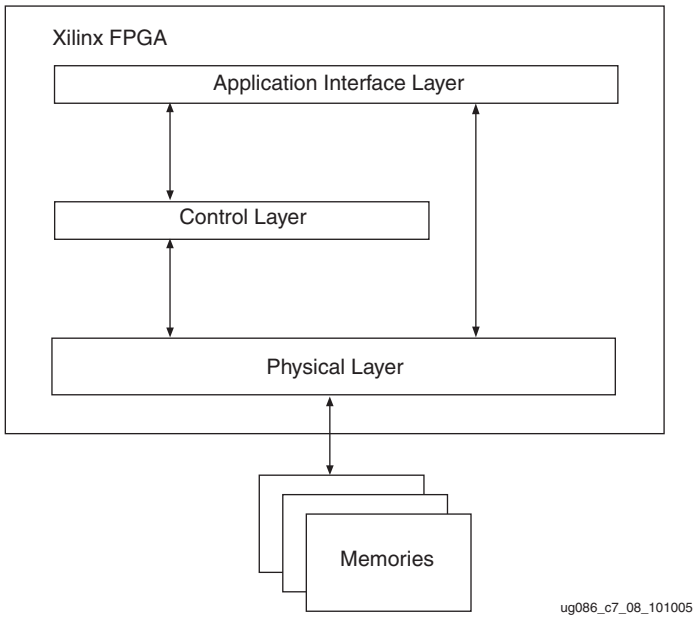


Figure 7-1: **Modular Memory Interface Representation**

Hierarchy

As shown in Figure 7-2, which illustrates the hierarchical structure of the modular design, physical and control layers are clearly separated. The MIG 1.5 tool generates the entire controller, as shown in this hierarchy, including the test bench. The user can replace the test bench with a design that makes use of the DDR SDRAM interface.

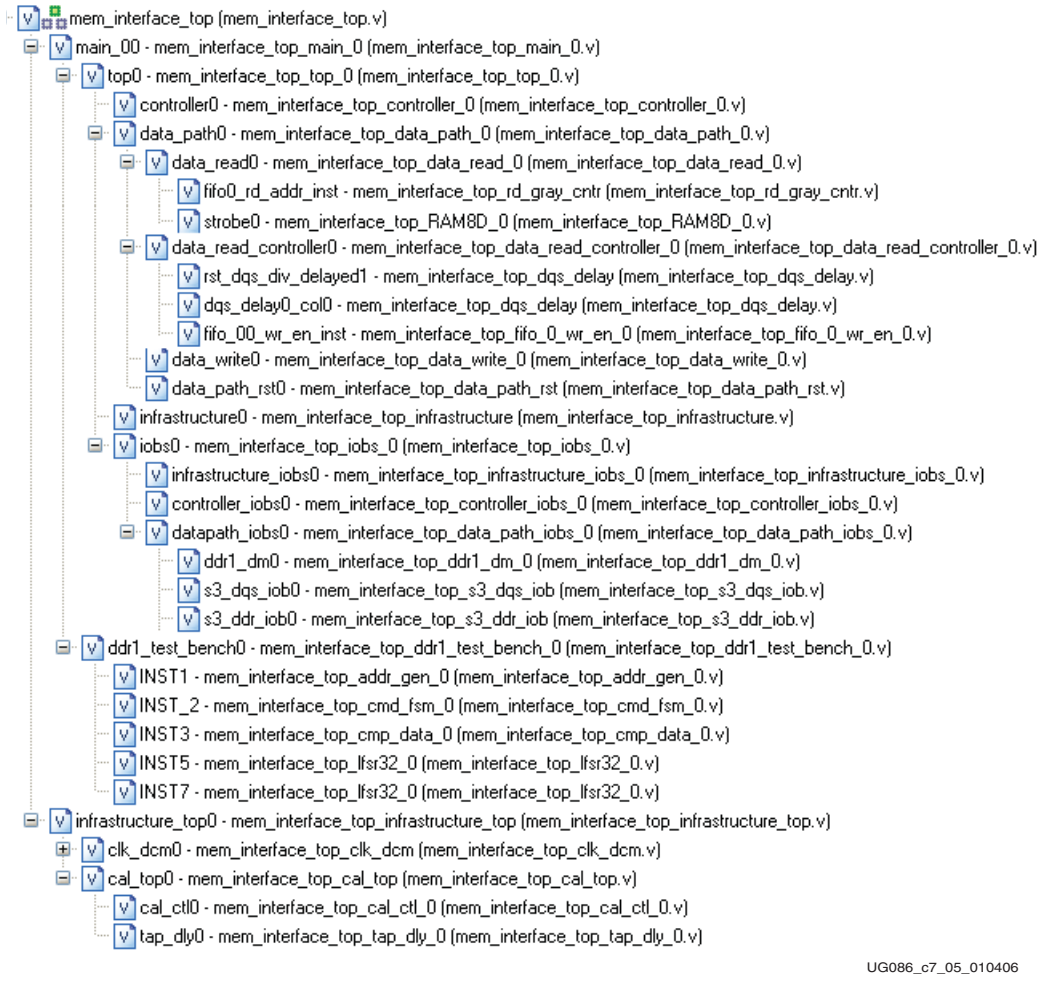


Figure 7-2: **Hierarchical Structure of the Design**

Figure 7-3 shows a detailed block diagram of the DDR SDRAM controller. All four blocks shown are sub-blocks of the ddr1_top module. The functionalities of these blocks are explained in following sections.

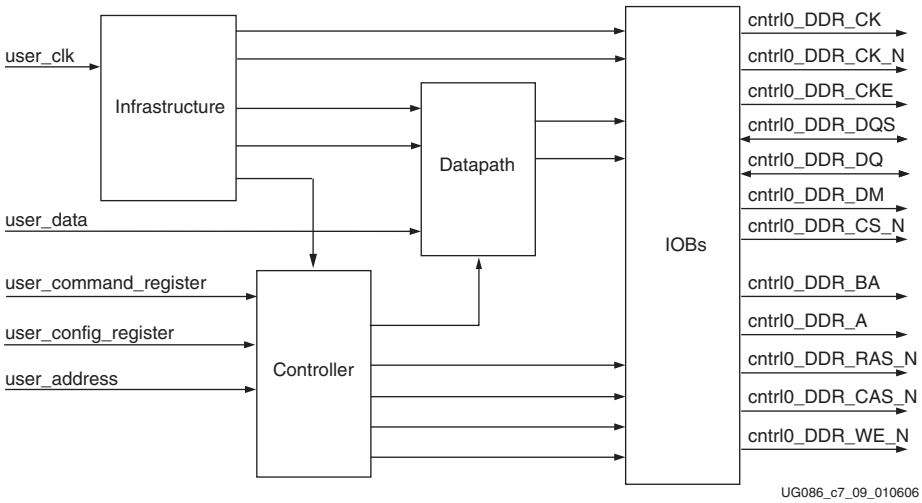


Figure 7-3: **Memory Controller Block Diagram**

Controller

The controller module accepts and decodes user commands and generates read, write, memory initialization, and load mode commands. The controller also generates signals for other modules.

The memory is initialized and powered up using a defined process. The controller state machine handles the initialization process upon receiving an initialization command.

Datapath

This module transmits and receives data to and from the memories. Major functions include storing the read data and transferring write data and write enable to the IOBS module. The data_read, data_write, data_path_IOBs, and data_read_controller modules perform the actual read and write functions. For more information, refer to XAPP768c [Ref 16].

Data Read Controller

This module generates all control signals that are used for data_read.

Data Read

The data_read module contains the read datapaths for the DDR SDRAM interface. Details for this module are described in XAPP768c [Ref 16].

Data Write

This module contains the write datapath for the DDR SDRAM interface. The write data and write enable signals are forwarded together to the DDR SDRAM through IOB flip-flops. The IOBs are implemented in the datapath_IOBs module.

Infrastructure

The infrastructure module generates the FPGA clock and reset signals. A DCM generates the clock and its inverted version. The calibration circuit is also implemented in this module. If there is no DCM, the clocks are driven from the user interface.

IOBs

All input and output signals of the FPGA are implemented in the IOB registers.

Interface Signals

Table 7-1 lists the DDR SDRAM interface signals, directions, and descriptions to and from DDR SDRAM controller. The signal direction is with respect to the DDR SDRAM controller.

Table 7-1: **DDR SDRAM Interface Signal Descriptions**

Signal Name	Signal Direction	Description
cntrl0_DDR_A	Output	Address
cntrl0_DDR_DQ	Input/Output	Data
cntrl0_DDR_DQS	Input/Output	Data Strobe
cntrl0_DDR_RAS_N	Output	Command
cntrl0_DDR_CAS_N	Output	Command
cntrl0_DDR_WE_N	Output	Command
cntrl0_DDR_BA	Output	Bank Address
cntrl0_DDR_CK	Output	Clock
cntrl0_DDR_CK_N	Output	Inverted Clock
cntrl0_DDR_CS_N	Output	Chip Select
cntrl0_DDR_CKE	Output	Clock Enable
cntrl0_DDR_DM	Output	Data Mask

Table 7-2 lists the DDR SDRAM clock and reset signals.

Table 7-2: **DDR SDRAM Clock and Reset Signals**

Signal Name	Direction	Description
SYS_CLK and SYS_CLKb	Input	These signals are the system clock differential signals. They are driven from the user application for designs with DCMs. These two signals are given to a differential buffer, and the output of the differential buffer is connected to a clock's DCM. The DCM generates the required clocks to the design modules. These signals are not present when the design is generated without a DCM. When there is no DCM, the user application should drive the required clocks to the design.
Clk_int and clk90_int	Input	These signals are the design clocks used in all modules. These clocks are to be driven from the user application only when the DDR SDRAM controller is generated without a DCM.
Reset_in	Input	This signal is the system reset signal.

Table 7-3 describes the DDR SDRAM controller user interface signals used between the ddr1_top (design top-level module) and user application modules. All the user interface signals in Table 7-3 are between the DDR SDRAM controller and user application. The design can be generated with or without a built-in test bench. When the design is generated with the built-in test bench option, the user interface signals are between the internal test bench and the design. Otherwise they are between the user application and the design.

Table 7-3: **DDR SDRAM Controller User Interface Signals**

Signal Names	Direction(1)	Description
user_input_data[(2*n*−1):0]	Input	This bus is the write data to the DDR SDRAM from the user interface, where *n* is the width of the DDR SDRAM data bus. The DDR SDRAM controller converts single data rate to double data rate on the physical layer side. The data is valid on the DDR SDRAM write command. In 2*n*, the MSB is rising-edge data and the LSB is falling-edge data.
user_data_mask[(2*m*−1):0]	Input	This bus is the data mask for write data. Like user_input_data, it is twice the size of the data mask bus at memory, where *m* is the size of the data mask at the memory interface. In 2*m*, the MSB applies to rising-edge data and the LSB applies to falling-edge data.
user_input_address [row_address + col_ap_width + bank_address–1: bank_address] (2)	Input	This bus is the DDR SDRAM row, column, and bank address. This bus is the combination of row, column, and bank addresses for DDR SDRAM writes and reads. For example, if row_address = 13, column_address = 9, bank_address = 2, and user_input_address = 26, then: • Bank Address = A[1:0] • Column Address = A[12:2] • Row Address part = A[25:13] To drive the A10 bit during precharge, the column address is extended by one or two bits depending on the number of column address bits.

Table 7-3: **DDR SDRAM Controller User Interface Signals *(Continued)***

<table>
<tr><th>Signal Names</th><th>Direction(1)</th><th>Description</th></tr>
<tr><td>user_config_register[9:0]</td><td>Input</td><td>This is the configuration data for DDR SDRAM initialization. The format for user_config_reg is as follows:
<table>
<tr><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>EMR (Enable/ Disable) DLL</td><td>BMR</td><td>BMR/ EMR</td><td colspan="3">CAS_latency</td><td>Burst_ type</td><td colspan="2">Burst_length</td></tr>
</table>
Bits 9, 8, and 7 are reserved. DDR controller supports only sequential Burst types.
The user_config_register bits are used during DDR SDRAM initialization of the Load Mode Register. Refer to the DDR SDRAM data sheet for details.</td></tr>
<tr><td>user_command_register[2:0]</td><td>Input</td><td>Supported user commands for the DDR SDRAM controller:
<table>
<tr><th>user_command[2:0]</th><th>User Command Description</th></tr>
<tr><td>000</td><td>NOP</td></tr>
<tr><td>010</td><td>Memory (DDR SDRAM) initialization</td></tr>
<tr><td>100</td><td>Write</td></tr>
<tr><td>101</td><td>Load Mode Register</td></tr>
<tr><td>110</td><td>Read</td></tr>
<tr><td>Others</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>burst_done</td><td>Input</td><td>This is the data transfer done signal. The user asserts this signal for two clock periods at the end of the data transfer. The DDR SDRAM controller supports write burst or read burst capability for a single row. The user must terminate the transfer on a column boundary and must re-initialize the controller for the next row of transactions on a column boundary.</td></tr>
<tr><td>user_output_data[(2n−1):0]</td><td>Output</td><td>This is the read data from the DDR SDRAM. The DDR SDRAM controller converts the DDR data from the DDR SDRAM to SDR data. As the DDR data is converted to SDR data, the width of this bus is 2n, where n is data width of the DDR SDRAM data bus.</td></tr>
<tr><td>user_data_valid</td><td>Output</td><td>When asserted, this signal indicates user_output_data[(2n−1):0] is valid.</td></tr>
<tr><td>user_cmd_ack</td><td>Output</td><td>This is the acknowledgement signal for a user read or write command. It is asserted by the DDR SDRAM controller during a write or read to/from the DDR SDRAM. The user should not issue any new commands to the controller until this signal is deasserted.</td></tr>
<tr><td>init_val</td><td>Output</td><td>The DDR SDRAM controller asserts this signal to indicate that the DDR SDRAM initialization is complete.</td></tr>
</table>

Table 7-3: **DDR SDRAM Controller User Interface Signals *(Continued)***

Signal Names	Direction(1)	Description
ar_done	Output	This indicates that the auto-refresh command was given to DDR SDRAM. The DDR SDRAM controller asserts this signal for one clock after giving an auto-refresh command to the DDR SDRAM and completion of T_{RFC} time. The T_{RFC} time is determined by the rfc_count_value in the parameter file. T_{RFC} is the minimum time required for the DDR SDRAM to complete the refresh command.
auto_ref_req	Output	This signal is asserted on every 7.7 µs. It is asserted until the controller issues an auto_refresh command to the memory. Upon seeing this signal, the user should terminate any ongoing command after completion of the current burst cycle by asserting the burst_done signal. To ensure reliable operation, users should terminate the current command within 15 to 20 clock cycles after auto_ref_req is asserted. The frequency with which this signal is asserted is determined by the max_ref_cnt value in the parameter file. The max_ref_cnt value is set in the parameter file based on the frequency selected from the tool.

Notes:

1. All of the signal directions are with respect to the DDR SDRAM controller.
2. col_ap_width includes an autoprecharge (A10) bit 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 for autoprecharge. The col_ap_width parameter is used internally for prepending the A10 bit during the precharge command. For a column address of 10, col_ap_width is defined as 11. For an 11-bit column address, col_ap_width is defined as 12. The user need consider only the column address parameter for driving the column address to the memory.

Resource Utilization

A local inversion clocking technique is used in this design. The DCM generates only clk0 and clk90. One DCM and two BUFGs are used. The Spartan-3 designs operate at 166 MHz and below.

DDR SDRAM Initialization

Before issuing the memory read and write commands, the controller initializes the DDR SDRAM using the memory initialization command. The user can give the initialization command only after all reset signals are deactivated. The controller is in the reset state for 200 µs after power up. For design optimization, a 200 µs timer is generated from the refresh counter. The refresh timer is a function of frequency. Therefore, at lower frequencies, the 200 µs timer waits more than 200 µs. Because wait200 happens only during the power-up sequence, design performance is not degraded. All resets are asserted for 200 µs because DDR SDRAM requires a 200 µs delay prior to applying an executable command after all power supply and reference voltages are stable. The controller asserts clock-enable to memory after 200 µs.

The user places the data to be written into the Load Mode Register on user_config_register[7:0] until the DDR SDRAM initialization is complete. The controller then writes this data into the Load Mode Register. Once the DDR SDRAM is initialized, the DDR SDRAM controller asserts the init_val signal.

Figure 7-4 shows the timing for the memory initialization command.

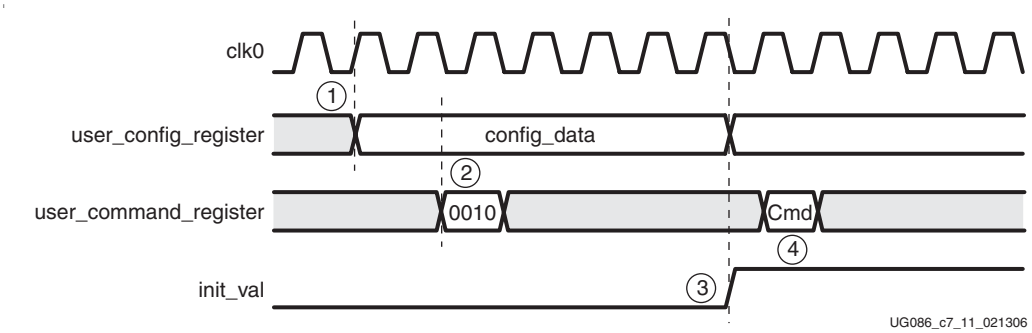


Figure 7-4: **DDR SDRAM Initialization**

1. The user places valid configuration data on user_config_register[9:0] two clocks prior to placing the initialization command on user_command_register[2:0].
2. The user places the initialization command on user_command_register[2:0] on a falling edge of clk0 for one clock cycle. This starts the initialization sequence.
3. The DDR SDRAM controller indicates that the configuration is complete by asserting the init_val signal on a falling edge of clk0. The init_val signal is asserted throughout the period.
4. After init_val is asserted, the user can pass the next command at any time.

DDR SDRAM Write

Figure 7-5 shows the timing diagram for a write to DDR SDRAM with a burst length of four. The user initiates the write command by sending a Write instruction to the DDR SDRAM controller. To terminate a write burst, the user asserts the burst_done signal for two clocks after the last user_input_address. For a burst length of two, the burst done signal should be asserted for one clock. For a burst length of four, the burst done signal should be asserted for two clocks. For a burst length of eight, the burst done signal should be asserted for four clock cycles.

The write command is asserted on the falling edge of clk0. In response to a write command, the DDR SDRAM controller acknowledges with the usr_cmd_ack signal on a falling edge of clk0. The usr_cmd_ack signal is generated in the next clock after the write command is asserted, if the controller is not busy. If there is an ongoing refresh command, the usr_cmd_ack signal is asserted after completion of the refresh command. The user asserts the first address (row + column address) with the write command and keeps it asserted for three clocks after usr_cmd_ack assertion. Any subsequent write addresses are asserted on an alternate falling edge of clk0 after deasserting the first memory address. For a burst length of two, subsequent addresses are asserted on each clock cycle, and for a burst length of eight, subsequent addresses are asserted once every four clock cycles. The first user data is asserted on a rising edge of clk90 after usr_cmd_ack is asserted. As the SDR data is converted to DDR data, the width of this bus is 2*n*, where *n* is data width of DDR SDRAM data bus.

For a burst length of four, only two data words (each of 2*n*) are given to the DDR SDRAM controller for each user address. For a burst length of two, one data word is passed for each burst. For a burst length of eight, four data words are passed for each burst. Internally, the DDR SDRAM controller converts into four data words, each of *n* bits. To terminate the write burst, the user asserts burst_done on a falling edge of clk0 for two clocks. The burst_done signal is asserted after the last memory address. Any further commands to the DDR SDRAM controller are given only after the usr_cmd_ack signal is deasserted. After

burst_done is asserted, the controller terminates the burst and issues a precharge to the memory. The usr_cmd_ack signal is deasserted after completion of the precharge.

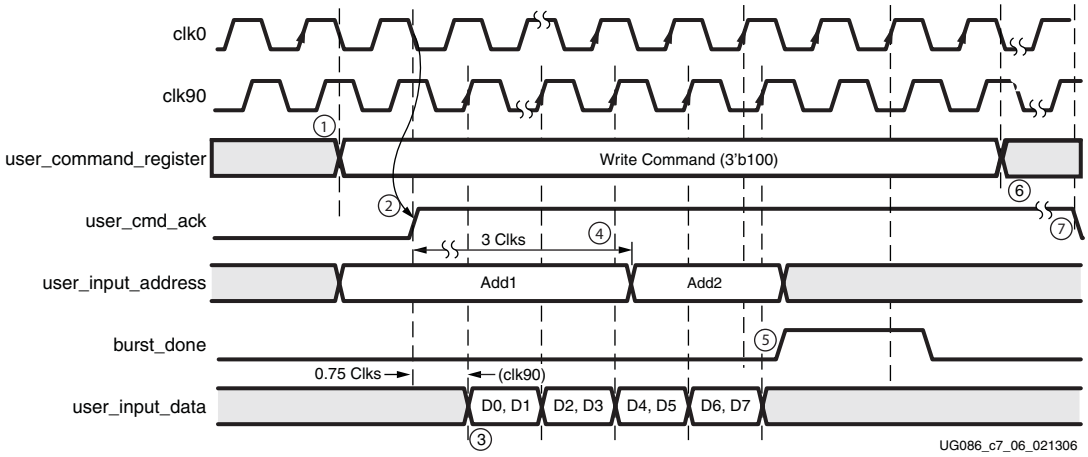


Figure 7-5: **DDR SDRAM Write Burst, Burst Lengths of Four and Two Bursts**

1. A memory write is initiated by issuing a write command to the DDR SDRAM controller. The write command must be asserted on a falling edge of clk0.
2. The DDR SDRAM controller acknowledges the write command by asserting the user_cmd_ack signal on a falling edge of clk0. The earliest this signal is asserted is one clock after the command. The maximum number of clock cycles it takes to assert cmd_ack signal depends on the refresh period.
3. The first user_input_address must be placed along with the command. The input data is asserted with the clk90 signal after the user_cmd_ack signal is asserted.
4. The user asserts the first address (row + column address) with the write command and keeps it asserted for three clocks after usr_cmd_ack assertion. The user_input_address signal is asserted on a falling edge of clk0. All subsequent addresses are asserted on alternate falling edges of clk0 for burst lengths of four, on each clock for burst lengths of two, and once in four clocks for burst lengths of eight.
5. To terminate the write burst, burst_done is asserted after the last user_input_address. The burst_done signal is asserted for two clock cycles with respect to the falling edge of clk0 for burst lengths of four.
6. The user command is deasserted after burst_done is asserted.
7. The controller deasserts the user_cmd_ack signal after completion of precharge to the memory. The next command must be given only after user_cmd_ack is deasserted. Back-to-back write operations are supported only within the same bank and row.

DDR SDRAM Read

The user initiates a memory read with a read command to the DDR SDRAM controller. Figure 7-6 shows the memory read timing diagram for a burst length of four.

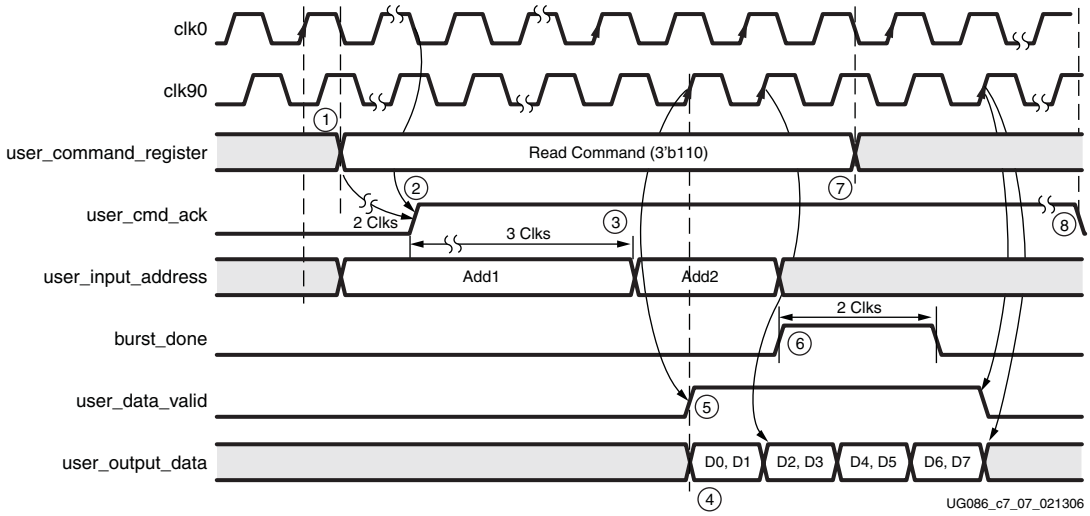


Figure 7-6: **DDR SDRAM Read, Burst Lengths of Four and Two Bursts**

The user provides the first memory address with the read command, and subsequent memory addresses upon receiving the usr_cmd_ack signal. Data is available on the user data bus with the user_data_valid signal. To terminate read burst, the user asserts the burst_done signal on a rising edge of clk0 for two clocks with the deassertion of the last user_input_address. The burst_done signal is asserted for one clock for burst lengths of two, two clocks for burst lengths of four, and four clocks for burst lengths of eight.

The read command flow is similar to the write command flow.

1. A memory read is initiated by issuing a read command to the DDR SDRAM controller. The read command is accepted on a falling edge of clk0.
2. The first read address must be placed along with the read command. In response to the read command, the DDR SDRAM controller asserts the user_cmd_ack signal on a falling edge of clk0. The usr_cmd_ack signal is asserted at least two clocks after the command assertion. This signal is delayed if there is an ongoing refresh cycle, in which case it is asserted after the current refresh command completes.
3. The user asserts the first address (row + column address) with the read command and keeps it asserted for three clocks after usr_cmd_ack is asserted. The user_input_address signal is then accepted on the falling edge of clk0. All subsequent memory read addresses are asserted on alternate falling edges of clk0 for burst lengths of four. The subsequent addresses are changed on every clock for burst lengths of two, on alternate clocks for burst lengths of four, and once in four clocks for burst lengths of eight.
4. The data on user_output_data is valid only when the user_data_valid signal is asserted.
5. The data read from the DDR SDRAM is available on user_output_data, which is asserted with clk90. Because the DDR SDRAM data is converted to SDR data, the width of this bus is 2*n*, where *n* is the data width of the DDR SDRAM memories. For a read burst length of four, the DDR SDRAM controller outputs only two data words

with each user address. For a burst length of two, the controller outputs one data word, and for a burst length of eight, the controller outputs four data words.

6. To terminate the read burst, burst_done is asserted for two clocks on the falling edge of clk0. The burst_done signal is asserted after the last memory address.
7. The user command is deasserted after burst_done is asserted.
8. The controller deasserts the user_cmd_ack signal after completion of precharge to the memory. Any further commands to the DDR SDRAM controller should be given after user_cmd_ack is deasserted. Back-to-back read operations are supported only within the same bank and row. Approximately 17 clock cycles pass between the time a read command is asserted on the user interface and the time data becomes available on the user interface.

Auto_Refresh DDR SDRAM

The DDR SDRAM controller does a memory refresh periodically. Every 7.7 µs, the controller raises an auto_refresh request. The user must terminate any ongoing commands within approximately 15 to 20 clock cycles, when auto_ref_req flag is asserted. The user must assert the burst_done signal at the end of the current burst transaction when sensing the auto_ref_req flag. The auto_ref_req flag is asserted until the controller issues a refresh command to the memory. The user must wait for completion of the auto_refresh command before giving any commands to the controller when auto_ref_req is asserted.

The controller sets the max_ref_cnt value in the parameter file according to the frequency selected for a refresh interval (7.7 µs). The rfc_count_value value in the parameter file defines T_{RFC}, the time between the refresh command to Active or another refresh command.

The ar_done signal is asserted by the DDR SDRAM controller upon completion of the auto_refresh command—i.e., after T_{RFC} time. The ar_done signal is asserted with clk180 for one clock cycle.

After completion of the auto_refresh command, the next command can be given any time after ar_done is asserted.

DDR SDRAM Load Mode

The user gives the LOAD MODE command for one clock on the falling edge of clk0. No acknowledge is given for this command. The user waits for T_{MRD} time for the LOAD MODE command to complete before giving the next command.

Placement of configuration data is similar to that of the initialization command. Configuration data must be placed on user_config_register four clocks prior to giving the LOAD MODE command to the controller.

Constraints

Some constraints are required to successfully create the design. The following examples explain the different constraints in the UCF file.

Calibration Circuit Constraints

All LUTs in the matched delay circuits are constrained to specific locations in the device.

For example:

```
INST "infrastructure_top0/cal_top0/tap_dly0/l0" RLOC_ORIGIN = X50Y104;
```

```
INST "infrastructure_top0/cal_top0/tap_dly0/l0" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/l0" U_SET = "cal_top0/tap_dly0/l0";
INST "infrastructure_top0/cal_top0/tap_dly0/r0" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/r0" U_SET = "cal_top0/tap_dly0/l0";
INST "infrastructure_top0/cal_top0/tap_dly0/r1" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/r1" U_SET = "cal_top0/tap_dly0/l0";
```

Data and Data Strobe Constraints

Data and data strobe signals are assigned to specific pins in the device; placement constraints related to the dqs_delay circuit and the FIFOs used for the data_read module are specified.

Example:

```
NET "cntrl0_DDR_DQS[0]" LOC = Y6;
INST "ddr1_top0/data_path0/data_read_controller0/dqs_delay0_col0/one" LOC =
SLICE_X0Y110;
INST "ddr1_top0/data_path0/data_read_controller0/dqs_delay0_col0/one" BEL = F;
NET "cntrl0_DDR_DQ[0]" LOC = Y4;
INST "ddr1_top0/data_path0/strobe0/fifo0_bit0" LOC = SLICE_X2Y111;
```

The I/O standards for all the memory interface signals are required to be specified. The max delay constraints on some critical nets must also be specified.

I/O Clocking Rules

There are three I/O clocking rules to be followed for I/O pin allocations. Additional information regarding I/O clocking rules is located in DS099 [Ref 17] and DS312 [Ref 18].

I/O Banking Rules

There are I/O banking rules to be followed for I/O pin allocations, stating that the I/O signals allocated in a bank should adhere to compatible I/O standards. Additional information regarding I/O banking rules is located in DS099 [Ref 17] and DS312 [Ref 18].

Design Notes

The DDR SDRAM design with Spartan-3 FPGAs is validated for 100 MHz at the low-end and 166 MHz at the high end. The maximum delay constraints in the UCF file are set to the selected frequency from the tool.

XILINX®

Chapter 8

Implementing DDR2 SDRAM Controllers

This chapter describes how to implement DDR2 SDRAM interfaces for Spartan-3 FPGAs created with the MIG 1.5 tool. This design is based on XAPP768c [Ref 16], which is available only under NDA.

Feature Summary

This section summarizes the supported features of the DDR2 SDRAM controller design.

Supported Features

The DDR2 SDRAM controller design supports:

- Burst lengths of four and eight
- CAS latency of 3
- Auto refresh
- Spartan-3 maximum frequency:
  - 133 MHz with a -4 speed grade device
  - 166 MHz with a -5 speed grade device
- Components, unbuffered DIMMs, and registered DIMMs
- Verilog and VHDL
- XST and Synplicity synthesis tools

Controller Architecture

DDR2 SDRAM Interface

High-speed memory interfaces are source-synchronous and double data rate. They transfer data on both edges of the clock cycle. A memory interface can be modularly represented as shown in Figure 8-1. Creating a modular interface has many advantages. It allows designs to be ported easily, and it also makes sharing parts of the design across different types of memory interfaces possible.

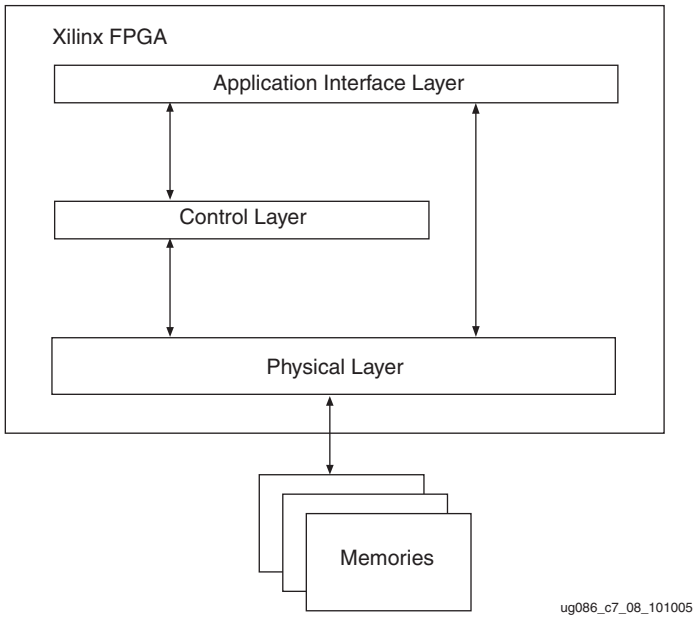


Figure 8-1: **Modular Memory Interface Representation**

Hierarchy

As shown in Figure 8-2, which illustrates the hierarchical structure of the modular design, physical and control layers are clearly separated. The MIG 1.5 tool generates the entire controller, as shown in this hierarchy, including the test bench. The user can replace the test bench with a design that makes use of the DDR2 SDRAM interface.

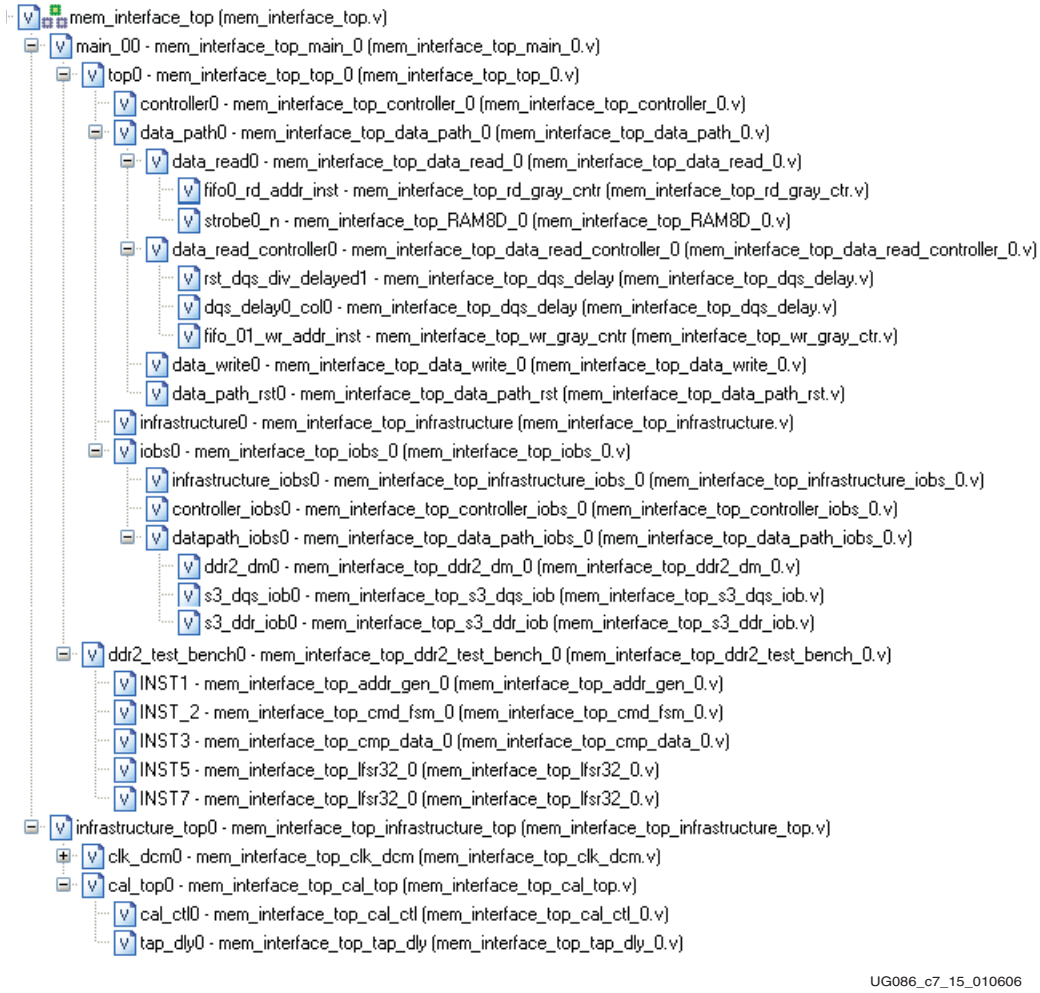


Figure 8-2: **Hierarchical Structure of the Design**

Figure 8-3 shows a detailed block diagram of the DDR2 SDRAM controller. All four blocks shown are sub-blocks of the ddr2_top module. Functionality of these blocks is explained in following sections.

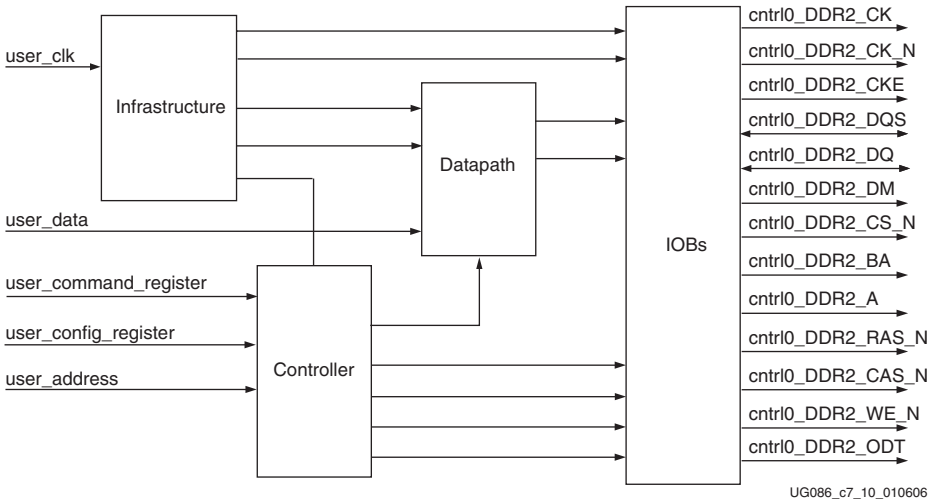


Figure 8-3: **Memory Controller Block Diagram**

Controller

The controller module accepts and decodes user commands and generates read, write, memory initialization, and load mode commands. The controller also generates signals for other modules.

The memory is initialized and powered up using a defined process. The controller state machine handles the initialization process upon receiving an initialization command.

Datapath

This module transmits and receives data to and from the memories. Major functions include storing the read data and transferring write data and write enable to the IOBS module. The data_read, data_write, data_path_IOBs, and data_read_controller modules perform the actual read and write functions. For more information, refer to XAPP768c [Ref 16].

Data Read Controller

This module generates all control signals that are used for the data_read module.

Data Read

The data_read module contains the read datapaths for the DDR2 SDRAM interface. Details for this module are described in XAPP768c [Ref 16].

Data Write

This module contains the write datapath for the DDR2 SDRAM interface. The write data and write enable signals are forwarded together to the DDR2 SDRAM through IOB flip-flops. The IOBs are implemented in the datapath_IOBs module.

Infrastructure

The infrastructure module generates the FPGA clock and reset signals. A DCM generates the clock and its inverted version. The calibration circuit is also implemented in this module.

IOBs

All input and output signals of the FPGA are implemented in the IOBs.

Interface Signals

Table 8-1 shows the DDR2 SDRAM interface signals, directions, and descriptions. The signal direction is with respect to the DDR2 SDRAM controller.

Table 8-1: **DDR2 SDRAM Interface Signal Descriptions**

Signal Name	Signal Direction	Description
cntrl0_DDR2_A	Output	Address
cntrl0_DDR2_DQ	Input/Output	Data
cntrl0_DDR2_DQS	Input/Output	Data Strobe
cntrl0_DDR2_RAS_N	Output	Command
cntrl0_DDR2_CAS_N	Output	Command
cntrl0_DDR2_WE_N	Output	Command
cntrl0_DDR2_BA	Output	Bank Address
cntrl0_DDR2_CK	Output	Clock
cntrl0_DDR2_CK_N	Output	Inverted Clock
cntrl0_DDR2_CS_N	Output	Chip Select
cntrl0_DDR2_CKE	Output	Clock Enable
cntrl0_DDR2_DM	Output	Data Mask
cntrl0_DDR2_ODT	Output	On-Die Termination

Table 8-2 describes the DDR2 SDRAM controller system interface signals.

Table 8-2: **DDR2 SDRAM Controller System Interface Signals**

Signal Names	Direction	Description
SYS_CLK and SYS_CLKb	Input	These are the system clock differential signals.
reset_in	Input	This is the system reset signal.

Table 8-3 describes the DDR2 SDRAM controller system interface signals used between the ddr2_top and user_controller modules.

Table 8-3: **DDR2 SDRAM Controller User Interface Signals**

<table>
<tr><th>Signal Names</th><th>Direction(1)</th><th>Description</th></tr>
<tr><td>user_input_data[(2n−1):0]</td><td>Input</td><td>This bus is the write data to the DDR2 SDRAM from the user interface, where n is the width of the DDR2 SDRAM data bus. The DDR2 SDRAM controller converts single data rate to double data rate on the physical layer side. The data is valid on the DDR2 SDRAM write command. In 2n, the MSB is rising-edge data and the LSB is falling-edge data.</td></tr>
<tr><td>user_data_mask[(2m−1):0]</td><td>Input</td><td>This bus is the data mask for write data. Like user_input_data, it is twice the size of the data mask bus at memory, where m is the size of the data mask at the memory interface. In 2m, the MSB applies to rising-edge data and the LSB applies to falling-edge data.</td></tr>
<tr><td>user_input_address [row_address + col_ap_width + bank_address – 1: bank_address](2)</td><td>Input</td><td>This bus is the DDR2 SDRAM row and column address. It is the combination of the row and column addresses for DDR2 SDRAM writes and reads. Low-order bits specify the column address, and high-order bits specify the row address.</td></tr>
<tr><td>user_bank_address [bank_address – 1:0]</td><td>Input</td><td>This bus is the DDR2 SDRAM bank address.</td></tr>
<tr><td>user_config_register1[14:0]</td><td>Input</td><td>This bus is the configuration data for DDR2 SDRAM initialization. The format for user_config_register1 is as follows:
<table>
<tr><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="2">MR</td><td>PD</td><td colspan="3">WR</td><td>DLL</td><td>TMC</td><td colspan="3">CAS Latency</td><td>BT</td><td colspan="3">Burst Length</td></tr>
</table>
The user_config_register1 bits are used during DDR2 SDRAM initialization of the Load Mode Register. Refer to the DDR2 SDRAM data sheet for details.</td></tr>
<tr><td>user_config_register2[12:0]</td><td>Input</td><td>The user_config_register2 bits control additional functions, including DLL enable/disable, output drive strength, ODT, RTT, posted AL, off-chip driver impedance calibration (OCD), $\overline{\text{DQS}}$ enable/disable, RDQS/$\overline{\text{RDQS}}$ enable/disable, and output enable/disable. These functions are controlled via the bits as follows:
<table>
<tr><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>out</td><td>RDQS</td><td>DQS#</td><td colspan="3">OCD Program</td><td colspan="3">Posted CAS#</td><td colspan="2">RTT</td><td>ODS</td><td>DLL</td></tr>
</table>
The user_config_register2 bits are programmed via the LOAD MODE (LM) command. They retain the stored information until it is reprogrammed or the device loses power. Reprogramming the EMR does not alter the contents of the memory array, provided it is performed correctly.</td></tr>
</table>

Table 8-3: **DDR2 SDRAM Controller User Interface Signals *(Continued)***

<table>
<tr><th>Signal Names</th><th>Direction(1)</th><th>Description</th></tr>
<tr><td>user_command_register[3:0]</td><td>Input</td><td>Supported user commands for the DDR2 SDRAM controller:
<table>
<tr><th>user_command[3:0]</th><th>User Command Description</th></tr>
<tr><td>0000</td><td>NOP</td></tr>
<tr><td>0001</td><td>Precharge</td></tr>
<tr><td>0010</td><td>Auto Refresh</td></tr>
<tr><td>0011</td><td>Self Refresh</td></tr>
<tr><td>0100</td><td>Write Request</td></tr>
<tr><td>0101</td><td>Load Mode Register</td></tr>
<tr><td>0110</td><td>Read Request</td></tr>
<tr><td>0111</td><td>Burst Terminate</td></tr>
</table></td></tr>
<tr><td>burst_done</td><td>Input</td><td>This is the data transfer done signal. The user asserts this signal for two clock periods at the end of the data transfer. The DDR2 SDRAM controller supports write burst or read burst capability for a single row. The user must terminate the transfer on a column boundary and must re-initialize the controller for the next row of transactions on a column boundary.</td></tr>
<tr><td>user_output_data[(2n−1):0]</td><td>Output</td><td>This is the read data from the DDR2 SDRAM. The DDR2 SDRAM controller converts the DDR2 data from the DDR2 SDRAM to SDR data. As the DDR2 data is converted to SDR data, the width of this bus is 2n, where n is data width of the DDR2 SDRAM data bus.</td></tr>
<tr><td>user_data_valid</td><td>Output</td><td>When asserted, this signal indicates user_output_data[(2n−1):0] is valid.</td></tr>
<tr><td>user_cmd_ack</td><td>Output</td><td>This is the acknowledgement signal for a user read or write command. It is asserted by the DDR2 SDRAM controller during a write or read to/from the DDR2 SDRAM. The user should not issue any new commands to the controller until this signal is deasserted.</td></tr>
<tr><td>init_val</td><td>Output</td><td>The DDR2 SDRAM controller asserts this signal to indicate that the DDR2 SDRAM initialization is complete.</td></tr>
<tr><td>ar_done</td><td>Output</td><td>This indicates that the auto-refresh command was given to DDR2 SDRAM. The DDR2 SDRAM controller asserts this signal for one clock after giving an auto-refresh command to the DDR2 SDRAM and completion of T_{RFC} time. The T_{RFC} time is determined by the rfc_count_value value in the parameter file. The user should change this value appropriately for the memory used.</td></tr>
</table>

Table 8-3: **DDR2 SDRAM Controller User Interface Signals *(Continued)***

Signal Names	Direction(1)	Description
auto_ref_req	Output	This signal is asserted on every 7.7 µs. It is asserted until the controller issues an auto_refresh command to the memory. Upon seeing this signal, the user should terminate any ongoing command after the current burst transaction by asserting the burst_done signal. The frequency with which this signal is asserted is determined by max_ref_cnt value in parameter file.

Notes:

1. All of the signal directions are with respect to the DDR2 SDRAM controller.
2. col_ap_width includes an autoprecharge (A10) bit 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 for autoprecharge. The col_ap_width parameter is used internally for prepending the A10 bit during the precharge command. For a column address of 10, col_ap_width is defined as 11. For an 11-bit column address, col_ap_width is defined as 12. The user need consider only the column address parameter for driving the column address to the memory.

Resource Utilization

A local inversion clocking technique is used in this design. The DCM generates only clk0 and clk90. One DCM and two BUFGs are used. The Spartan-3 designs operate at 166 MHz and below.

DDR2 SDRAM Initialization

Before issuing the memory read and write commands, the controller initializes the DDR2 SDRAM using the memory initialization command. The user can give the initialization command only after all reset signals are deactivated. The controller is in the reset state for 200 µs after power up. For design optimization, a 200 µs timer is generated from the refresh counter. The refresh timer is a function of frequency. Therefore, at lower frequencies, the 200 µs timer waits more than 200 µs. Because wait200 happens only during the power-up sequence, design performance is not degraded. All resets are asserted for 200 µs because DDR2 SDRAM requires a 200 µs delay prior to applying an executable command after all power supply and reference voltages are stable. The controller asserts clock-enable to memory after 200 µs.

The user places the data to be written into the Load Mode Register on user_config_register[14:0] until the DDR2 SDRAM initialization is complete. The controller then writes this data into the Load Mode Register. Once the DDR2 SDRAM is initialized, the DDR2 SDRAM controller asserts the init_val signal.

Figure 8-4 shows the timing for the memory initialization command.

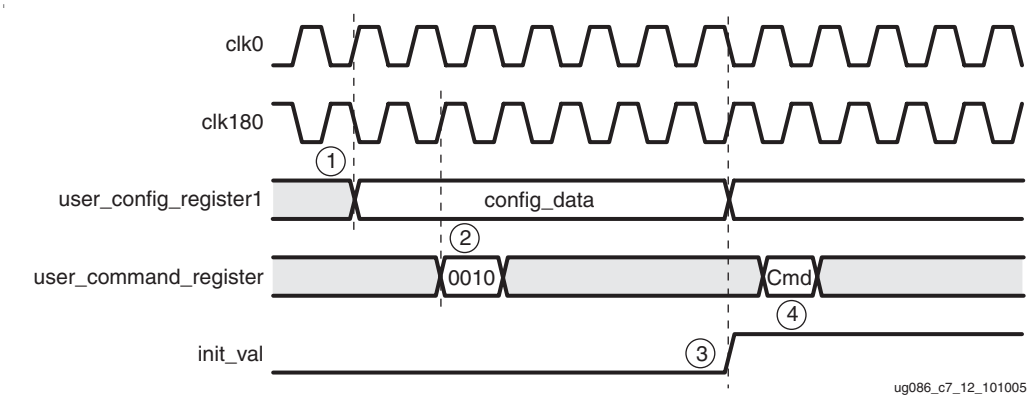


Figure 8-4: **DDR2 SDRAM Initialization**

1. The user places valid configuration data on user_config_register[14:0] two clocks prior to placing the initialization command on user_command_register[3:0].
2. The user places the initialization command on user_command_register[3:0] on a rising edge of clk180 for one clock cycle. This starts the initialization sequence.
3. The DDR2 SDRAM controller indicates that the configuration is complete by asserting the init_val signal on a rising edge of clk180. The init_val signal is asserted throughout the period.
4. After init_val is asserted, the user can pass the next command at any time.

DDR2 SDRAM Write

Figure 8-5 shows the timing diagram for a write to DDR2 SDRAM for a burst length of four. The user initiates the write command by sending a Write instruction to the DDR2 SDRAM controller. To terminate a write burst, the user asserts the burst_done signal for two clocks after the last user_input_address. The burst_done signal should be asserted for two clocks for burst lengths of four and four clocks for burst lengths of eight.

The write command is asserted on the rising edge of clk180. In response to a write command, the DDR2 SDRAM controller acknowledges with the usr_cmd_ack signal on a rising edge of clk180. If the controller is busy with a refresh, the usr_cmd_ack signal is not asserted until after the refresh command cycle completes. The user asserts the first address (row + column address) with the write command and keeps it asserted for three clocks after usr_cmd_ack assertion. Any subsequent write addresses are asserted on an alternate positive edge of clk180 after deasserting the first memory address for a burst length of four, and it is asserted once in four clocks for a burst length of eight. The first user data is asserted on a rising edge of clk90 after usr_cmd_ack is asserted. As the SDR data is converted to DDR2 data, the width of this bus is 2*n*, where *n* is data width of DDR2 SDRAM data bus.

For a burst length of four, only two data words (each of 2*n*) are given to the DDR2 SDRAM controller for each user address, and four data words are given for a burst length of eight.. Internally, the DDR2 SDRAM controller converts into four data words for a burst length of four and eight data words for a burst length of eight, each of *n* bits. To terminate the write burst, the user asserts burst_done on a rising edge of clk180 for two clocks for a burst length of four and four clocks for a burst length of eight. The burst_done signal is asserted after the last memory address. Any further commands to the DDR2 SDRAM controller are given only after the usr_cmd_ack signal is deasserted. After burst_done is asserted, the controller terminates the burst and issues a precharge to the memory. The usr_cmd_ack signal is deasserted after completion of the precharge.

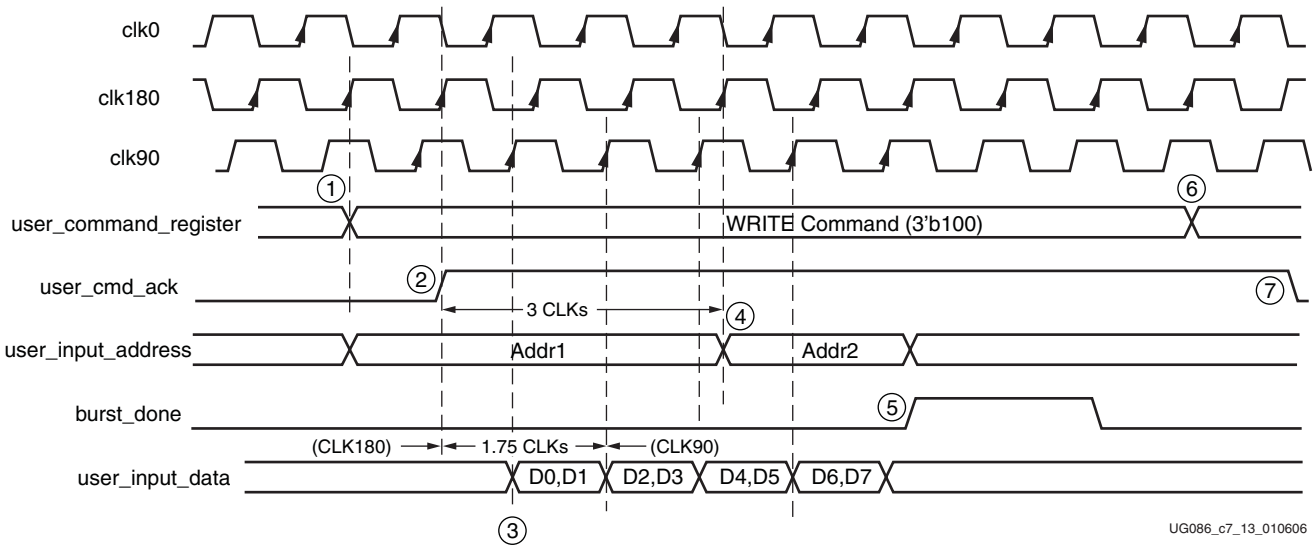


Figure 8-5: **DDR2 SDRAM Write Burst, Burst Lengths of Four and Two Bursts**

1. A memory write is initiated by issuing a write command to the DDR2 SDRAM controller. The write command must be asserted on a rising edge of clk180.
2. The DDR2 SDRAM controller acknowledges the write command by asserting the user_cmd_ack signal on a rising edge of clk180.
3. The first user_input_address must be placed along with the command. The input data is asserted with the clk90 signal after the user_cmd_ack signal is asserted.
4. The user asserts the first address (row + column address) with the write command and keeps it asserted for three clocks after usr_cmd_ack assertion. The user_input_address signal is asserted on a rising edge of clk180. All subsequent addresses are asserted on alternate positive edges of clk180.
5. To terminate the write burst, burst_done is asserted after the last user_input_address. The burst_done signal is asserted for two clock cycles.
6. The user command is deasserted after burst_done is asserted.
7. The controller deasserts the user_cmd_ack signal after completion of precharge to the memory. The next command must be given only after user_cmd_ack is deasserted. Back-to-back write operations are supported only within the same bank and row.

DDR2 SDRAM Read

The user initiates a memory read with a read command to the DDR2 SDRAM controller. Figure 8-6 shows the memory read timing diagram for a burst length of four.

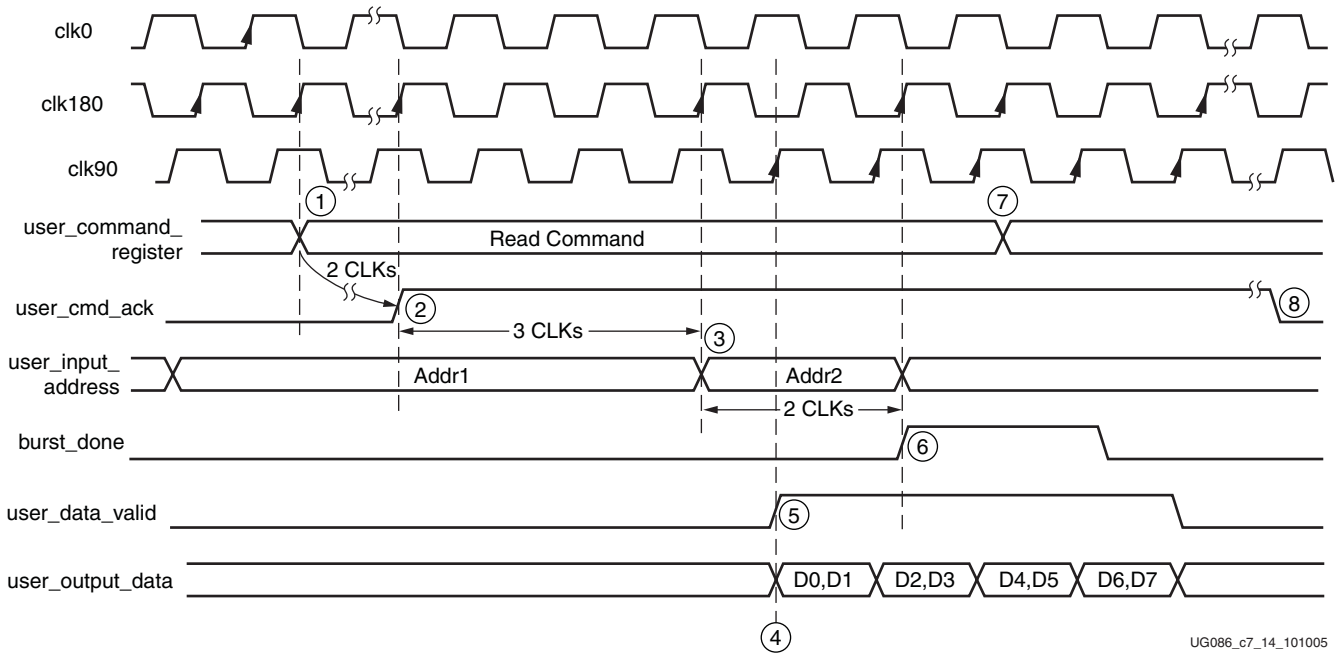


Figure 8-6: **DDR2 SDRAM Read, Burst Lengths of Four and Two Bursts**

The user provides the first memory address with the read command, and subsequent memory addresses upon receiving the usr_cmd_ack signal. Data is available on the user data bus with the user_data_valid signal. To terminate read burst, the user asserts the burst_done signal on a rising edge of clk180 for two clocks with the deassertion of the last user_input_address. All subsequent addresses are asserted on alternate clocks for burst lengths of four, and subsequent addresses are asserted once every four clock cycles for burst lengths of eight.

For burst lengths of four, the burst_done signal is asserted for two clocks after the last address and for four clocks for burst lengths of eight.

The read command flow is similar to the write command flow.

1. A memory read is initiated by issuing a read command to the DDR2 SDRAM controller. The read command is accepted on a rising edge of the clk180.
2. The first read address must be placed along with the read command. In response to the read command, the DDR2 SDRAM controller asserts the user_cmd_ack signal on a rising edge of clk180.
3. The user asserts the first address (row + column address) with the read command and keeps it asserted for three clocks after usr_cmd_ack is asserted. The user_input_address signal is then accepted on the rising edge of clk180. All subsequent memory read addresses are asserted on alternate positive edges of clk180.
4. The data on user_output_data is valid only when the user_data_valid signal is asserted.
5. The data read from the DDR2 SDRAM is available on user_output_data, which is asserted with clk90. Because the DDR2 SDRAM data is converted to SDR data, the width of this bus is 2*n*, where *n* is the data width of the DDR2 SDRAM memories. For a read burst length of four, the DDR2 SDRAM controller outputs only two data words with each user address.

6. To terminate the read burst, burst_done is asserted for two clocks on the rising edge of clk180. The burst_done signal is asserted after the last memory address.
7. The user command is deasserted after burst_done is asserted.
8. The controller deasserts the user_cmd_ack signal after completion of precharge to the memory. Any further commands to the DDR2 SDRAM controller should be given after user_cmd_ack is deasserted. Back-to-back read operations are supported only within the same bank and row. Approximately 17 clock cycles pass between the time a read command is asserted on the user interface and the time data becomes available on the user interface.

Auto_Refresh DDR2 SDRAM

The DDR2 SDRAM controller does a memory refresh periodically. Every 7.7 µs, the controller raises an auto_refresh request. The user must terminate any ongoing commands when auto_ref_req flag is asserted after the current burst transaction by asserting the burst_done signal. The auto_ref_req flag is asserted until the controller issues a refresh command to the memory. The user must wait for completion of the auto_refresh command before giving any commands to the controller when auto_ref_req is asserted.

The controller sets the max_ref_cnt value in the parameter file according to the frequency and selected memory component for a refresh interval (7.7 µs). The rfc_count_value setting in the parameter file defines T_{RFC}, the time between the refresh command to Active or another refresh command.

The ar_done signal is asserted by the DDR2 SDRAM controller upon completion of the auto_refresh command—i.e., after T_{RFC} time. The ar_done signal is asserted with clk180 for one clock cycle.

After completion of the auto_refresh command, the next command can be given any time after ar_done is asserted.

DDR2 SDRAM Load Mode

The user gives the LOAD MODE command for one clock on the rising edge of clk180. No acknowledgement is given for this command. The user waits for T_{MRD} time for the LOAD MODE command to complete before giving the next command.

Placement of configuration data is similar to that of the initialization command. Configuration data must be placed on user_config_register1 and user_config_register2 four clocks prior to giving the LOAD MODE command to the controller.

Constraints

Some constraints are required to successfully create the design. The following examples explain the different constraints in the UCF file.

Calibration Circuit Constraints

All LUTs in the matched delay circuits are constrained to specific locations in the device.

Example:

```
INST "infrastructure_top0/cal_top0/tap_dly0/l0" RLOC_ORIGIN = X50Y104;
INST "infrastructure_top0/cal_top0/tap_dly0/l0" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/l0" U_SET = "cal_top0/tap_dly0/l0";
INST "infrastructure_top0/cal_top0/tap_dly0/r0" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/r0" U_SET = "cal_top0/tap_dly0/l0";
```

```
INST "infrastructure_top0/cal_top0/tap_dly0/r1" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/r1" U_SET = "cal_top0/tap_dly0/l0";
```

Data and Data Strobe Constraints

Data and data strobe signals are assigned to specific pins in the device; placement constraints related to the dqs_delay circuit and the FIFOs used for the data_read module are specified.

Example:

```
NET "cntrl0_DDR2_DQS[0]" LOC = Y6;
INST "ddr2_top0/data_path0/data_read_controller0/dqs_delay0_col0/one" LOC =
SLICE_X0Y110;
INST "ddr2_top0/data_path0/data_read_controller0/dqs_delay0_col0/one" BEL = F;
NET "cntrl0_DDR2_DQS[0]" LOC = Y5;
INST "ddr2_top0/data_path0/strobe0/fifo0_bit0" LOC = SLICE_X2Y111;
```

I/O Clocking Rules

There are three I/O clocking rules to be followed for I/O pin allocations. Additional information regarding I/O clocking rules is located in DS099 [Ref 17].

I/O Banking Rules

There are I/O banking rules to be followed for I/O pin allocations, stating that the I/O signals allocated in a bank should adhere to compatible I/O standards. Additional information regarding I/O banking rules is located in DS099 [Ref 17].

Design Notes

The DDR2 SDRAM design is not validated on hardware. The maximum delay constraints in the UCF file are set to the selected frequency from the tool.

Appendix A

Memory Implementation Guidelines

This appendix provides rules for designing reference design boards generated by the MIG tool. It is organized into two sections:

- "Generic Memory Interface Guidelines"

  The rules in this section apply to all memory interfaces discussed in this document.

- "Memory-Specific Guidelines"

  The rules in this section relate to specific memories:

  - DDR/DDR2 SDRAM
  - QDRII SRAM
  - RLDRAM II

UG079 [Ref 8] provides more detailed analysis. UG072 [Ref 7] provides additional information on how to obtain maximum performance for high-speed interfaces.

Generic Memory Interface Guidelines

This section specifies rules common to all memory interfaces. The "Memory-Specific Guidelines" section provides exceptions or additions to any and all guidelines in this section.

Figure A-1 illustrates a typical FPGA bank used to capture read data.

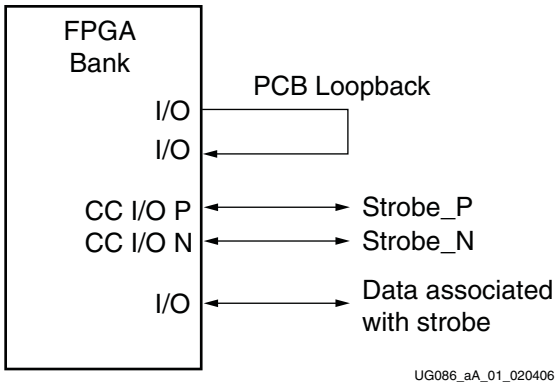


Figure A-1: **FPGA Bank with Data, Strobes, and PCB Loopback**

Pin Assignments

These rules apply to pin assignments:

1. All strobe signals must be placed on clock-capable inputs (such as DQS, CQ, QK) with P connected to the P side and N connected to the N side of the pair. If only single-ended strobes are provided, the signal is placed on the P-input of the clock-capable I/O pair.
2. Data lines used to read data from a memory are placed in the same bank as its associated strobe. The strobe is used to capture the associated data in the FPGA.
3. For memory interfaces that do not provide a *data valid* signal to indicate when the read data is valid, a data valid signal is to be provided on the PCB board. A loopback signal on the PCB is needed for each bank used to strobe in read data. Two pins per bank are needed: one to output the signal and one to input the return signal. The length of the loopback is:

   PCB loopback = CLK delay to memory + strobe delay
4. Address and control signals are to be placed together in the same bank (see "Memory-Specific Guidelines" for exceptions), to minimize the route delays for these signals inside the FPGA.

Termination

These rules apply to termination:

1. IBIS simulation is highly recommended for all high-speed interfaces
2. Single-ended signals are to be terminated with a pull-up of 50Ω to V_{TT} at the load (see Figure A-2). A split 100Ω termination to V_{CCO} and 100Ω termination to GND can be used (see Figure A-3), but takes more power. For bidirectional signals, the termination is needed at both ends of the signal (DCI/ODT or external termination).

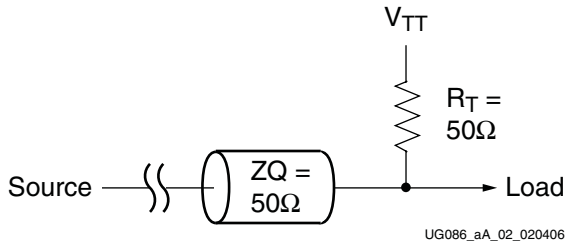


Figure A-2: **50Ω Termination to V_{TT}**

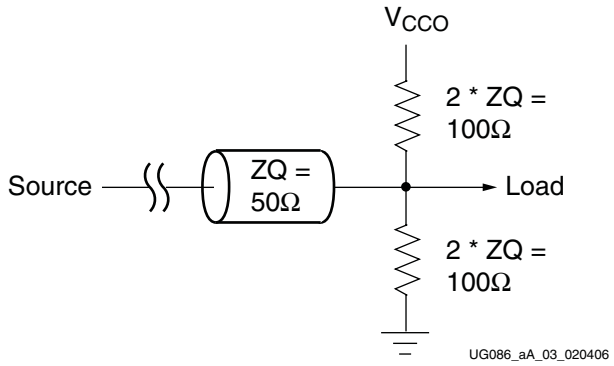


Figure A-3: **100Ω Split Termination to V_{TT} and GND**

3. Differential signals are to be terminated with a 100Ω differential termination at the load (see Figure A-4). For bidirectional signals, termination is needed at both ends of the signal (DCI/ODT or external termination).

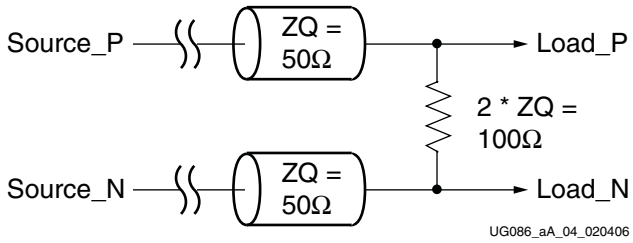


Figure A-4: **100Ω Differential Termination**

4. All termination must be placed as close to the load as possible. The termination can be placed before or after the load provided that the termination is placed within one inch of the load pin.
5. DCI can be used at the FPGA as long as the DCI rules are followed (such as VRN/VRP).

Memory-Specific Guidelines

Each memory interface has three sections:

- Pin assignments
- Termination
- Trace lengths

Trace lengths given are for high-speed operation and can be relaxed depending on the applications target bandwidth requirements.

DDR/DDR2 SDRAM

Pin Assignments

These rules apply to pin assignments for DDR and DDR2 SDRAM:

1. The DQ and DM bits of a byte are to be placed in the same bank as the associated DQS.

   The DQ bits must be kept close together for better routing.
2. Address and control signals are to be placed in the same bank.

   CK, ODT, and CKE are exceptions to this rule and can be placed in another bank.
3. Each bank that contains DQ/DQS/DM signals needs a loopback signal.

If a bank is pin-limited and there is a need to free up a few pins, the following actions are to be considered:

1. The loopback signals can be eliminated in Virtex-4 MIG 1.5 designs because they are no longer required. Other device families require significant user modifications to the MIG design to eliminate the PCB loopback.
2. The CKE signals can be tied together for multiple devices.
3. For DIMMs, non-critical features need not be implemented, such as PAR_IN/PAR_OUT and the SPD interface (SA, SPD, SCL).

Termination

These rules apply to termination for DDR/DDR2 SDRAM:

1. For DIMMs, the CK signals are to be terminated by a 5 pF capacitor between the two legs of the differential signal instead of the 100Ω resistor termination, because these signals are already terminated on each DIMM.

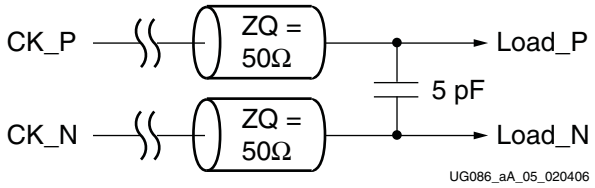


Figure A-5: **5 pF Differential Termination on Clocks**

2. The ODT and CKE signals are not terminated. These signals are required to be pulled down during memory initialization with a 4.7 kΩ resistor connected to GND.
3. ODT, which terminates a signal at the memory, applies to the DQ/DQS/DM signals only. If ODT is used, the Mode register must be set appropriately in the RTL design.

To save board space, DCI at the FPGA and ODT at the memory can be used to minimize the number of external resistors on the board.

Trace Lengths

These rules indicate the maximum electrical delays between DDR/DDR2 SDRAM signals:

1. ±25 ps maximum electrical delay between data and its associated DQS.
2. ±50 ps maximum electrical delay between address and control signals.
3. ± 100 ps maximum electrical delay between address/control and data.

QDRII SRAM

Pin Assignments

These rules apply to pin assignments for QDRII SRAM:

1. All CQ signals are placed on clock-capable I/O pairs, with P connected to the P side and N to the N side of the pair.
2. The Q bits of a byte are placed in the same bank as its associated CQ.

   The Q bits must be kept close together for optimal routing.
3. Each bank that contains Q/CQ signals needs a loopback signal, where two pins per bank are allocated for the loopback.

If a bank is pin-limited and there is a need to free up a few pins, the following actions are to be considered:

1. The loopback signals can be eliminated. However, the MIG tool does not support QDRII without PCB loopback. The MIG design must be modified by the user for the specific configuration desired.
2. If QDRII+ memory is to be considered, either CK_P is connected or CK_P and CK_N are left out.

Termination

These rules apply to termination of QDRII SRAM signals:

1. No termination is used for the DLL_OFF signal because this signal is required to be pulled down during memory initialization. The signal should be pulled down with a 4.7 kΩ resistor connected to GND.
2. DCI can also be used on CK for QDRII+ support (QVLD signal from memory to FPGA).

To save board space, DCI is to be used at the FPGA to minimize the number of external resistors on the board.

Trace Lengths

These rules provide the maximum electrical delays between QDRII SRAM signals:

1. ±25 ps maximum electrical delay between data and its associated CQ.
2. ±50 ps maximum electrical delay between address and control signals.
3. ±100 ps maximum electrical delay between address/control and data.

RLDRAM II

Pin Assignments

These rules apply to pin assignments for RLDRAM II:

1. All QK signals are to be placed on Clock-Capable I/O pairs, with P connected to the P side and N connected to the N side of the pair.
2. The DQ bits of a byte are placed in the same bank as the associated DQS.

   The DQ bits must be kept as close as possible for optimal routing.
3. The loopback signal is not required because RLDRAM II provides a data valid signal for capturing the read data.

If the design is pin constrained, only common I/O (CIO) can use a bidirectional DQ data bus.

Termination

This rule applies to termination of RLDRAM II signals:

1. DCI can be used on DQ/QK at the FPGA provided that DCI rules are followed (such as VRN/VRP).

To save board space, use DCI at the FPGA and ODT at the memory to minimize the number of external resistors on the board.

Trace Lengths

These rules provide the maximum electrical delays between RLDRAM II signals:

1. ±25 ps maximum electrical delay between data and its associated QK.
2. ±50 ps maximum electrical delay between address and control signals.
3. ±100 ps maximum electrical delay between address/control and data.