

# **Xilinx Memory Interface Generator (MIG) User Guide**

***DDR SDRAM, DDRII SRAM,  
DDR2 SDRAM, QDRII SRAM,  
and RLDRAM II Interfaces***

UG086 (v2.2) March 3, 2008





Xilinx is disclosing this Document and Intellectual Property (hereinafter “the Design”) to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED “AS IS” WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

© 2004-2008 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM Corp. and is licensed for use. All other trademarks are the property of their respective owners.

## 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.
03/28/06	1.5.1	Updated <a href="#">Table 3-7</a> and added <a href="#">Table 3-8</a> .
07/28/06	1.6	MIG 1.6 release.
03/21/07	1.7	MIG 1.7 release.

---

Date	Version	Revision
04/30/07	1.7.2	MIG 1.72 release. Added support for Spartan-3AN FPGAs.
07/05/07	1.7.3	MIG 1.73 release. Added support for Spartan-3A DSP FPGAs. Corrected minor typographical errors.
09/18/07	2.0	MAJOR REVISION. MIG Wizard guide added to Chapter 1. Design Frequency Range and Hardware Tested Configuration tables added in most chapters. Diagram and table updates throughout.
01/09/08	2.1	MIG 2.1 release. Revisions and added material throughout, including new Chapter 12, Appendix B, and Appendix D.
03/03/08	2.2	MIG 2.2 release. Added Qimonda support. Updated screen captures in Chapter 1. Added Spartan-3E FPGA support to Chapters 7 and 8. Added footnote to <a href="#">Table 9-1, page 300</a> . Added <a href="#">“Timing Analysis” in Appendix A</a> . Added information on loading of address, command, and control signals to <a href="#">“Pin Assignments” in Appendix A</a> . Replaced Appendix D.





# Table of Contents

---

Revision History .....	2
------------------------	---

## **Preface: About This Guide**

Guide Contents .....	17
References .....	18
Additional Resources .....	19
Typographical Conventions .....	19
Type Case of Port and Signal Names .....	20

## **SECTION I: INTRODUCTION**

### **Chapter 1: Using MIG**

MIG 2.2 Changes from MIG 2.1 .....	23
MIG 2.1 Changes from MIG 2.0 .....	24
MIG 2.0 Changes from MIG 1.73 .....	25
MIG 1.73 Changes from MIG 1.72 .....	25
MIG 1.72 Changes from MIG 1.7 .....	26
MIG 1.7 Changes from MIG 1.6 .....	26
MIG 1.6 Changes from MIG 1.5 .....	26
MIG 1.5 Changes from MIG 1.4 .....	27
Tool Features .....	29
Design Tools .....	31
Installation .....	31
Getting Started .....	31
MIG User Interface .....	32
Getting Help .....	32
Version Information .....	32
CORE Generator Options .....	33
MIG Output Options .....	33
Create Design .....	35
Output Files .....	59
Create Design for Xilinx Reference Boards .....	60
Verify UCF/Update Design .....	64
Create Preset Configuration .....	72
Spartan-3A FPGA DDR2 SDRAM 200 MHz Design .....	73
Using MIG in Batch Mode .....	74
XCO File .....	74
MIG.prj File .....	74
Running in Batch Mode .....	77

## SECTION II: VIRTEX-4 FPGA TO MEMORY INTERFACES

### Chapter 2: Implementing DDR SDRAM Controllers

<b>Feature Summary</b> .....	81
Supported Features .....	81
Design Frequency Ranges .....	81
Unsupported Features .....	82
<b>Architecture</b> .....	82
Interface Model .....	82
Implemented Features .....	82
Burst Length .....	83
CAS Latency .....	83
Registered DIMMs .....	83
Unbuffered DIMMs and SODIMMs .....	84
Precharge .....	84
Auto Refresh .....	84
Linear Addressing .....	84
Different Memories (Density/Speed) .....	84
Hierarchy .....	86
Controller .....	92
Datapath .....	92
User Interface .....	92
Infrastructure .....	92
IOBS Module .....	92
<b>DDR SDRAM Initialization and Calibration</b> .....	92
<b>DDR SDRAM System and User Interface Signals</b> .....	94
User Interface Accesses .....	96
Write Interface .....	96
Correlation between the Address and Data FIFOs .....	98
Read Interface .....	99
<b>Simulating the DDR SDRAM Design</b> .....	101
Changing the Refresh Rate .....	101
<b>Supported Devices</b> .....	102
<b>Hardware Tested Configurations</b> .....	103

### Chapter 3: Implementing DDR2 SDRAM Controllers

<b>Interface Model</b> .....	105
<b>Direct Clocking Interface</b> .....	106
Feature Summary .....	106
Supported Features .....	106
Design Frequency Ranges .....	106
Unsupported Features .....	107
Architecture .....	107
Implemented Features .....	107
Hierarchy .....	111
DDR2 Controller Submodules .....	117
DDR2 SDRAM Initialization and Calibration .....	118
DDR2 SDRAM System and User Interface Signals .....	118
User Interface Accesses .....	120
Write Interface .....	121

Correlation between the Address and Data FIFOs . . . . .	123
Read Interface . . . . .	124
User to Controller Interface . . . . .	126
Dynamic Command Request . . . . .	127
Controller to Physical Layer Interface. . . . .	128
Deep Memory Configurations . . . . .	130
Components . . . . .	130
DIMMs . . . . .	133
Simulating the DDR2 SDRAM Design . . . . .	136
Changing the Refresh Rate . . . . .	136
Supported Devices . . . . .	136
Hardware Tested Configurations . . . . .	138
<b>SerDes Clocking Interface</b> . . . . .	139
Feature Summary . . . . .	139
Supported Features . . . . .	139
Design Frequency Ranges . . . . .	140
Unsupported Features . . . . .	140
Architecture . . . . .	140
Implemented Features . . . . .	140
Hierarchy . . . . .	144
DDR2 Controller Submodules . . . . .	147
DDR2 SDRAM Initialization and Calibration . . . . .	149
DDR2 SDRAM System and User Interface Signals . . . . .	150
User Interface Accesses . . . . .	151
Write Interface . . . . .	152
Correlation between the Address and Data FIFOs . . . . .	154
Read Interface . . . . .	155
User to Controller Interface . . . . .	158
Dynamic Command Request . . . . .	160
Controller to Physical Layer Interface. . . . .	161
Simulating the DDR2 SDRAM Design . . . . .	162
Changing the Refresh Rate . . . . .	162
Supported Devices . . . . .	163
Hardware Tested Configurations . . . . .	165

## Chapter 4: Implementing QDRII SRAM Controllers

<b>Feature Summary</b> . . . . .	167
Design Frequency Range . . . . .	167
Limitations . . . . .	167
<b>Architecture</b> . . . . .	168
Interface Model . . . . .	168
Hierarchy . . . . .	169
QDRII Memory Controller Modules . . . . .	175
Controller . . . . .	176
Datapath . . . . .	177
Infrastructure . . . . .	178
IOBS . . . . .	178
<b>QDRII SRAM Initialization and Calibration</b> . . . . .	179
<b>QDRII Controller System and User Interface Signals</b> . . . . .	180
Write Interface . . . . .	183
Read Interface . . . . .	186
Supported Devices . . . . .	189

Simulating the QDRII SRAM Design .....	190
Hardware Tested Configurations .....	191

## Chapter 5: Implementing DDRII SRAM Controllers

<b>Feature Summary</b> .....	193
Supported Features .....	193
Design Frequency Range .....	193
Unsupported Features .....	193
<b>Architecture</b> .....	194
Interface Model .....	194
Hierarchy .....	195
DDRII SRAM Controller Modules .....	201
Controller .....	202
Datapath .....	202
Infrastructure .....	202
IOBS .....	202
<b>DDRII SRAM Initialization and Calibration</b> .....	203
<b>User Interface</b> .....	204
<b>DDRII SRAM Controller Interface Signals</b> .....	204
Write Interface .....	207
Read Interface .....	210
Supported Devices .....	213
<b>Simulating the DDRII SRAM Design</b> .....	214
<b>Hardware Tested Configurations</b> .....	214

## Chapter 6: Implementing RLDRAM II Controllers

<b>Feature Summary</b> .....	215
Supported Features .....	215
Design Frequency Range .....	216
Unsupported Features .....	216
Supported RLDRAM II Devices .....	216
<b>Architecture</b> .....	217
Implemented Features .....	223
Address Multiplexing .....	223
CIO/SIO .....	223
Data Capture Using the Direct Clocking Technique .....	224
Memory Initialization .....	224
Block Diagram Description .....	225
User Interface .....	225
Address FIFO .....	225
Write Data FIFO .....	226
Read Data FIFO .....	226
Configuration Registers .....	226
Clock Generator .....	227
Reset Generator .....	227
Control Logic .....	227
RLDRAM II Control Signal Physical Layer .....	228
<b>RLDRAM II Interface Signals</b> .....	228
<b>User Command Interface</b> .....	230

User Interface Accesses . . . . .	230
Write Interface . . . . .	231
Read Interface . . . . .	233
Refresh Commands . . . . .	235
<b>Simulating the RLDRAM II Design . . . . .</b>	<b>236</b>
<b>Hardware Tested Configurations . . . . .</b>	<b>237</b>

## SECTION III: SPARTAN-3/3E/3A/3AN/3A DSP FPGA TO MEMORY INTERFACES

### Chapter 7: Implementing DDR SDRAM Controllers

<b>Feature Summary . . . . .</b>	<b>241</b>
Design Frequency Ranges . . . . .	242
<b>Controller Architecture . . . . .</b>	<b>242</b>
DDR SDRAM Interface . . . . .	242
Hierarchy . . . . .	243
Controller . . . . .	248
Datapath . . . . .	248
Data Read Controller . . . . .	248
Data Read . . . . .	248
Data Write . . . . .	248
Infrastructure_top . . . . .	249
IOBs . . . . .	249
<b>Interface Signals . . . . .</b>	<b>249</b>
<b>Resource Utilization . . . . .</b>	<b>253</b>
DDR SDRAM Initialization . . . . .	253
DDR SDRAM Write and Read Operations . . . . .	253
Write . . . . .	254
Read . . . . .	255
Auto Refresh . . . . .	256
Changing the Refresh Rate . . . . .	257
Load Mode . . . . .	257
UCF Constraints . . . . .	257
Calibration Circuit Constraints . . . . .	257
Data and Data Strobe Constraints . . . . .	257
MAXDELAY Constraints . . . . .	258
<b>I/O Banking Rules . . . . .</b>	<b>259</b>
<b>Design Notes . . . . .</b>	<b>259</b>
Spartan-3/3E/3A/3AN/3A DSP Pin Allocation Rules . . . . .	259
Pin Allocation Rules for Left/Right Banks . . . . .	259
Pin Allocation Rules for Top/Bottom Banks . . . . .	259
<b>Supported Devices . . . . .</b>	<b>260</b>
<b>Simulating the Spartan-3/3E/3A/3AN/3A DSP FPGA Design . . . . .</b>	<b>263</b>
<b>Hardware Tested Configurations . . . . .</b>	<b>264</b>

### Chapter 8: Implementing DDR2 SDRAM Controllers

<b>Feature Summary . . . . .</b>	<b>265</b>
----------------------------------	------------

Design Frequency Ranges .....	266
<b>Controller Architecture</b> .....	266
DDR2 SDRAM Interface .....	266
Hierarchy .....	267
Controller .....	272
Datapath .....	272
Data Read Controller .....	272
Data Read .....	272
Data Write .....	272
Infrastructure_top .....	273
IOBs .....	273
<b>Interface Signals</b> .....	273
<b>Resource Utilization</b> .....	277
DDR2 SDRAM Initialization .....	277
Write .....	278
Read .....	279
Auto Refresh .....	280
Changing the Refresh Rate .....	280
Load Mode .....	281
UCF Constraints .....	281
Calibration Circuit Constraints .....	281
Data and Data Strobe Constraints .....	281
MAXDELAY Constraints .....	281
<b>I/O Banking Rules</b> .....	283
<b>Design Notes</b> .....	283
<b>Tool Output</b> .....	283
<b>Supported Devices</b> .....	283
Maximum Data Widths .....	288
DIMM Support for Spartan-3 Generation Devices .....	293
Design Frequency Range in MHz for Spartan-3 Generation Devices .....	294
<b>Hardware Tested Configurations</b> .....	295

## SECTION IV: VIRTEX-5 FPGA TO MEMORY INTERFACES

### Chapter 9: Implementing DDR2 SDRAM Controllers

Interface Model .....	299
<b>Feature Summary</b> .....	300
Supported Features .....	300
Design Frequency Ranges .....	300
Unsupported Features .....	300
<b>Architecture</b> .....	301
Implemented Features .....	301
Burst Length .....	301
CAS Latency .....	301
Additive Latency .....	301
Data Masking .....	301
Precharge .....	302
Auto Refresh .....	302

Bank Management . . . . .	302
Linear Addressing . . . . .	302
Different Memories (Density/Speed) . . . . .	302
On-Die Termination . . . . .	302
Generic Parameters . . . . .	303
Hierarchy . . . . .	306
Constraints . . . . .	307
MIG Tool Design Options . . . . .	308
DDR2 Controller Submodules . . . . .	312
Infrastructure . . . . .	312
ldelay_ctrl . . . . .	313
Ctrl . . . . .	313
phy_top . . . . .	313
usr_top . . . . .	314
<b>DDR2 SDRAM Initialization</b> . . . . .	314
<b>DDR2 SDRAM Design Calibration</b> . . . . .	314
<b>DDR2 SDRAM System and User Interface Signals</b> . . . . .	315
User Interface Accesses . . . . .	317
Write Interface . . . . .	317
Read Interface . . . . .	320
Simulating the DDR2 SDRAM Design . . . . .	322
Supported Devices . . . . .	322
<b>Hardware Tested Configurations</b> . . . . .	324

## Chapter 10: Implementing QDRII SRAM Controllers

<b>Feature Summary</b> . . . . .	325
Supported Features . . . . .	325
Design Frequency Ranges . . . . .	325
Unsupported Features . . . . .	325
<b>Architecture</b> . . . . .	326
Interface Model . . . . .	326
Hierarchy . . . . .	327
QDRII Memory Controller Modules . . . . .	333
Controller . . . . .	333
Infrastructure . . . . .	335
top_phy . . . . .	335
DCI Cascading . . . . .	335
CQ/CQ_n Implementation . . . . .	337
Pinout Considerations . . . . .	337
User Interface . . . . .	337
<b>QDRII SRAM Initialization and Calibration</b> . . . . .	338
<b>QDRII Controller Interface Signals</b> . . . . .	338
User Interface Accesses . . . . .	341
Write Interface . . . . .	342
Read Interface . . . . .	344
Supported Devices . . . . .	347
<b>Simulating the QDRII SRAM Design</b> . . . . .	348
<b>Hardware Tested Configurations</b> . . . . .	348

## Chapter 11: Implementing DDR SDRAM Controllers

<b>Interface Model</b> .....	349
<b>Feature Summary</b> .....	350
Supported Features .....	350
Design Frequency Ranges .....	350
Unsupported Features .....	350
<b>Architecture</b> .....	350
Implemented Features .....	350
Burst Length .....	352
CAS Latency .....	352
Precharge .....	352
Data Masking .....	353
Auto Refresh .....	353
Bank Management .....	353
Linear Addressing .....	353
Different Memories (Density/Speed) .....	353
<b>Hierarchy</b> .....	353
<b>MIG Design Options</b> .....	355
Infrastructure .....	359
idelay_ctrl .....	360
ctrl .....	360
phy_top .....	360
usr_top .....	361
System Interface Signals .....	362
<b>DDR SDRAM Initialization</b> .....	363
<b>DDR SDRAM Design Calibration</b> .....	364
<b>User Interface Accesses</b> .....	365
<b>Write Interface</b> .....	366
<b>Read Interface</b> .....	369
<b>Supported Devices</b> .....	372
<b>Simulating a DDR SDRAM Design</b> .....	373
<b>Hardware Tested Configurations</b> .....	373

## SECTION V: DDR2 DEBUG GUIDE

### Chapter 12: Debugging MIG DDR2 Designs

<b>Introduction</b> .....	377
<b>Verifying Board Layout</b> .....	378
Introduction .....	378
Memory Implementation Guidelines .....	378
Calculate WASSO .....	378
Run SI Simulation Using IBIS .....	379
<b>Verifying Design Implementation</b> .....	379
Introduction .....	379
Behavioral Simulation .....	379
Verify Modifications to MIG Output .....	380



Changing the Pinout Provided in the Output UCF . . . . .	380
Changing Design Parameters . . . . .	380
Migrating MIG Output into ISE Project . . . . .	381
Verify Successful Placement and Routing . . . . .	381
Verify IDELAYCTRL Instantiation for Virtex-4 and Virtex-5 FPGA Designs . . . . .	381
Verify TRACE Timing . . . . .	382
<b>Debugging the Spartan-3 FPGA Design . . . . .</b>	<b>382</b>
Introduction . . . . .	382
Read Data Capture . . . . .	382
Verify Placement and Routing . . . . .	383
DQ Routing . . . . .	383
DQS Routing . . . . .	385
Debugging Physical Layer in Hardware . . . . .	386
Loopback Timing . . . . .	387
Incorrect DQS Delay . . . . .	387
Proceed to General Board-Level Debug . . . . .	387
<b>Debugging the Virtex-4 FPGA Direct Clocking Design . . . . .</b>	<b>388</b>
Introduction . . . . .	388
Read Data Capture Timing Calibration . . . . .	388
Signals of Interest . . . . .	389
Proceed to General Board-Level Debug . . . . .	390
<b>Debugging the Virtex-4 FPGA SerDes Design . . . . .</b>	<b>390</b>
Introduction . . . . .	390
Read Data Capture Timing Calibration . . . . .	390
Signals of Interest . . . . .	391
Proceed to General Board-Level Debug . . . . .	392
<b>Debugging the Virtex-5 FPGA Design . . . . .</b>	<b>392</b>
Introduction . . . . .	392
Verify Placement and Routing . . . . .	392
Signals of Interest . . . . .	392
Physical Layer Debug Port . . . . .	393
Proceed to General Board-Level Debug . . . . .	393
<b>General Board-Level Debug . . . . .</b>	<b>393</b>
Overall Flow . . . . .	393
Isolating Bit Errors . . . . .	394
Board Measurements . . . . .	395
Supply Voltage Measurements . . . . .	395
Clocking . . . . .	395
Synthesizable Testbench . . . . .	395
Varying Read Capture Timing . . . . .	396

## SECTION VI: APPENDICES

### Appendix A: Memory Implementation Guidelines

<b>Generic Memory Interface Guidelines . . . . .</b>	<b>399</b>
Timing Analysis . . . . .	400
Pin Assignments . . . . .	400
Spartan-3/3E/3A/3A DSP FPGA Memory Implementation Guidelines for DDR/DDR2 SDRAM Interfaces . . . . .	400

XIL_ROUTE_ENABLE_DATA_CAPTURE .....	402
Virtex-4 FPGA Direct Clocking Pins .....	402
Virtex-4 FPGA SerDes Clocking and Virtex-5 FPGA Pins .....	403
Termination .....	403
I/O Standards .....	404
Trace Lengths .....	405
<b>Memory-Specific Guidelines .....</b>	<b>405</b>
DDR/DDR2 SDRAM .....	405
Pin Assignments .....	405
Termination .....	406
Trace Lengths .....	406
QDRII SRAM .....	407
Pin Assignments .....	407
Termination .....	407
I/O Standards .....	407
Trace Lengths .....	407
RLDRAM II .....	408
Pin Assignments .....	408
Termination .....	408
I/O Standards .....	408
Trace Lengths .....	408

## Appendix B: Required UCF and HDL Modifications for Pinout Changes

Introduction .....	409
UCF / HDL Constraint Generation Procedure .....	410
Read Data Capture Block Diagram .....	413
UCF / HDL Changes Overview .....	414
Setting HDL Code Top-Level Placement Parameters .....	414
Setting UCF Constraints .....	416
Determining FPGA Element Site Locations .....	416
Setting DQS Gate Circuit Location Constraints .....	417
Setting RLOC_ORIGIN Constraints .....	418
Verifying UCF/HDL Modifications .....	421

## Appendix C: WASSO Limit Implementation Guidelines

### Appendix D: Debug Port

Overview .....	425
Enabling the Debug Port .....	425
Signal Descriptions .....	426
Virtex-5 FPGA: DDR2 SDRAM .....	426
Virtex-5 FPGA: DDR SDRAM .....	429
Virtex-5 FPGA: QDRII SRAM .....	431
Virtex-4 FPGA: DDR SDRAM .....	435
Virtex-4 FPGA: DDRII SRAM .....	437
Virtex-4 FPGA: QDRII SRAM .....	439
Virtex-4 FPGA: RLDRAM II .....	441
Spartan-3 FPGA: DDR/DDR2 SDRAMs .....	443
Adjusting the Tap Delays .....	444

Virtex FPGA Designs . . . . .	444
Spartan-3 FPGA Designs . . . . .	445
<b>Sample Control/Monitoring of the Debug Port . . . . .</b>	<b>446</b>



## About This Guide

---

The Memory Interface Generator (MIG) generates DDRII SRAM, DDR SDRAM, DDR2 SDRAM, QDRII SRAM, and RLDRAM II interfaces for Virtex™-4 FPGAs and generates DDR SDRAM, DDR2 SDRAM, and QDRII SRAM interfaces for Virtex-5 FPGAs. It also generates DDR and DDR2 SDRAM interfaces for Spartan™-3, Spartan-3A, Spartan-3E, and Spartan-3A DSP 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:

- [Section I: “Introduction”](#)
  - ◆ [Chapter 1, “Using MIG,”](#) shows how to install and use the MIG design tool.
- [Section II: “Virtex-4 FPGA to Memory Interfaces”](#)
  - ◆ [Chapter 2, “Implementing DDR SDRAM Controllers,”](#) describes how to implement DDR SDRAM interfaces that MIG creates for Virtex-4 FPGAs.
  - ◆ [Chapter 3, “Implementing DDR2 SDRAM Controllers,”](#) describes how to implement DDR2 SDRAM interfaces that MIG creates for Virtex-4 FPGAs.
  - ◆ [Chapter 4, “Implementing QDRII SRAM Controllers,”](#) describes how to implement QDRII SRAM interfaces that MIG creates for Virtex-4 FPGAs.
  - ◆ [Chapter 5, “Implementing DDRII SRAM Controllers,”](#) describes how to implement DDRII SRAM interfaces that MIG creates for Virtex-4 FPGAs.
  - ◆ [Chapter 6, “Implementing RLDRAM II Controllers,”](#) describes how to implement RLDRAM II interfaces that MIG creates for Virtex-4 FPGAs.
- [Section III: “Spartan-3/3E/3A/3AN/3A DSP FPGA to Memory Interfaces”](#)
  - ◆ [Chapter 7, “Implementing DDR SDRAM Controllers,”](#) describes how to implement DDR SDRAM interfaces that MIG creates for Spartan-3 FPGAs.
  - ◆ [Chapter 8, “Implementing DDR2 SDRAM Controllers,”](#) describes how to implement DDR2 SDRAM interfaces that MIG creates for Spartan-3 FPGAs.
- [Section IV: “Virtex-5 FPGA to Memory Interfaces”](#)
  - ◆ [Chapter 9, “Implementing DDR2 SDRAM Controllers,”](#) describes how to implement DDR2 SDRAM interfaces that MIG creates for Virtex-5 FPGAs.

- ◆ Chapter 10, “Implementing QDRII SRAM Controllers,” describes how to implement QDRII SRAM interfaces that MIG creates for Virtex-5 FPGAs.
- ◆ Chapter 11, “Implementing DDR SDRAM Controllers,” describes how to implement DDR SDRAM interfaces that MIG creates for Virtex-5 FPGAs.
- Section V: “DDR2 Debug Guide”
  - ◆ Chapter 12, “Debugging MIG DDR2 Designs,” provides a step-by-step process for debugging designs that use MIG-generated memory interfaces.
- Section VI: “Appendices”
  - ◆ Appendix A, “Memory Implementation Guidelines,” provides helpful rules for reference designs.
  - ◆ Appendix B, “Required UCF and HDL Modifications for Pinout Changes,” provides detailed information about modifying pinout-dependent UCF constraints and top-level parameters when required by various design circumstances.
  - ◆ Appendix C, “WASSO Limit Implementation Guidelines,” gives references to data and tools necessary for ensuring compliance with Simultaneous Switching Output (SSO) limitations.
  - ◆ Appendix D, “Debug Port,” provides information on the Debug port added to all memory interface designs for MIG 2.2 and later.

## 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](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](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/rl dram/part.aspx?part=MT49H16M18FM-25>
5. Micron Data Sheet MT46V16M16FG-5B  
<http://www.micron.com/products/dram/ddrsdram/partlist.aspx>
6. Xilinx ChipScope™ Pro documentation  
<http://www.xilinx.com/literature/literature-chipscope.htm>
7. [UG070](#), *Virtex-4 User Guide*
8. [UG072](#), *Virtex-4 PCB Designer’s Guide*
9. [UG079](#), *Virtex-4 ML461 Memory Interfaces Development Board User Guide*
10. [UG190](#), *Virtex-5 FPGA User Guide*
11. [UG203](#), *Virtex-5 PCB Designer’s Guide*
12. [UG195](#), *Virtex-5 FPGA Packaging and Pinout Specification*
13. [UG199](#), *Virtex-5 ML561 Memory Interfaces Development Board User Guide*
14. [XAPP454](#), *DDR2 SDRAM Memory Interface for Spartan-3 FPGAs*
15. [XAPP458](#), *Implementing DDR2-400 Memory Interfaces in Spartan-3A FPGAs*

16. [XAPP645](#), *Single Error Correction and Double Error Detection*
17. [XAPP701](#), *Memory Interfaces Data Capture Using Direct Clocking Technique*
18. [XAPP702](#), *DDR-2 Controller Using Virtex-4 Devices*
19. [XAPP703](#), *QDR II SRAM Interface*
20. [XAPP709](#), *DDR SDRAM Controller Using Virtex-4 FPGA Devices*
21. [XAPP710](#), *Synthesizable CIO DDR RLD RAM II Controller for Virtex-4 FPGAs*
22. [XAPP721](#), *High-Performance DDR2 SDRAM Memory Interface Data Capture Using ISERDES and OSERDES*
23. [XAPP768c](#), *Interfacing Spartan-3 Devices With 166 MHz or 333 Mb/s DDR SDRAM Memories* (available under click license)
24. [XAPP851](#), *DDR SDRAM Controller Using Virtex-5 FPGA Devices*
25. [XAPP853](#), *QDR II SRAM Interface for Virtex-5 Devices*
26. [XAPP858](#), *High-Performance DDR2 SDRAM Interface In Virtex-5 Devices*
27. [DS099](#), *Spartan-3 FPGA Family: Complete Data Sheet*
28. [DS312](#), *Spartan-3E FPGA Family: Complete Data Sheet*
29. [Chapter 6: PARTGen](#), *Development System Reference Guide*
30. WASSO Calculator for Virtex-4 devices  
<https://secure.xilinx.com/webreg/clickthrough.do?cid=30163>
31. WASSO Calculator for Virtex-5 devices  
<https://secure.xilinx.com/webreg/clickthrough.do?cid=30154>
32. Micron Technical Note TN-47-01, *DDR2-533 Memory Design Guide for Two-DIMM Unbuffered Systems*  
[http://download.micron.com/pdf/technotes/ddr2/tn\\_47\\_01.pdf](http://download.micron.com/pdf/technotes/ddr2/tn_47_01.pdf)

## 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
<i>Italic font</i>	References to other documents	See the <i>Virtex-4 Configuration Guide</i> for more information.
	Emphasis in text	The address (F) is asserted <i>after</i> clock event 2.
<u>Underlined Text</u>	Indicates a link to a web page.	<a href="http://www.xilinx.com/virtex4">http://www.xilinx.com/virtex4</a>

## Type Case of Port and Signal Names

Some port and signal names given in the figures and tables in this document might appear in uppercase type, even though those same names are in lowercase type in the designs themselves. This is strictly a typographical issue in the User Guide, and does not imply that the port and signal names in the designs need to be changed.





## *Section I: Introduction*

### *Chapter 1, "Using MIG"*



## Using MIG

---

MIG is a tool used to generate memory interfaces for Xilinx FPGAs. MIG generates Verilog or VHDL RTL design files, user constraints file (UCF) constraints, and script files. The script files are used to run simulations, synthesis, map, and par for the selected configuration.

This chapter describes the user interface details of all memory interfaces supported in MIG. It provides MIG features, usage, and installation details and describes the output files. This chapter also summarizes the changes and enhancements made from earlier versions of MIG.

### MIG 2.2 Changes from MIG 2.1

The new features of MIG 2.2 are summarized in this section:

- Support of Qimonda memory parts for the DDR2 SDRAM interface of all FPGA families.
- Multiple interface support in Virtex™-5 FPGAs for DDR2 SDRAM and QDRII SRAM designs:
  - ◆ Provides an option to select DDR2 SDRAM and QDRII SRAM interfaces for multicontroller designs.
  - ◆ Supports different frequencies for different memory interfaces.
  - ◆ Provides controller-wise DCI Cascade support.
- Creates different UCF files for all the selected compatible FPGAs.
- Enhanced support for the Debug port option using VIO.
- Updates to Virtex-5 and Virtex-4 FPGA designs:
  - ◆ Supports updated designs.
  - ◆ Provides an option to browse the old project file (.prj) in the Verify UCF page.
  - ◆ Provides IDELAYCTRL location constraints in the UCF.
- Added Debug port to all memory interface designs

## MIG 2.1 Changes from MIG 2.0

The new features of MIG 2.1 are summarized in this section:

- Support for 64-bit/32-bit Linux Red Hat Enterprise 4.0
- Support for 64-bit Microsoft Windows® XP Professional
- Support for 32-bit Microsoft Vista Business
- Support for 64-bit SUSE 10 Enterprise
- Data mask enable/disable option for DDR and DDR2 SDRAM designs
- Debug signals support
- Real-time pin allocation implemented in the GUI. As the user selects the banks, the GUI displays the information as the total number of required pin count and the number of pins allocated for each group of signals.
- Implements the priority bank selection for the data. Priority is given for exclusive Data banks first, then Data banks with the combination of other groups.
- Creates the RLOC and DQS gate constraints to older versions of UCF files that use the design from MIG 2.0 or following versions for Virtex-5 FPGA DDR2 SDRAMs. An option is provided to add or not add the constraints while verifying the UCF.
- Simulations support for custom memory parts
- Reserve Pin banks are changed from list view to hierarchical view
- Implemented the DCI Cascade and Master Bank selection option for QDRII SRAM Virtex-5 FPGA designs
- Support for Spartan-3A FPGA DDR2 SDRAM 200 MHz design
- 166 MHz frequency support for all possible data widths for Spartan-3E, Spartan-3A, and Spartan-3A DSP families
- Uncommon banks are faded out in the Bank Selection page when the user selects compatible FPGAs, allowing only the common banks for pin allocation
- Attributes X\_CORE\_INFO and CORE\_GENERATION\_INFO support for all designs
- Updates to Virtex-5 FPGA designs:
  - ◆ DDR2 SDRAM
    - Changing the MIG 1.73 or prior versions of UCF files compatible to MIG 2.0 or following versions of designs using Verify UCF feature
  - ◆ QDRII SRAM
    - BL2 support
    - DCI cascade support
- Updates to Virtex-4 FPGA designs:
  - ◆ DDR2 SDRAM Direct Clocking
    - CAS latency 5 support
    - Linear addressing support from the user interface
    - Calibration algorithm modified to fix the low-frequency issues
  - ◆ DDR2 SDRAM SerDes
    - Linear addressing support from the user interface
  - ◆ DDR SDRAM
    - Linear addressing support from the user interface

- ◆ DDRII SRAM
  - Two address FIFOs replaced by a common address FIFO for both write and read commands
- Updates to Spartan FPGA designs:
  - ◆ DDR2 SDRAM and DDR SDRAM
    - Linear addressing support from the user interface

For MIG 2.1 release notes and a list of specific issues addressed in this release, consult Xilinx Answer Record [29767](#).

## MIG 2.0 Changes from MIG 1.73

The new features of MIG 2.0 are summarized in this section:

- MIG GUI is changed to WIZARD implementation
- Supports 32-bit Linux Red Hat Enterprise 4.0
- Generates a compatible simulation testbench for the generated design
- Supports Preset Configuration
- Updates to Virtex-5 FPGA designs:
  - ◆ DDR SDRAM
    - Support for DIMMs
  - ◆ DDR2 SDRAM
    - Major physical layer changes: Read capture architecture modified, support added for read postamble DQS glitch gating, operation of PHY logic at half clock speed. See [XAPP858](#) for details.
    - Support for unbuffered DIMMs. Implemented 2T timing to support unbuffered DIMMs
    - 72-bit ECC support
  - ◆ QDRII SRAM
    - Partial support for DCI Cascade
    - Allocating CQ, CQ# pins
    - Allocating K, K# to P and N pairs
    - Read data FIFOs removed from the user interface
- Unsupported features:
  - ◆ Edit signal names

For MIG 2.0 release notes and a list of specific issues addressed in this release, consult Xilinx Answer Record [29312](#).

## MIG 1.73 Changes from MIG 1.72

The new features of MIG 1.73 are summarized in this section:

- Spartan-3A DSP FPGAs are supported

## MIG 1.72 Changes from MIG 1.7

There are no new features added to this release from MIG 1.7.

For MIG 1.72 release notes and a list of specific issues addressed in this release, consult Xilinx Answer Record [25056](#).

## MIG 1.7 Changes from MIG 1.6

The new features of MIG 1.7 are summarized in this section:

- Supports creating a new memory part by modifying an existing part
- Generates a script file to create an ISE™ project
- Updates to Virtex-5 FPGA designs:
  - ◆ Supports DDR SDRAM Verilog and VHDL
  - ◆ Supports QDRII SRAM and DDR2 SDRAM VHDL
- Updates to Virtex-4 FPGA designs:
  - ◆ DDR2 SDRAM
    - ECC supported in Pipelined or Unpipelined modes
    - Add per-bit deskew for DDR2 Direct clocking
    - Change SerDes clock scheme
  - ◆ QDRII SRAM
    - No DCM support
  - ◆ DDRII SRAM
    - No DCM support
- Updates to Spartan-3 FPGA designs:
  - ◆ Spartan-3A FPGA support for DDR and DDR2 SDRAMs
  - ◆ Pinout compatibility with MIG 1.6 and MIG 1.5 versions for Spartan-3 and Spartan-3E devices. There are several limitations to this feature. Contact Xilinx support for more details.

For MIG 1.7 release notes and a list of specific issues addressed in this release, consult Xilinx Answer Record [25406](#).

## MIG 1.6 Changes from MIG 1.5

The new features of MIG 1.6 are summarized in this section:

- Supports Virtex-5 FPGA interfaces
- Outputs two different folders with and without a testbench for the selected memory interface. This feature is supported for all interfaces.
- Supports batch mode
- Virtex-4 FPGA GUI changes
  - ◆ DDR SDRAM
    - No DCM support
  - ◆ RLDRAM II
    - No DCM support

- ◆ DCI for data
- ◆ DCL for address and control
- Spartan-3 FPGA GUI changes
  - ◆ DDR2 SDRAM
    - No DCM support
- Removed Add Testbench button. The tool by default outputs with and without testbench designs, hence it is not required to have the Add Testbench button.

## MIG 1.5 Changes from MIG 1.4

The new features of MIG 1.5 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 FPGA interfaces) support:
  - ◆ Synplicity Synplify 8.2 support
  - ◆ SODIMM support
  - ◆ Modified Read Enable implementation
- ISE 8.1.01i tool support (all MIG 1.5 designs support this ISE tool version)
- DDR2 SDRAM SerDes clocking (Virtex-4 FPGA interfaces) support
- DDR SDRAM for Virtex-4 FPGA interfaces:
  - ◆ Synplicity Synplify 8.2 support
  - ◆ CL = 2, 2.5, and 4
  - ◆ BL = 2 and 8
  - ◆ SODIMMs
  - ◆ 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 DDRII SRAMs:
  - ◆ Synplicity Synplify 8.2 support
- Supports skip wait 200  $\mu$ s delay for Verilog simulations. This feature is not supported for VHDL cases.
  - ◆ To skip 200  $\mu$ s initial delay, users should use the following run-time options for Verilog in ModelSim.
  - ◆ For DDR SDRAM for Virtex-4 FPGA 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 FPGA interfaces:  

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

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



## Tool Features

The key features of MIG are listed below:

- Supported memory types for Virtex-5 FPGA interfaces:
  - ◆ DDR2 SDRAM components and single-rank DIMMs  
See [“Supported Devices” in Chapter 9](#) for a complete listing of supported devices.
  - ◆ QDRII SRAM  
See [“Supported Devices” in Chapter 10](#) for a complete listing of supported devices.
  - ◆ DDR SDRAM components and single-rank DIMMs  
See [“Supported Devices” in Chapter 11](#) for a complete listing of supported devices.

Both Verilog and VHDL RTL are generated. Additional devices can be created using the “Create Custom Part” feature.

- Supported memory types for Virtex-4 FPGA interfaces:
  - ◆ DDR SDRAM components, registered DIMMs, unbuffered DIMMs, and SODIMMs.  
See [“Supported Devices” in Chapter 2](#) for a complete listing of supported devices.
  - ◆ DDR2 SDRAM components and single-rank DIMMs. The DDR2 controller supports deep memory depths from one to four.  
See [“Supported Devices” in Chapter 3](#) for a complete listing of supported devices.
  - ◆ QDRII and DDRII SRAMs  
See [“Supported Devices” in Chapter 4](#) for a complete listing of supported QDRII devices.  
See [“Supported Devices” in Chapter 5](#) for a complete listing of supported DDRII devices.
  - ◆ RLDRAM II CIO and SIO memories  
See [“Supported RLDRAM II Devices” in Chapter 6](#) for a complete listing of supported devices.

Additional devices can be created using the “Create Custom Part” feature.

- Supported memory types for Spartan-3 FPGA interfaces:
  - ◆ DDR SDRAM components, registered DIMMs, unbuffered DIMMs, and SODIMMs.  
See [“Supported Devices” in Chapter 7](#) for a complete listing of supported devices.
  - ◆ DDR2 SDRAM components, registered DIMMs, unbuffered DIMMs, and SODIMMs.  
See [“Supported Devices” in Chapter 8](#) for a complete listing of supported devices.

Additional devices can be created using the “Create New Memory Part” feature.

- Supported memory types for Spartan-3E FPGA interfaces:
  - ◆ DDR SDRAM components  
See [“Supported Devices” in Chapter 7](#) for a complete listing of supported devices.

Additional devices can be created using the “Create New Memory Part” feature.

- Supported memory types for Spartan-3A/3AN FPGA interfaces:
  - ◆ DDR SDRAM components, registered DIMMs, unbuffered DIMMs, and SODIMMs.  
See “Supported Devices” in Chapter 7 for a complete listing of supported devices.
  - ◆ DDR2 SDRAM components, registered DIMMs, unbuffered DIMMs, and SODIMMs.  
See “Supported Devices” in Chapter 8 for a complete listing of supported devices.  
Additional devices can be created using the “Create New Memory Part” feature.
- Supported memory types for Spartan-3A DSP FPGA interfaces:
  - ◆ DDR SDRAM components, unbuffered DIMMs, and SODIMMs.  
See “Supported Devices” in Chapter 7 for a complete listing of supported devices.
  - ◆ DDR2 SDRAM components, unbuffered DIMMs, and SODIMMs.  
See “Supported Devices” in Chapter 8 for a complete listing of supported devices.  
Additional devices can be created using the “Create New Memory Part” feature.
- Supported synthesis and place-and-route tools:
  - ◆ XST (Xilinx ISE Design Suite 10.1) and Synplify Pro Version 8.8.0.4 are supported for Virtex-5, Virtex-4, and Spartan-3/3E/3A/3AN/3A DSP FPGA interfaces
- All currently available Virtex-5, Virtex-4, Spartan-3A, Spartan-3AN, Spartan-3A DSP, Spartan-3E, and Spartan-3 FPGAs are supported.
- DDR2 designs can use either the SerDes or the Direct clocking technique. The individual bits are deskewed in the Direct clocking technique used in DDR2 designs. The Direct clocking technique for other memories does not deskew each bit. Details are explained in the appropriate application notes referenced in this document.
- Direct and SerDes clocking techniques for data capture for Virtex-4 FPGA interfaces.  
Direct clocking using per-bit skew is explained in XAPP701 [Ref 17]. 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 22]. 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, Spartan-3A/3AN/3A DSP, and Spartan-3E FPGA interfaces.  
The data capture technique using Spartan-3 FPGAs is explained in XAPP768c [Ref 23].
- VHDL and Verilog RTLs are supported for all designs.
- Variable data widths in multiples of 8 up to 144 bits.  
The actual width depends upon the selected component. For a 9-bit wide component, data widths of 9, 18, 36, and 72 are supported.  
For DDR2 SDRAM, most of the components support up to a 144-bit data width. 16-bit or 8-bit wide components can be used to create designs of any data width that is a multiple of 8.
- User-selectable banks for address, data, system control, and system clock signals.  
For QDRII SRAM and RLDRAM II (SIO) memories, the user selects the data banks for reads and writes separately.
- Different banks are supported with different I/O standards.

MIG 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 MIG to use. If insufficient banks are selected, MIG 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 for Virtex-4 FPGA designs.  
Users can create up to eight controllers.
- Multiple DDR2 and QDRII interfaces for Virtex-5 FPGA designs and the combination of both interfaces can be selected.
- Different frequencies can be set for different memory interfaces in Virtex-5 FPGA designs.
- Pin compatibility.  
Users can select multiple devices with the same package to generate compatible pinouts.
- Update UCF.  
Users can update the old UCF files to be compatible with the latest MIG designs.

## Design Tools

All MIG designs have been tested with ISE Design Suite 10.1 and Synplify Pro. MIG is currently supported on the following operating systems: 64-bit/32-bit Microsoft Windows XP, 64-bit/32-bit Linux Red Hat Enterprise 4.0, 32-bit Vista Business, and 64-bit SUSE 10 Enterprise.

## Installation

MIG provides Xilinx CORE Generator™ reference designs and is included in the latest IP update. IP updates are available through the Xilinx Download Center or WebUpdate. Visit the Xilinx Download Center for the latest IP update and full documentation on both installation methods at <http://www.xilinx.com/download>.

## Getting Started

MIG is a self-explanatory tool. This section is intended to help with understanding the various steps involved in using it.

The following steps launch MIG:

1. The CORE Generator system is launched by selecting **Start** → **Xilinx ISE Design Suite 10.1** → **ISE** → **Accessories** → **CORE Generator**.
2. Create a CORE Generator project.
3. The Xilinx part must be correctly set because it cannot be changed inside MIG. Virtex-5, Virtex-4, and Spartan-3/Spartan-3E/Spartan-3A/3AN/3A DSP 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" and "vendor" must be chosen appropriately. The vendor choices are "Synplicity" for Synplify and "ISE" for XST.

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. MIG is launched by selecting **Memories & Storage Elements** → **Memory Interface Generator** → **MIG**.
6. The name of the module to be generated is entered in the Component Name text box. After entering all the parameters in the GUI, 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 **Close** button.

The "Generated IP" tab to the left lists the generated modules.

## MIG User Interface

### Getting Help

At any point in time, the MIG user manual can be accessed by clicking the **User Guide** button.

### Version Information

The **Version Info** Button gives the information on new features added and the bugs fixed in the current version. It opens the web browser to display the contents.

## CORE Generator Options

[Memory Interface Generator](#)

The Memory Interface Generator (MIG) creates memory controllers for Xilinx FPGAs. MIG creates complete customized Verilog or VHDL RTL source code, pin-out and design constraints for the FPGA selected, and script files for implementation and simulation.

---

[CORE Generator Options](#)

This GUI includes all configurable options along with explanations to aid in generation of the required controller. Please note that some of the options selected in the CORE Generator Project Options will be used in generation of the controller. It is very important that the correct CORE Generator Project Options are selected. These options are listed below.

Selected CORE Generator Project Options:

<b>FPGA Family</b>	Virtex-4
<b>FPGA Part</b>	xc4vlx25-ff668
<b>Speed Grade</b>	-10
<b>Synthesis Tool</b>	XST
<b>Design Entry</b>	VERILOG

**If any of these options are incorrect, please click on "Cancel", change the CORE Generator Project Options, and restart MIG .**

UG086\_c1\_04\_091307

Figure 1-1: CORE Generator Options

The CORE Generator Options screen displays the details of the selected CORE Generator options that are selected before invoking MIG.

**Note:** CORE Generator project options are used in the generation of the memory controller. Correct CORE Generator project options must be selected.

If the displayed CORE Generator Project Options are inaccurate, click the **Cancel** button and reselect the CORE Generator Project Options.

Click **Next** to move ahead. A new window shows the MIG Output Options page.

## MIG Output Options

MIG can have five different output options. They are:

1. [Create Design](#)
2. [Create Design for Xilinx Reference Boards](#)
3. [Verify UCF/Update Design](#)
4. [Create Preset Configuration](#)
5. [Spartan-3A FPGA DDR2 SDRAM 200 MHz Design](#)

MIG outputs are generated with folder name *<Component Name>*.

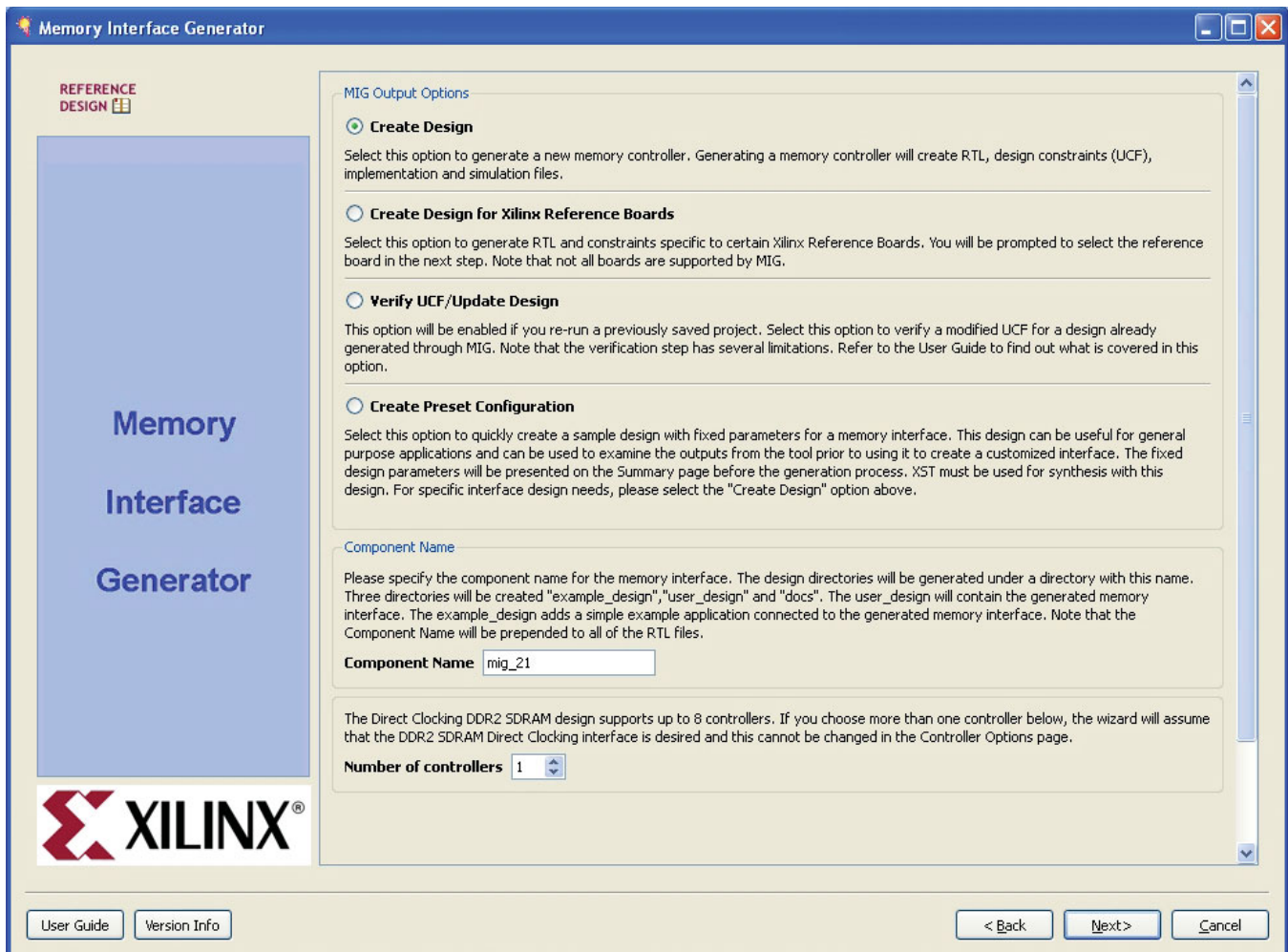
**Note:** *<Component Name>* does not accept special characters. Only alphanumeric characters can be used to specify a component name. It should always start with an alphabet character and can end with an alphanumeric character.

For multicontroller applications, the number of controllers should be selected at the **Number of controllers** spin box. More than one controller can be selected for DDR2 SDRAM Direct clocking interface for Virtex-4 FPGA designs and for DDR2 SDRAM and QDRII SRAM designs in Virtex-5 FPGA designs. In case more than one controller is selected, MIG limits the design generation to DDR2 SDRAM for Virtex-4 FPGA designs, and MIG limits the design generation to DDR2 SDRAM and QDRII SRAM for Virtex-5 FPGA designs. Select the appropriate number (1-8) in the pull-down menu. The **Number of controllers** selection is enabled only for Virtex-5 and Virtex-4 FPGA families.

The Create Preset Configuration option is not supported for Virtex-5 FPGA designs, and the Verify UCF/ Update Design option is not supported for Spartan designs.

**Note:** The Create Design option can use a multiple number of controllers. For the Create Design for Xilinx Reference Boards, Verify UCF/Update Design, and Create Preset Configuration options, the number of controllers is limited to one.

Click **Back** to return to previous page. Click **Cancel** to quit from the tool. Click **Next** to move ahead. The next page display depends upon the options selected in the current page.

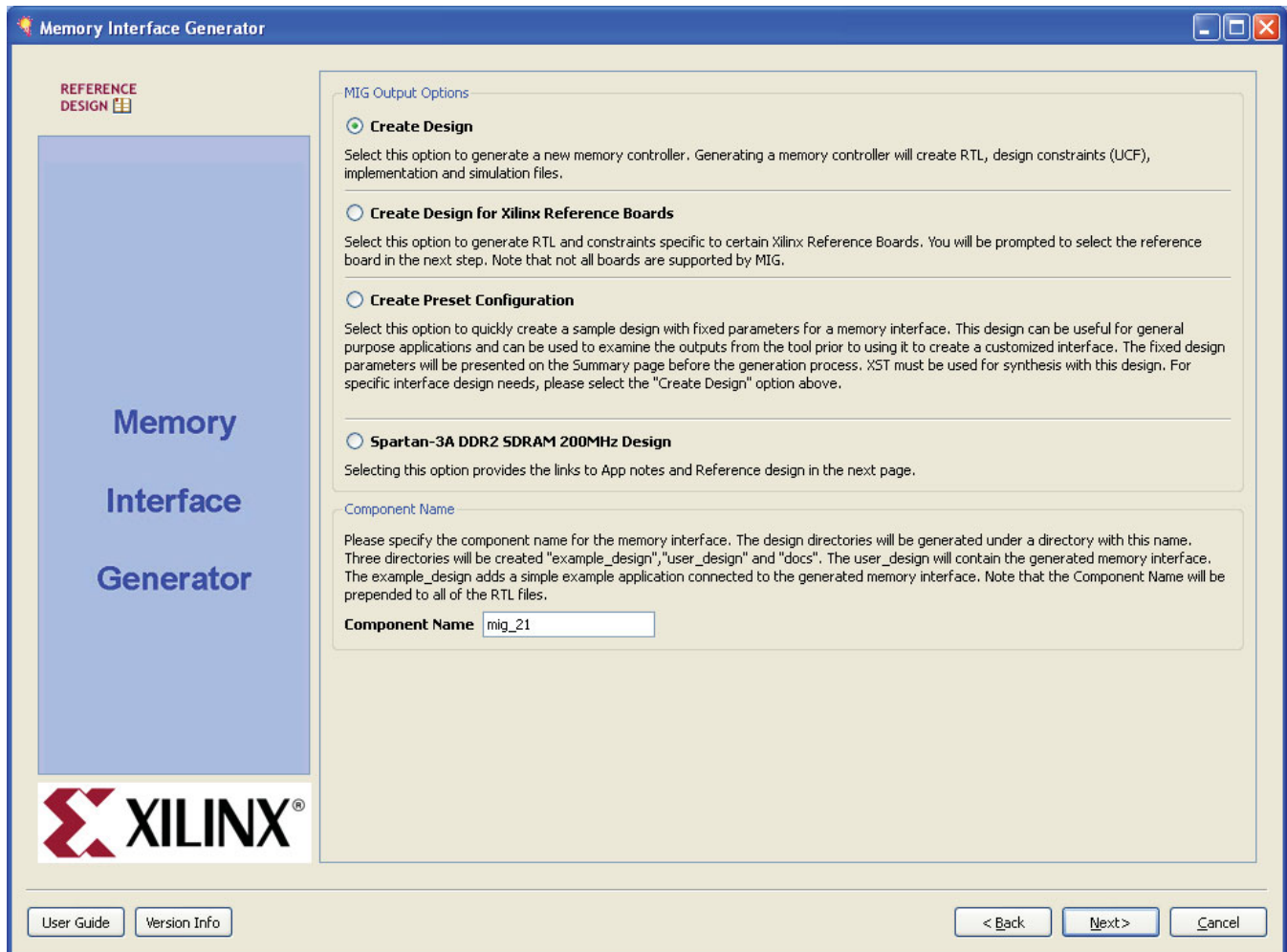


UG086\_c1\_05\_022008

Figure 1-2: MIG Output Options / Component Name / Number of Controllers



The Spartan-3A DDR2 SDRAM 200MHz Design option appears only for Spartan-3A FPGA designs (see [Figure 1-3](#)).



UG086\_c1\_55\_022008

Figure 1-3: MIG Output Options Page of Spartan-3A FPGA Design

## Create Design

Using the Create Design option, designs can be generated that are supported for that FPGA family. For example, the Virtex-4 family supports DDR2 SDRAM, DDR SDRAM, QDRII SRAM, DDRII SRAM, and RLDRAM II. Here is the flow for creating a design:

1. [Pin Compatible FPGAs](#)
2. [Memory Selection](#)
3. [Controller Options](#)
4. [Set Mode Registers](#)
5. [Set Extended Mode Registers](#)
6. [FPGA Options](#)
7. [Reserve Pins](#)
8. [Bank Selection](#)
9. [Summary](#)

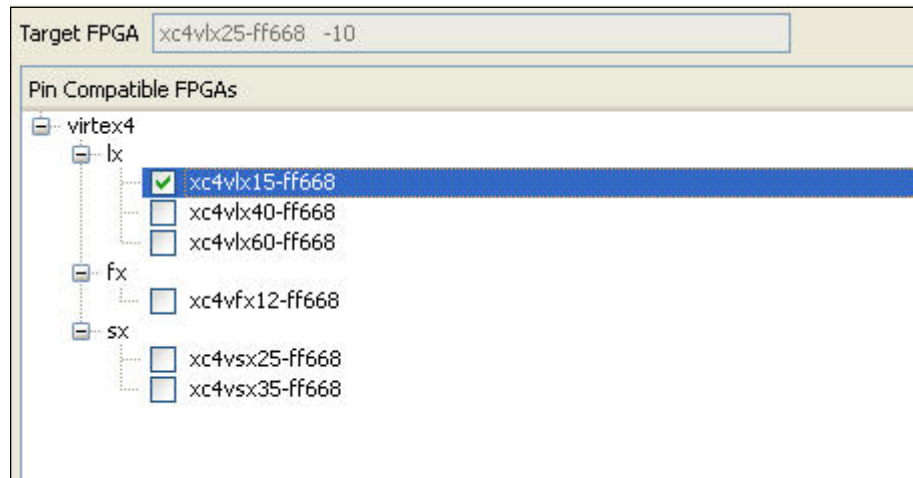
- 10. [Memory Model License](#)
- 11. [PCB Information](#)
- 12. [Finish](#)

All the options are described in this section.

### Pin Compatible FPGAs

FPGAs in the selected family with the same package are listed here. In case the generated pinout from MIG needs to be reusable with any of these other FPGAs, use this option to select the FPGAs with which the pinout has to be compatible.

**Note:** The SerDes design is only supported for FPGAs with PMCDs. In case the target FPGA or the selected compatible FPGA has no PMCD, the capture method for DDR2 SDRAM is restricted to Direct clocking.



UG086\_c1\_06\_122707

Figure 1-4: Pin Compatible FPGAs

Select any number of compatible FPGAs out of the listed ones. Only the common pins between target and selected FPGAs are used by MIG. The name in the text box signifies the Target FPGA selected. Click **Next** to move ahead. The Memory Selection is displayed.

### Memory Selection

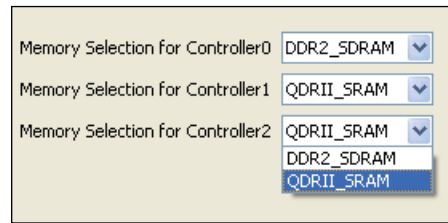
This page displays all memory types that are supported by the selected FPGA family. An example is shown in [Figure 1-5](#) for Virtex-4 FPGA designs and in [Figure 1-6](#) for Virtex-5 FPGA designs. In Virtex-5 FPGA designs, the user can select the combination of both DDR2 SDRAM and QDRII SRAM interfaces for a multicontroller design.



UG086\_c1\_07\_083007

Figure 1-5: Memory Selection for Virtex-4 FPGA Designs





UG086\_c1\_56\_022208

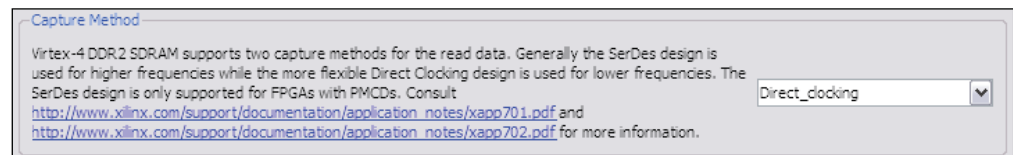
Figure 1-6: Memory Selection for Virtex-5 FPGA Designs

Select the appropriate option, and then click **Next** to move ahead. The Controller Options window is displayed.

### Controller Options

This page shows the various controller options that can be selected. If the design has multiple controllers, this page is repeated for each of the controllers. The page is partitioned into a maximum of nine sections. The number of partitions depends on the type of selected memory.

- **Capture Method.** This feature deals with the data capture method. The DDR2 SDRAM controller for Virtex-4 devices supports two types of capture method. For other designs, the capture method is displayed, but it cannot be changed.

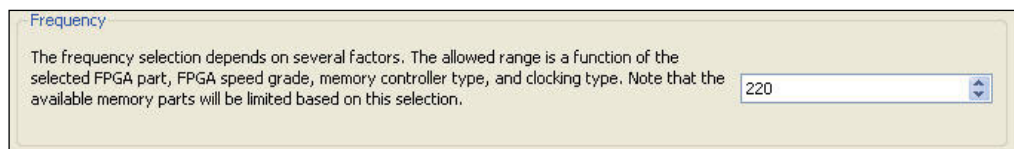


UG086\_c1\_08\_022708

Figure 1-7: Capture Method

Click the pull-down menu button and select an option. Certain other options such as frequency and ECC are restricted based on this selection.

- **Frequency.** This feature indicates the desired frequency for all the controllers. This frequency block is limited by factors such as the selected FPGA, device speed grade, and clocking type.



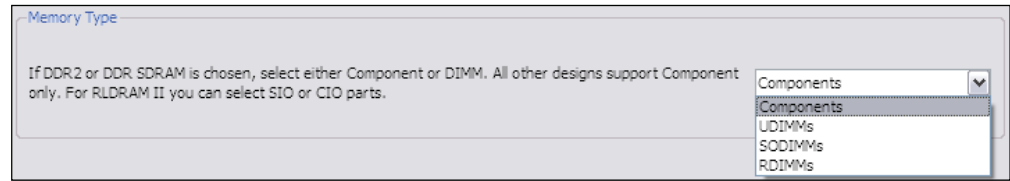
UG086\_c1\_09\_022708

Figure 1-8: Frequency

Vary the frequency as required. Either use the spin box or enter a valid value through the keyboard. Values entered are restricted based on the minimum and maximum frequencies supported.

**Note:** For a Virtex-4 multicontroller design, the frequency selected for the first controller is used for all other controllers with the same memory interface. Memory parts and data width are restricted based on the frequency selection.

- **Memory Type.** For DDR2 SDRAM, MIG categorizes different memory components and modules available into components, UDIMMs, SODIMMs, and RDIMMs. This can vary according to the memory selected.

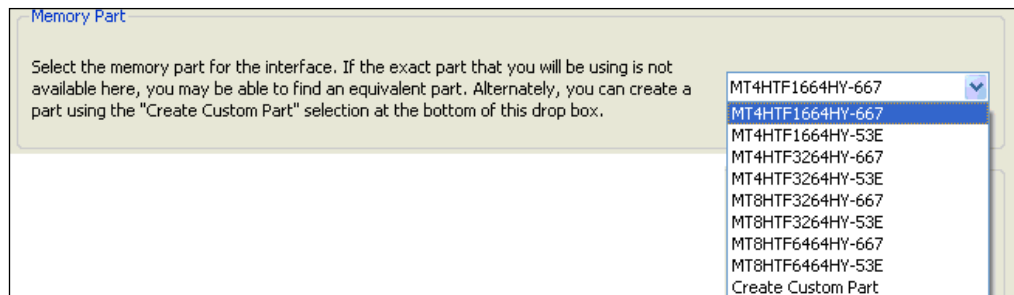


UG086\_c1\_10\_022708

Figure 1-9: Memory Type

Click the pull-down menu combo box and select the memory type. This selection restricts the available choices in memory part selection list and data width.

- **Memory Part.** This feature helps the selection of a memory part for the design. Selection can be made from an existing list, or a new part can be created.

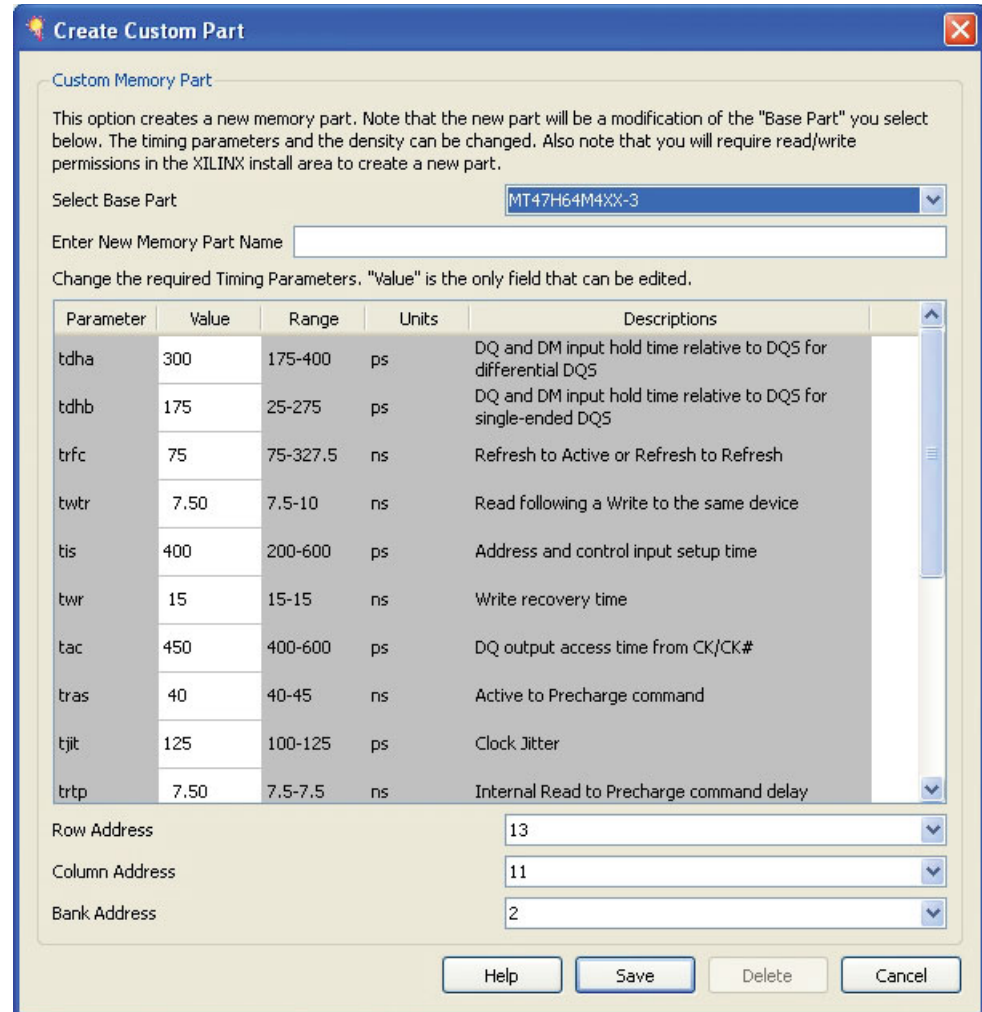


UG086\_c1\_11\_091707

Figure 1-10: Memory Part

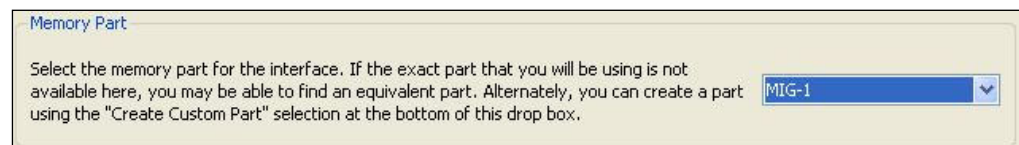
Select the appropriate memory part from the list. If the required part or its equivalent is unavailable, a new memory part can be created. To create a custom part, select the Create Custom Part from the drop down combo box. A new window appears as shown in Figure 1-11.

The window called Create Custom Part includes all the details of the memory component selected in Select Base Part. Enter the appropriate memory part name in the text box. Select the suitable base part from the Select base part list. Edit the Value column as needed. Select the suitable values from the Row, Column, and Bank options as per the requirements. After editing the required fields, click the **Save** button. The new part can be saved with the selected name. This new part is added in the Memory Parts list as shown in Figure 1-12 and saved into the database for reuse and to produce the design.



UG086\_c1\_12\_022008

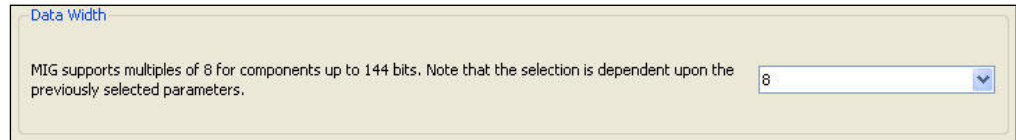
Figure 1-11: Create Custom Part



UG086\_c1\_13\_083007

Figure 1-12: Memory Part

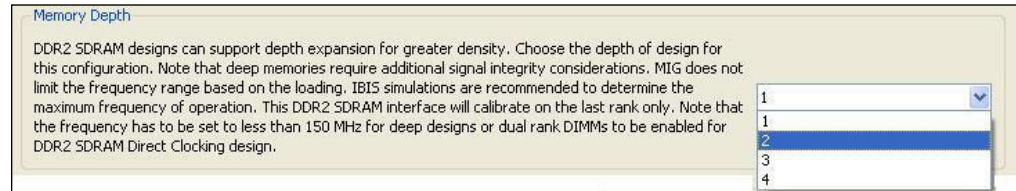
- **Data Width.** The data width value can be selected here based on the memory type selected earlier. The list shows all supported data widths for the selected part. Choose one of them. These values are generally multiples of the individual device data widths. In some cases, the width might not be an exact multiple. For example, though 16 bits is the default data width for x16 components, 8 bits is also a valid value.



UG086\_c1\_14\_122907

Figure 1-13: Data Width

- **Memory Depth.** The DDR2 SDRAM Virtex-4 FPGA controller with Direct clocking as capture method and frequency less than or equal to 150 MHz supports memory depth of one to four. For other designs, this option is unavailable.

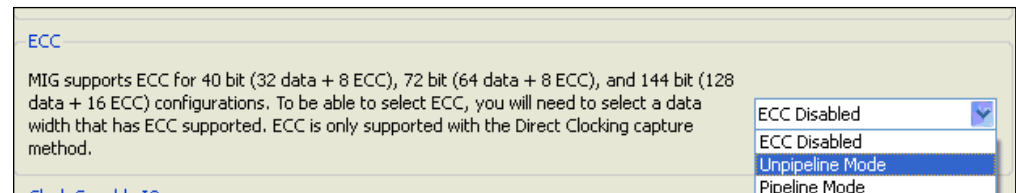


UG086\_c1\_15\_122907

Figure 1-14: Memory Depth

Select the appropriate option from the Memory Depth option.

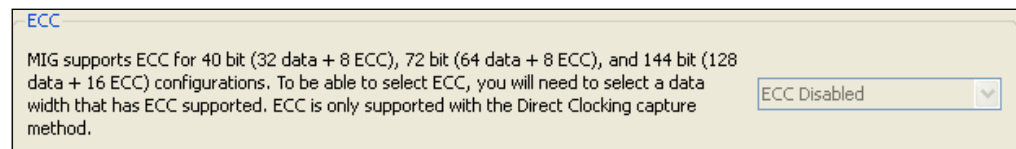
- **ECC.** ECC stands for Error Correction Code. This feature enables the generation of ECC along with the code. This section is enabled based on selected data width. This option is available only for DDR2 SDRAM Virtex-4 and Virtex-5 FPGA designs.



UG086\_c1\_16\_090407

Figure 1-15: ECC (a)

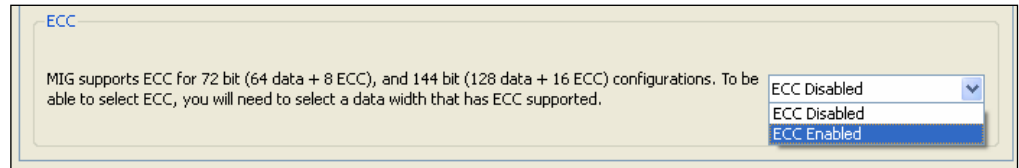
Note that ECC selection is enabled only when the appropriate data width is selected. DDR2 SDRAM Virtex-4 FPGA design supports three modes: ECC Disabled, Unpipeline Mode, and Pipeline Mode, as shown in Figure 1-15. Select the appropriate mode. The Pipeline mode improves frequency performance at the cost of an extra pipeline stage.



UG086\_c1\_17\_090407

Figure 1-16: ECC (b)

For other Virtex-4 FPGA designs, this window is disabled as shown in Figure 1-16. For Virtex-5 FPGA DDR2 SDRAM designs, the two options are ECC Enabled and ECC Disabled.

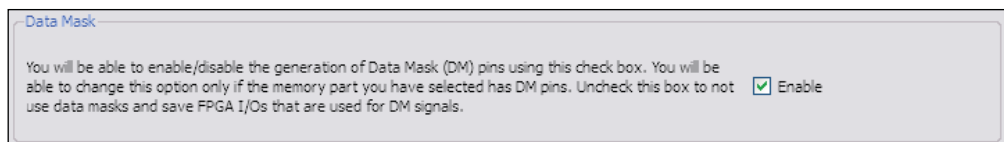


UG086\_c1\_18\_083007

Figure 1-17: ECC (c)

Figure 1-17 shows the ECC option section for the Virtex-5 FPGA design GUI. For Virtex-5 devices, ECC is supported for 72-bit or 144-bit DDR2 SDRAM designs.

- **Data Mask.** When this Data Mask check box is marked, the Data Mask pins are allocated. When this Data Mask check box is not checked, the data mask pins are not allocated, which increases the pin efficiency. This option is disabled and cannot be changed for memory parts that do not support data masks. This option is available only for DDR2 and DDR SDRAMs.

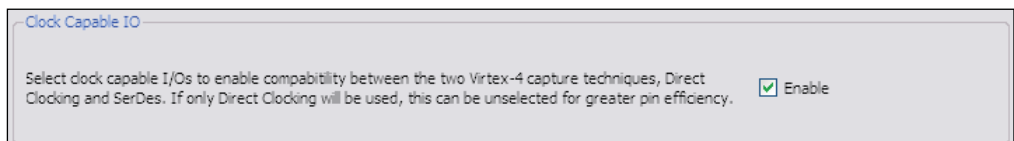


UG086\_c1\_43\_022708

Figure 1-18: Data Mask

Select the option as per the requirement.

- **Clock Capable I/O.** Checking the Clock Capable I/O box makes use of the CC pins available in Virtex-4 FPGAs for strobes or read clocks. This option is enabled and cannot be changed for DDR2 SDRAM SerDes designs, but is editable for other designs.

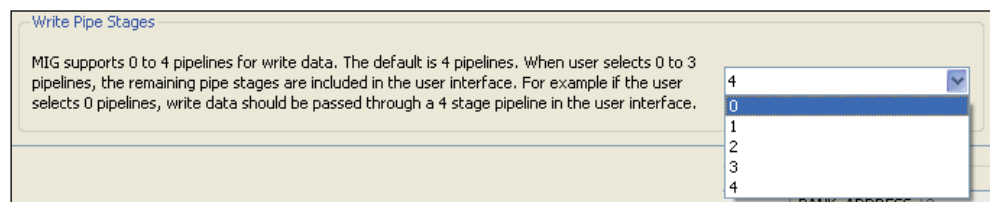


UG086\_c1\_19\_022708

Figure 1-19: Clock Capable I/O

Select the option as per the requirement.

- **Write Pipe Stages.** The Write Pipe Stages is supported only for Spartan FPGA designs. This option allows users to implement the write data pipelines in the user interface.



UG086\_c1\_57\_022208

Figure 1-20: Write Pipe Stages

- **Memory Details.** This section displays details about the selected memory. For DIMMs, the details listed are the base component memory details.

UG086\_c1\_20\_091307

Figure 1-21: Memory Details

The memory details change based on the selected Memory part.

Click **Next** to move ahead. The Set Mode Registers window is displayed for RLD RAM II, DDR, and DDR2 SDRAM devices. For other memories, the next window displayed is FPGA Options.

### Set Mode Registers

This feature allows selection of various memory mode register values as supported by the controller type.

UG086\_c1\_21\_022208

Figure 1-22: Mode Register Data

The Mode Register Value is loaded into the Load Mode register during initialization.

**Note:** CAS latency values listed on this GUI are restricted by the frequency and the memory part selected in the prior page.

Click **Next** to move ahead. The Set Extended Mode Register window or FPGA Options window is displayed.

## Set Extended Mode Registers

Select the memory extended mode register values here. This page appears for DDR SDRAM and DDR2 SDRAM only, and the contents can change according to the selected memory.

These values are programmed into memory during initialization.

**Extended Mode Register Data for Controller0 - DDR2 SDRAM**

Choose the extended mode register settings for the memory device. Settings are restricted to those supported by the controller. For more information consult the memory vendor's data sheet.

<b>DLL Enable</b>	Enable-Normal(0)
<b>Output Drive Strength</b>	Fullstrength(0)

The Virtex-5 DDR2 interface requires that if parallel termination is used at the memory I/Os, then it must be ODT rather than external termination resistor(s). This is a requirement of the read capture scheme in use.

<b>RTT (nominal) - ODT</b>	75ohms(01)
<b>Additive Latency (AL)</b>	0(000)
<b>OCD Operation</b>	OCD Exit(000)
<b>DQS# Enable</b>	Enable(0)
<b>RDQS Enable</b>	Disable(0)
<b>Outputs</b>	Enable(0)

Extended Mode Register Value: 0\_0000\_0000\_0100

UG086\_c1\_22\_022708

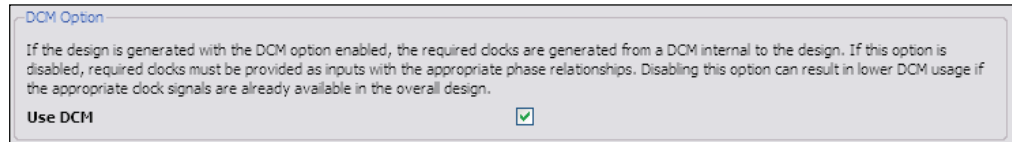
Figure 1-23: Extended Mode Register Data

Click **Next** to move ahead. The FPGA Options window is displayed.

## FPGA Options

This feature is partitioned into five sections: DCM, DCI, DCI Cascading Information, SSTL Class, and Debug Signals Control.

- **DCM.** DCM allows design generation with or without a DCM in the design.



UG086\_c1\_23\_022708

Figure 1-24: DCM Option

- **DCI.** This feature indicates whether the Digitally Controlled Impedance is Disabled or Enabled. DCI can be enabled or disabled for Input/Inout pins or Outputs. This option can change according to the memory selected. They are listed as follows:

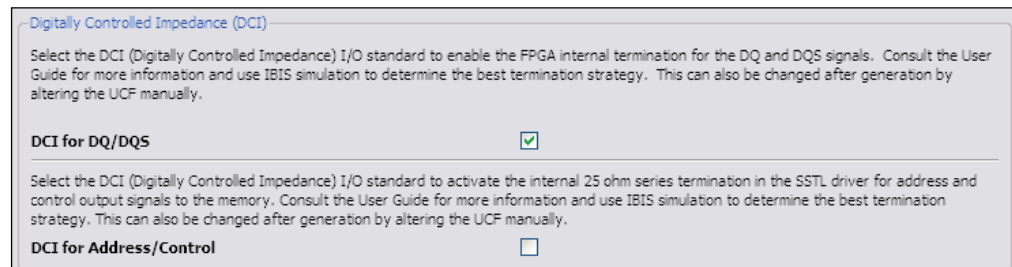
**DDR2 SDRAM** — DCI for DQ/DQS and DCI for Address/Control

**DDR SDRAM** — DCI for DQ/DQS and DCI for Address/Control

**RLDRAM II** — DCI for Data, Read Clocks, and Data Valid Signals and DCI for Address/Control

**QDR II SRAM** — DCI for Data and Read Clocks

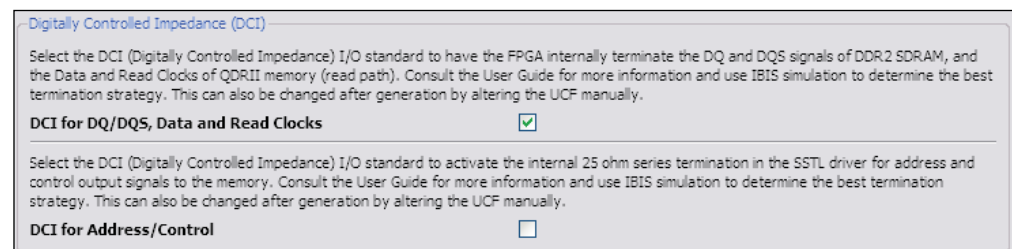
**DDR II SRAM** — DCI for Data and Read Clocks



UG086\_c1\_24\_022708

Figure 1-25: DCI Options

For multiple interfaces in Virtex-5 FPGA designs, when selected, DCI is applied for all the interfaces. The DCI for Address/Control is applicable only for DDR2 SDRAM designs when multiple interfaces are selected.



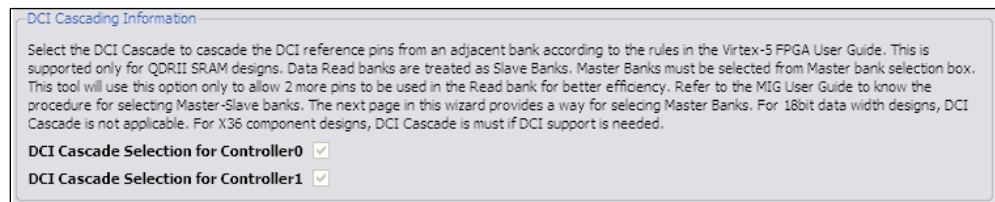
UG086\_c1\_58\_022708

Figure 1-26: DCI Option for Multiple Interfaces Selected in Virtex-5 FPGA Designs

If DCI is enabled, the pins are characterized by the DCI I/O standards.



- **DCI Cascading Information.** This option appears only for QDRII Virtex-5 FPGA designs. This option is necessary for generating 36-bit component designs with DCI support.

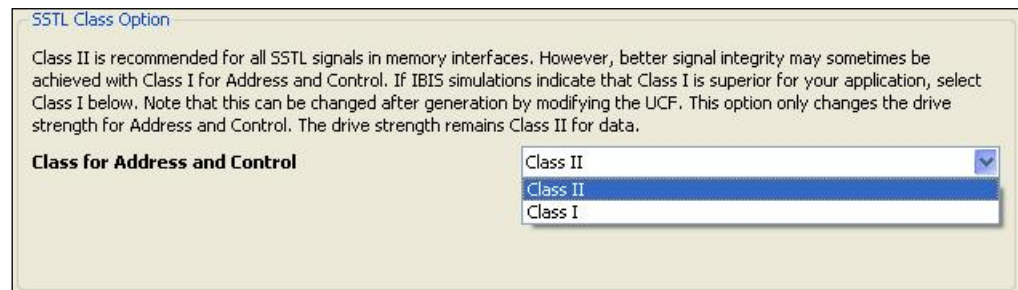


UG086\_c1\_25\_022708

Figure 1-27: DCI Cascading Information Option

**Note:** If the DCI Cascading Information option is checked, the Bank Selection window shows the Master Bank selection box. The user must not reserve VRN/VRP pins in the Reserve Pins window for the selected master banks.

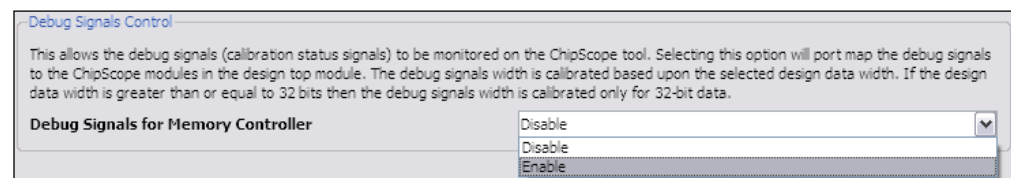
- **SSTL Class Options.** SSTL Class Option determines the I/O standard drive strength in the UCF of DDR SDRAM and DDR2 SDRAM. These I/O standards can be changed based on their application.



UG086\_c1\_26\_083007

Figure 1-28: SSTL Class Options

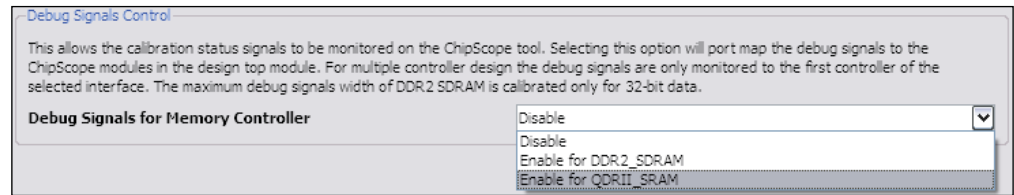
- **Debug Signals Control.** Selecting this option enables the debug signals to be port-mapped to the ChipScope™ modules in the design top module. This helps in monitoring the debug signals on the ChipScope tool. When the generated design is run in batch mode using `ise_flow.bat` in the design's par folder, the CORE Generator system is called to generate ChipScope modules (that is, EDIF files are generated). Deselecting this option leaves the debug signals unconnected in the design top module. No ChipScope modules are instantiated in the design top module, and no ChipScope modules are generated by the CORE Generator system.



UG086\_c1\_44\_022708

Figure 1-29: Debug Signals Control

In Virtex-5 FPGA multiple interface designs, the Debug port is supported for either the DDR2 SDRAM or the QDRII SRAM interface of the first controller.



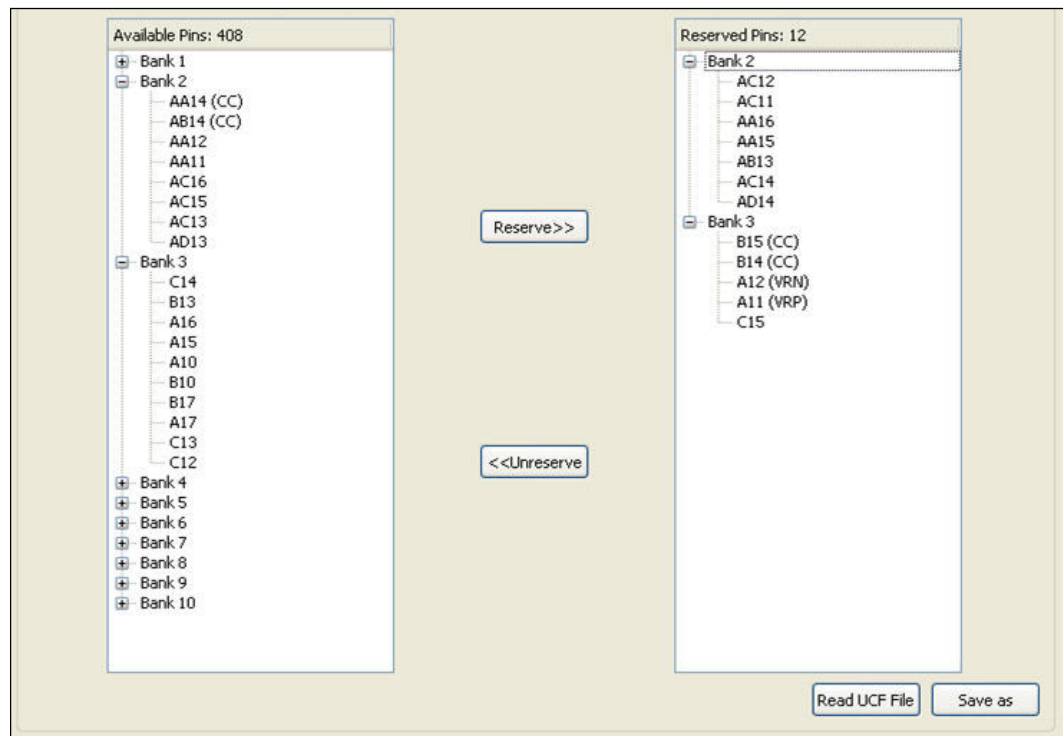
UG086\_c1\_59\_022708

Figure 1-30: Debug Signals in Virtex-5 FPGA Multiple Interface Designs

Click **Next** to move ahead. The Reserve Pins window is displayed.

### Reserve Pins

This feature allows reservation of specific pins for other applications. After selecting suitable pins as necessary, the reserved pins are not used by MIG while generating the pinout for that particular design.



UG086\_c1\_27\_122907

Figure 1-31: Reserve Pins

Select the pins from the Available Pins column, and click the **Reserve** button. The particular pin is transferred to Reserve Pins column along with its bank information. This signifies that the selected pin has been reserved. To unreserve a reserved pin, click the appropriate pin that needs to be removed, and then click the **Unreserve** button. The number 408 in the Available Pins header signifies the number of pins available for pinout, whereas the number 16 in the Reserve Pins header signifies the number of pins selected to be reserved.

The reserved pins information can be saved in a user defined file using the **Save as** button. A browser window appears after clicking the **Save as** button. Set the file location here.

Use the **Read UCF File** button to read a reserve pins from a UCF. When the **Read UCF File** button is clicked, a new window pop ups. Select the UCF to be read. After reserving the pins, click the **Next** button to continue. The Bank Selection window is displayed.

### Bank Selection

This feature allows selection of banks for the Memory interface. Banks can be selected for different classes of memory signals. The different classes are:

- Address and Control Signals
- Data Signals
- System Control Signals
- System Clock

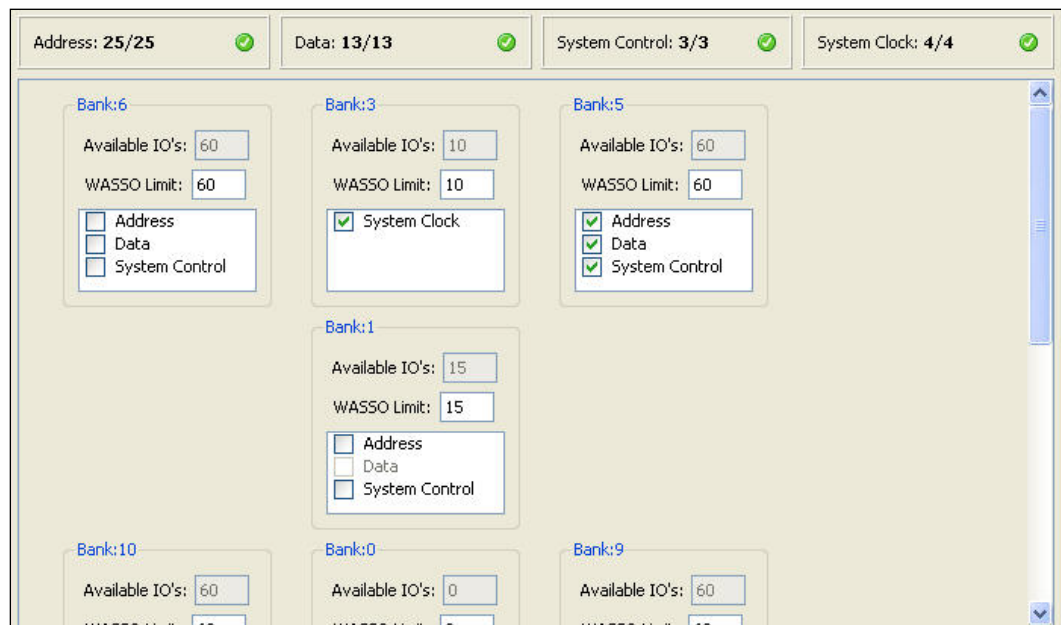
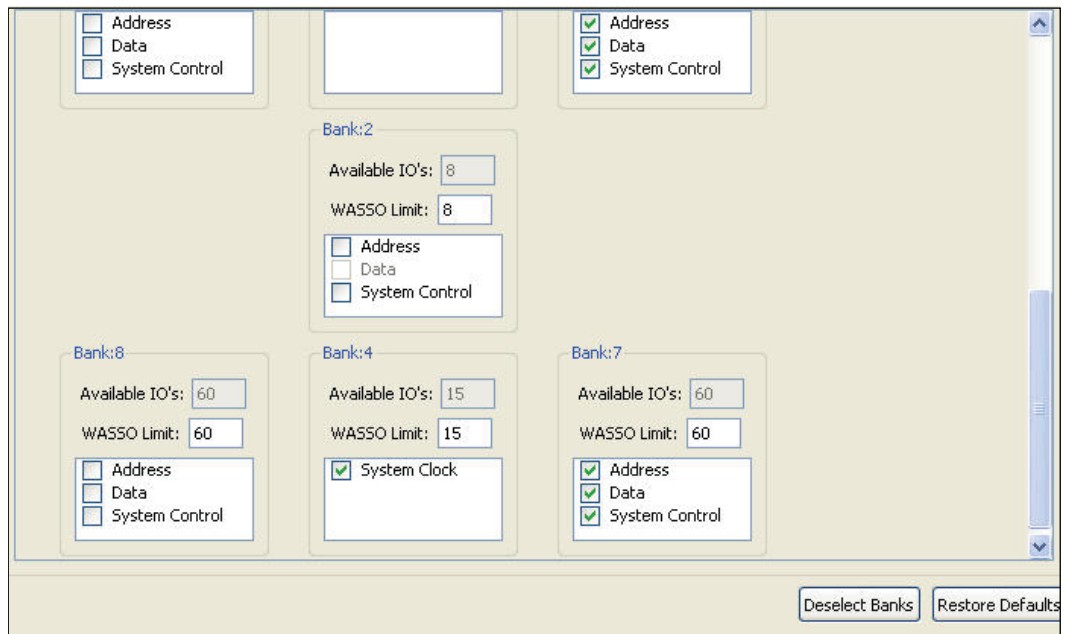


Figure 1-32: Bank Select (a)

UG086\_c1\_28\_122907



UG086\_c1\_29\_122907

Figure 1-33: Bank Select (b)

Select the appropriate bank and memory signals as required.

The WASSO limit in conjunction with the Reserve pins limits the number of available I/Os in a bank. For more information on the WASSO limit, refer [Appendix C, “WASSO Limit Implementation Guidelines.”](#)

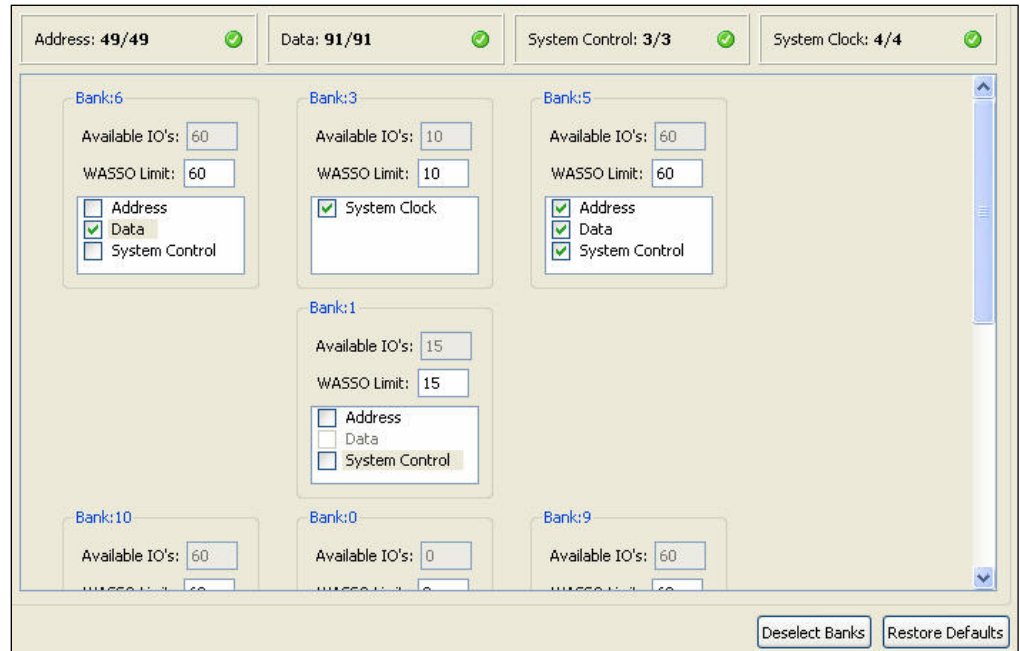
To unselect the banks that are selected, click the **Deselect Banks** button. To restore the defaults, click the **Restore Defaults** button.

In certain banks, global clock pins are not allowed for system clock. This is because system clock signals have different I/O standards as compared to those of any other signals in the design. In such banks, global clock pins are left unused.

- **Real-time pin allocation.** As the user selects the banks, pin allocation is done dynamically, and the number of pins required is updated for each group of signals.
  - ◆ The red circle with a cross mark at each group indicates that sufficient pins are not allocated, and additional pins are required for the selected configuration.
  - ◆ The green circle with a tick mark at each group indicates that sufficient pins are allocated for the selected configuration.
  - ◆ The denominator in each group indicates the total number of pins required for each group.

The user must select banks until the numerator equals the denominator. The user cannot move to the next page unless sufficient pins are allocated for each group.

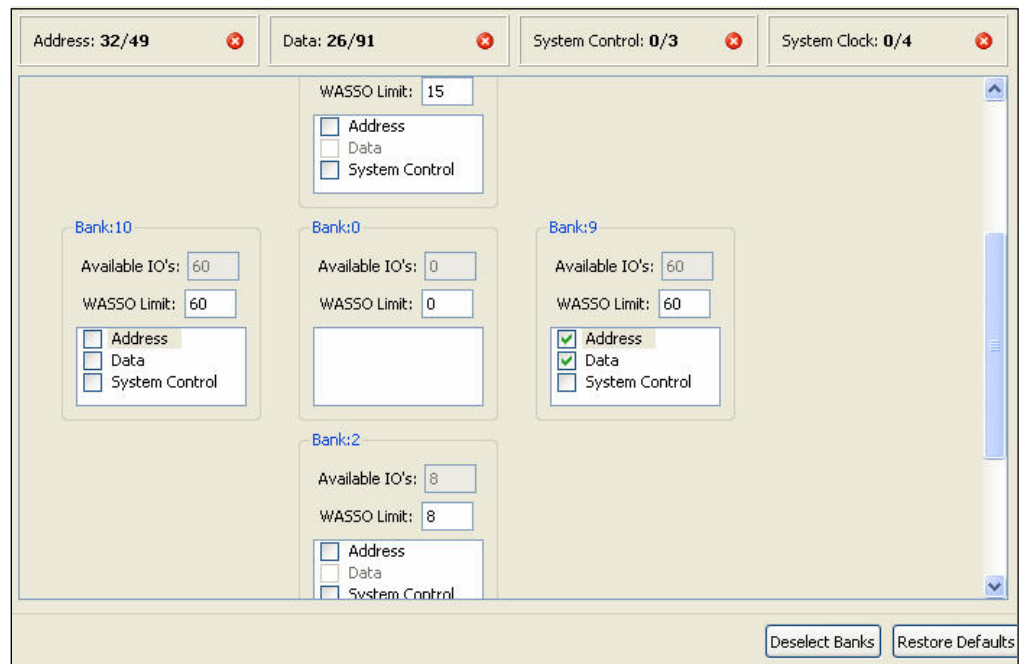
[Figure 1-34](#) illustrates the conditions where sufficient banks are selected in order to successfully generate the design.



UG086\_c1\_45\_122907

Figure 1-34: Real-Time Pin Allocation: Sufficient Banks Selected

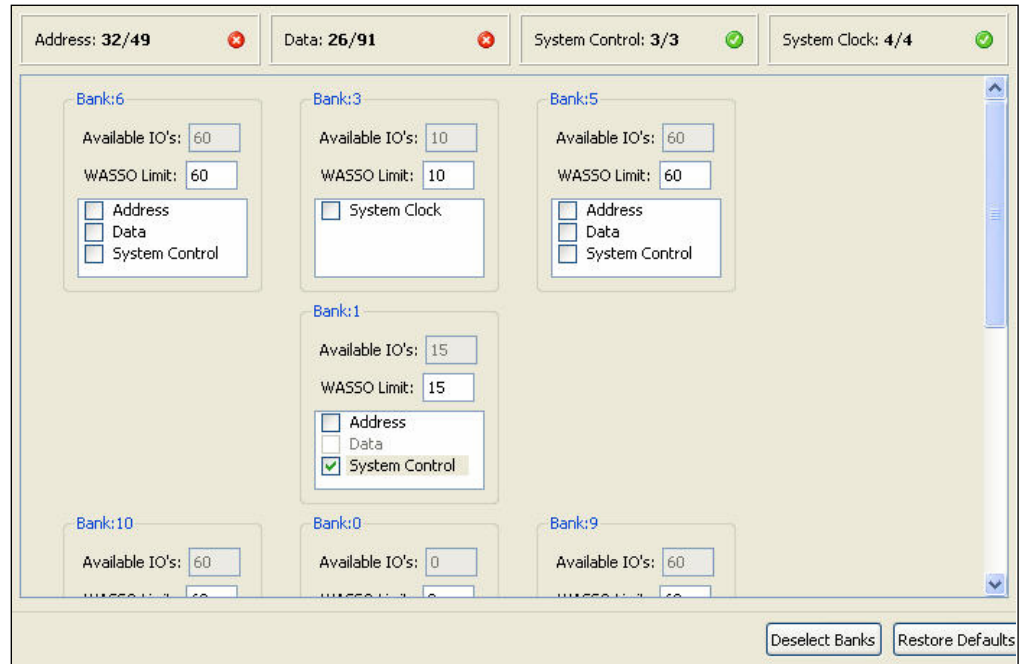
Figure 1-35 indicates when sufficient banks are *not* allocated for each signal group.



UG086\_c1\_46\_122907

Figure 1-35: Real-Time Pin Allocation: Sufficient Banks Not Selected

Figure 1-36 indicates sufficient pins are allocated for System Control and System Clock groups, but sufficient pins are *not* allocated for Data and Address groups.

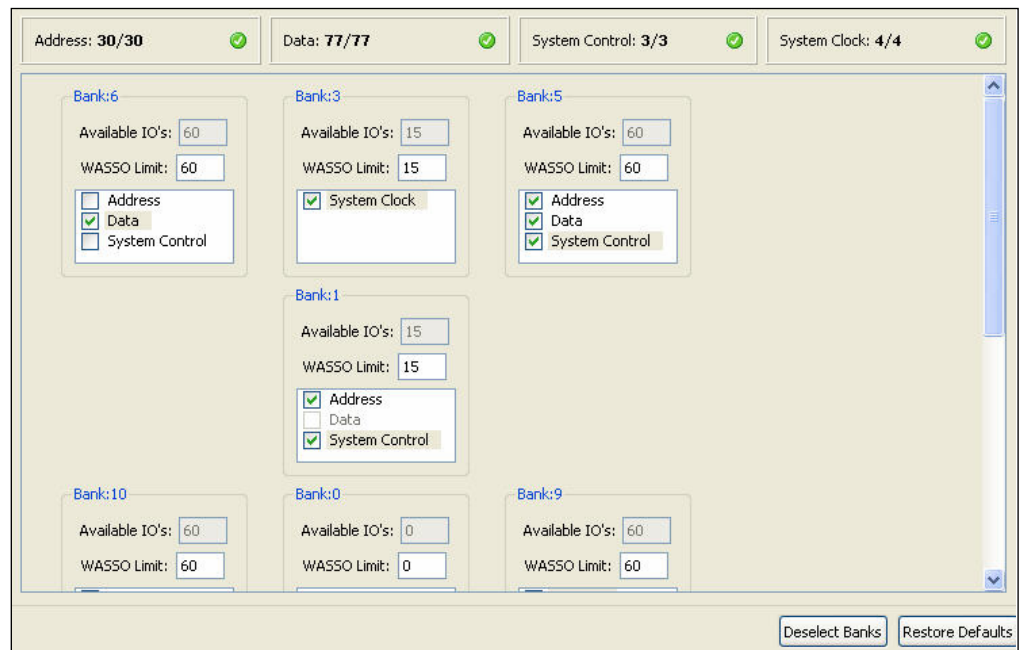


UG086\_c1\_47\_122907

Figure 1-36: Real-Time Pin Allocation: Insufficient Pins for Data/Address Groups

- **Pin Allocation Priority.** MIG allocates the pins starting with exclusive data banks first, followed by data banks that combine with other groups.

Figure 1-37 indicates that data banks are selected in bank 6 and bank 5. In bank 6, only data is selected; in bank 5, data, address, and system control are selected. Here, data is allocated first in bank 6 and then in bank 5. This Pin Allocation Priority is applicable only for data group signals in Virtex-4 and Virtex-5 devices.

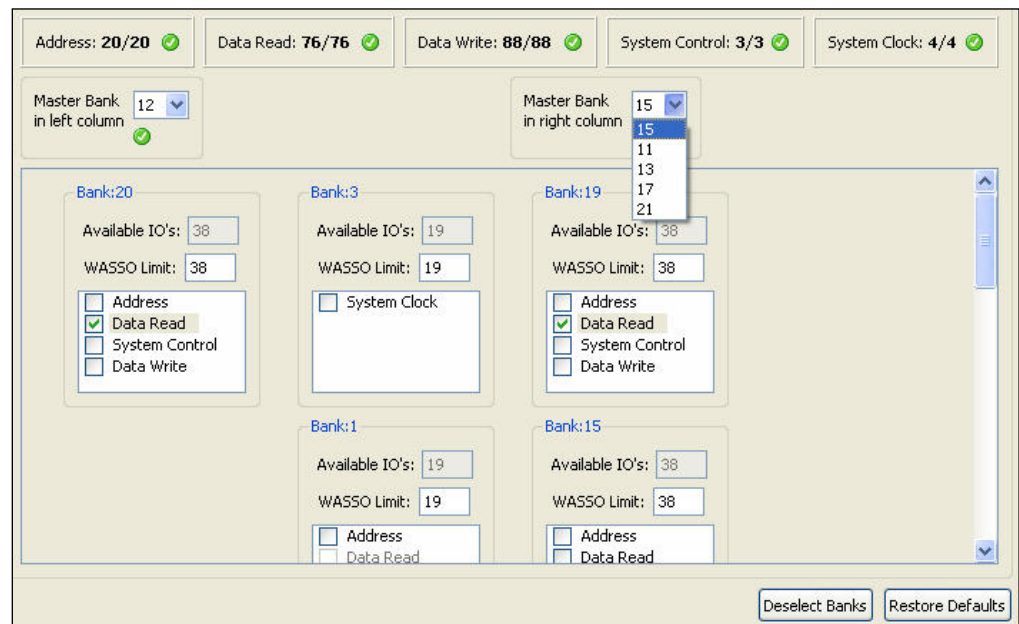


UG086\_c1\_48\_122907

Figure 1-37: Pin Allocation Priority

- Master Bank selection.** This is applicable only for QDRII Virtex-5 FPGA designs when the DCI Cascading Information option is selected. A Master bank should be selected in each column when a Data Read is selected in that particular column. *There is an exception for the middle column.* The middle column is divided into two parts: above zero bank and below zero bank. The middle column can have two Master banks, depending on where the Read Data banks are selected. If the Read Data bank is selected *either above or below the Zero bank*, only one Master Bank is required. If the Read Data banks are selected *both above and below Zero bank*, two Master banks are required.

Figure 1-38 shows that the Data Read is selected in both the columns and user needs to select the Master Banks in both the columns. Master bank combo box lists all the possible banks that can be selected as Master Bank. MIG does not show the Master Bank selection check box for a column if that column does not have enough pins in the banks.

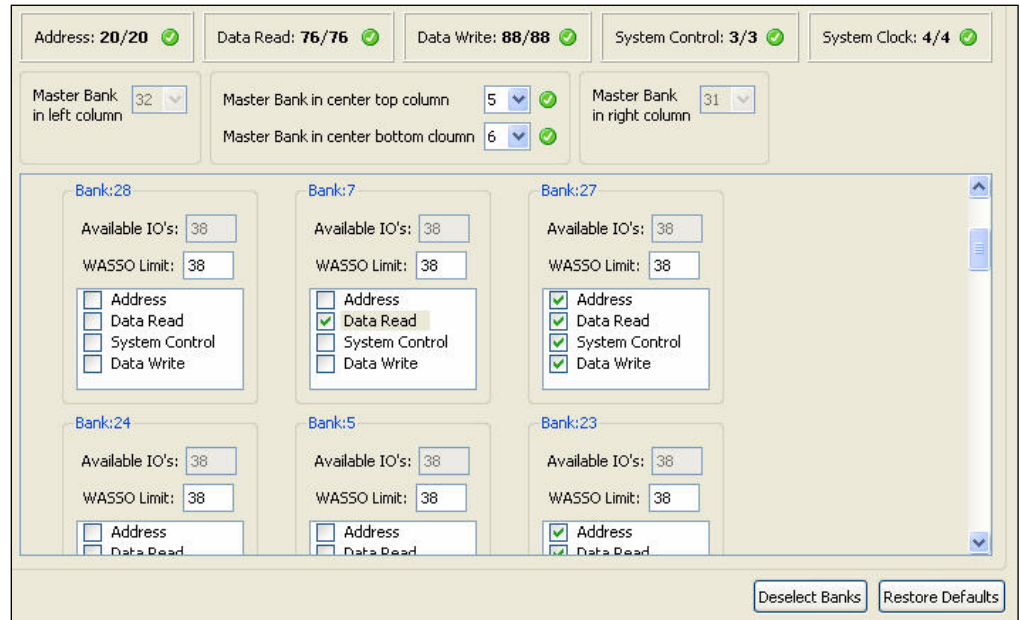


UG086\_c1\_49\_122907

Figure 1-38: Master Bank Selection (a)



Figure 1-39 shows the Master Bank selection in the center column. It uses all the pins for Read Data from the center column.



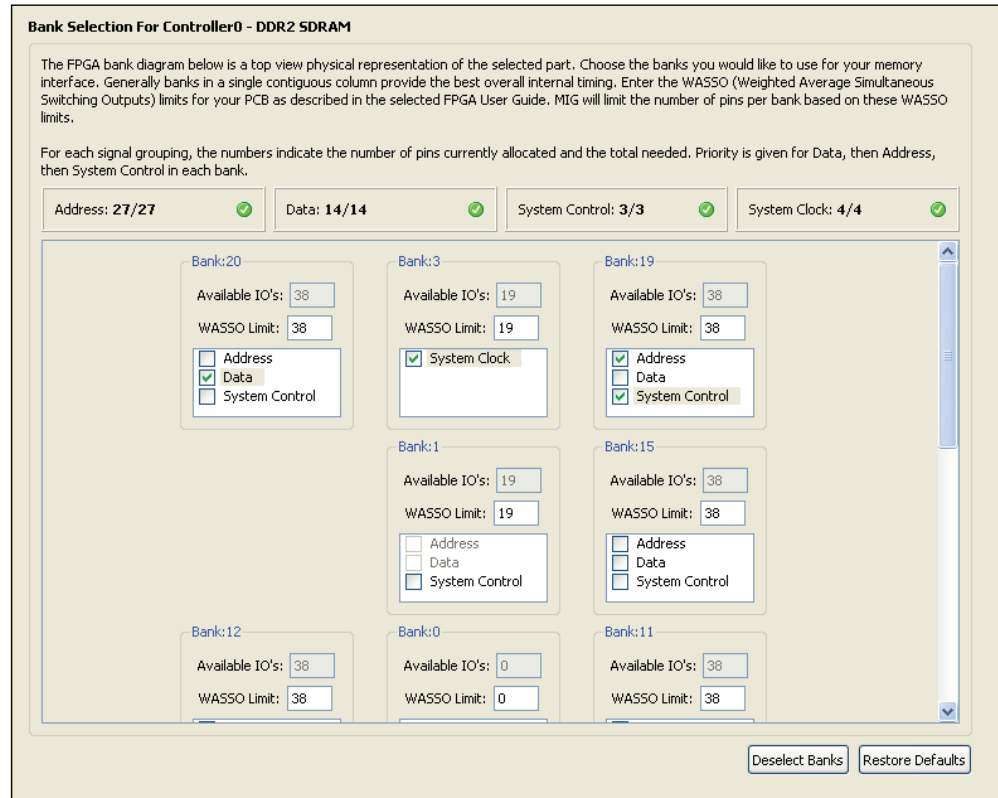
UG086\_c1\_50\_122907

Figure 1-39: Master Bank Selection (b)

After the selection of the banks, click the **Next** button to move ahead. The Memory Model License window is displayed.

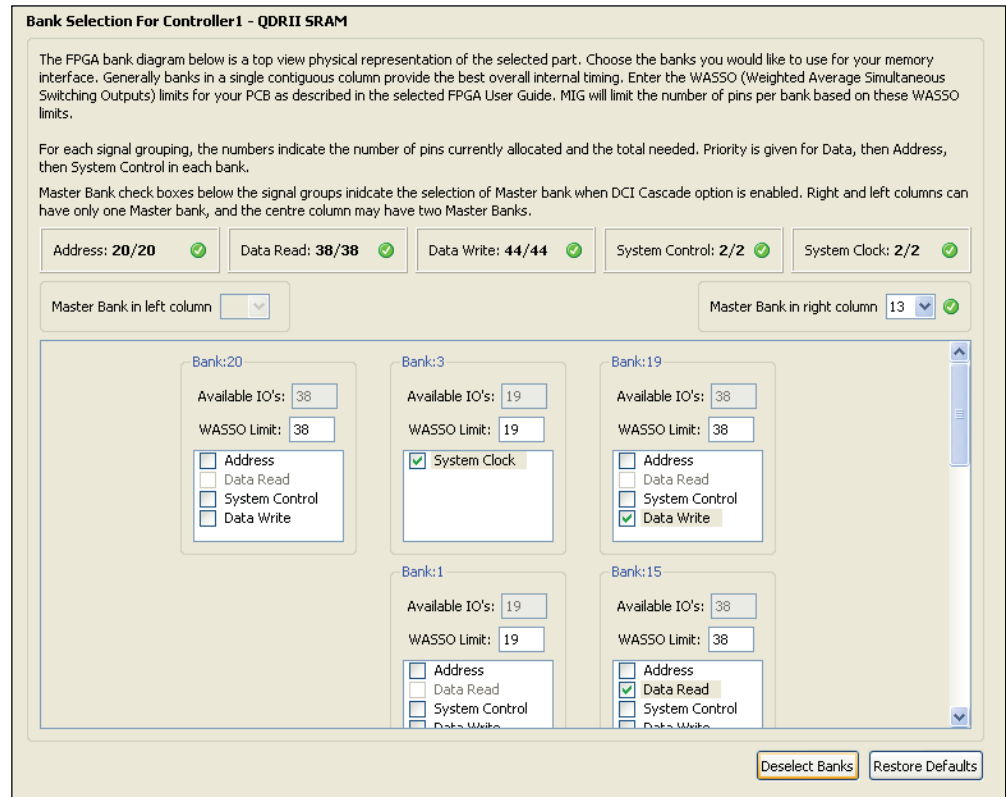


- Bank Selections for Multiple Memory Interfaces in Virtex-5 FPGA Designs.** For a multiple interface design, a particular group is allowed to select in a bank only for compatible I/O standards. For example, Controller 0 is DDR2 SDRAM (see [Figure 1-40](#)) and Controller 1 is QDRII SRAM (see [Figure 1-41](#)). In DDR2 SDRAM, bank 20 is checked for Data and Bank 19 is checked for Address and System Control. In QDRII SRAM, neither bank 20 nor bank 19 is allowed to select Data Read, because the I/O standard for DDR2 SDRAM Data and Address is SSTL18\_II\_DCI, and the I/O standard for QDRII SRAM Data Read is HSTL\_I\_DCI\_18. These two I/O standards are not compatible. Hence MIG does not allow bank selection for the group of signals that do not follow the I/O standard compatibility rules.



UG086\_c1\_60\_022108

Figure 1-40: DDR2 SDRAM Bank Selection in a Multiple Interface Design

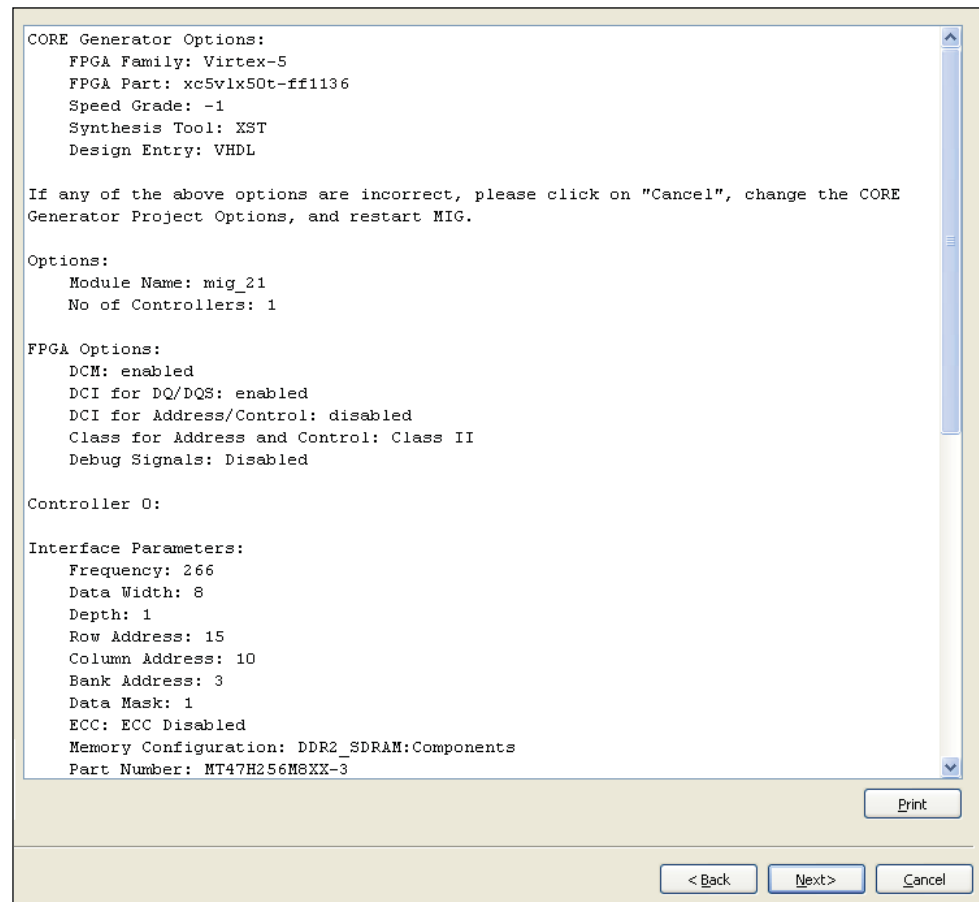


UG086\_c1\_61\_022108

Figure 1-41: QDRII SRAM Bank Selection in a Multiple Interface Design

## Summary

This window provides complete details about the bank selection, Interface parameters, CORE Generator options, and FPGA options of the active project.



UG086\_c1\_31\_022108

Figure 1-42: Summary

Click the **Next** button to move to the License Agreement page of the selected memory of the Micron memory model.

## Memory Model License

MIG outputs a Micron memory model for simulation purposes for memories such as DDR SDRAM, DDR2 SDRAM, and RLDRAM II. To access the models in the output sim folder, click the **Micron License Agreement** check box. Read the License Agreement carefully and mark the **Accept License Agreement** check box to accept it.

If the License Agreement is not agreed to, the memory model is not available. The user then needs to get the appropriate memory model by some other means to simulate the design.

Micron Technology, Inc. Simulation Model License Agreement

PLEASE READ THIS SIMULATION MODEL LICENSE AGREEMENT ("AGREEMENT") FROM MICRON TECHNOLOGY, INC. ("MTI") CAREFULLY BEFORE INSTALLING OR USING THIS SIMULATION MODEL (THE "MODEL"). BY INSTALLING OR USING THE MODEL, YOU ARE ACCEPTING AND AGREEING TO THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE WITH THE TERMS AND CONDITIONS OF THIS AGREEMENT, THEN DO NOT INSTALL OR USE THE MODEL.

**SOFTWARE LICENSE:** You acknowledge and agree that it is your sole responsibility to obtain the appropriate license or permission from the owner(s) of the software platform(s) that are necessary for you to operate the Model. MTI is under no obligation whatsoever to offer, provide or secure such license or permission for you.

**MODEL LICENSE:** MTI hereby grants to you the right to install, use and modify the Model solely for testing the Model and designing your product(s) in connection with the Model. You shall not use the Model or any modifications for any other purpose, and shall not copy, rent, or lease the Model or the modifications to any third party. MTI may make changes to the Model at any time without notice to you. MTI is under no obligation whatsoever to update, maintain, or provide new versions or other support for the Model.

**OWNERSHIP OF MATERIALS:** You acknowledge and agree that the Model is proprietary property of MTI and is protected by United States copyright law and international treaty provisions. The Model may not be copied, reproduced, published, uploaded, posted, transmitted, or distributed in any way without MTI's prior written permission. Except as expressly provided herein, MTI does not grant any express or implied right to you under any patents, copyrights, trademarks, or trade secret information. This Agreement does not convey to you an interest in or to the Model, but only a limited right to use and modify the Model in accordance with the terms of this Agreement.

**DISCLAIMER OF WARRANTY:** THE MODEL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. MTI EXPRESSLY DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, NONINFRINGEMENT OF THIRD PARTY RIGHTS, AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR

**Accept License Agreement (If you do not accept this agreement, the simulation output directory will not contain the memory model so you will need to acquire and configure a memory model appropriately).**

Print

< Back Generate Cancel

UG086\_c1\_30\_022108

Figure 1-43: License Agreement

Click the **Generate** button to generate the design files. MIG generates two output directories `example_design` and `user_design`. After generating the design, a new window called PCB Information page is displayed.

## PCB Information

This page displays the PCB related information to be considered while designing the board that uses MIG generated designs. Click **Next** to go to the Finish page.

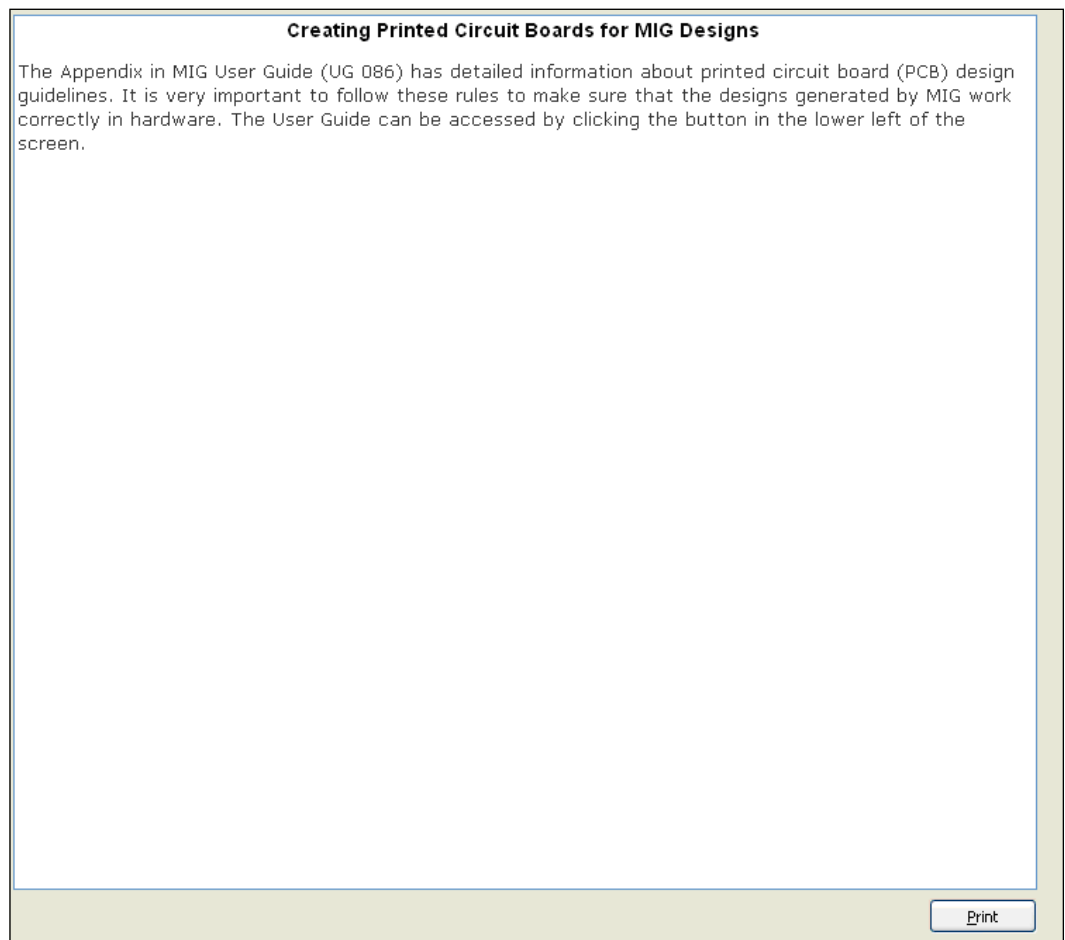
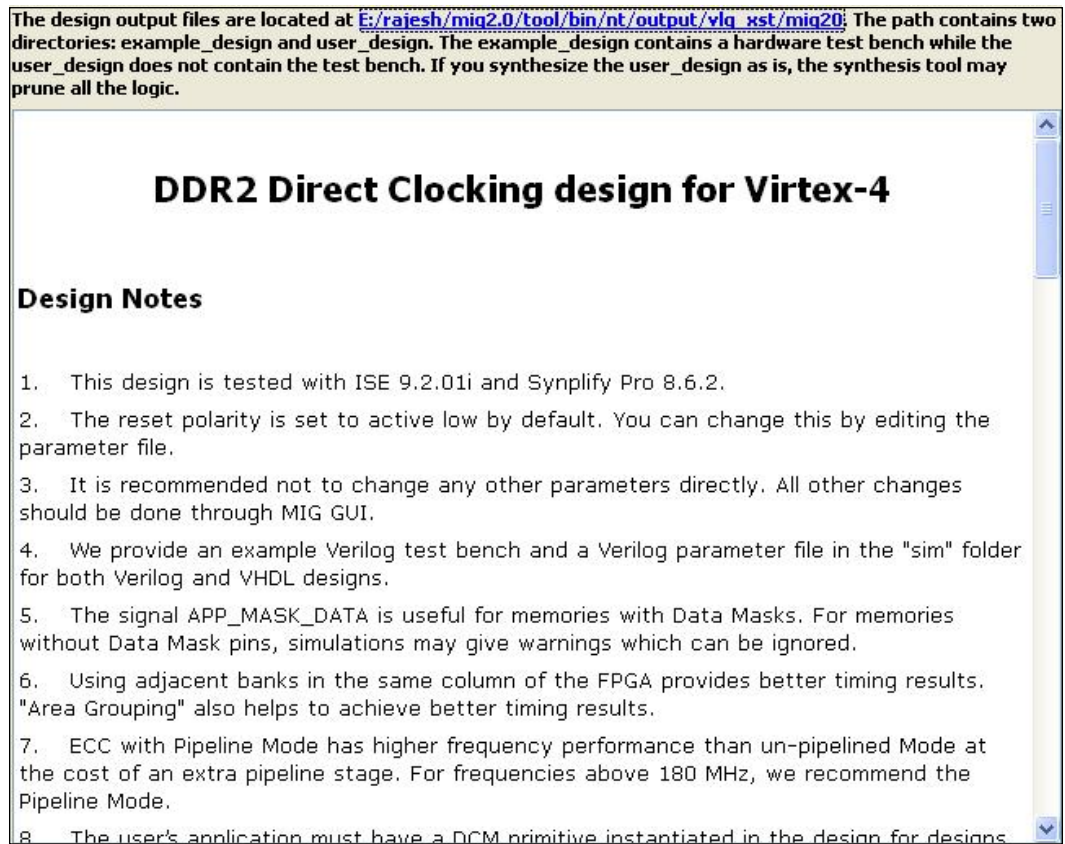


Figure 1-44: PCB Information

UG086\_c1\_42\_090407

## Finish

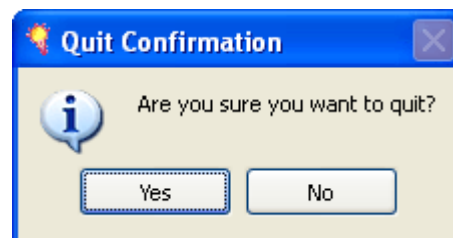
This window shows if the design was generated successfully. This page provides the design notes that should be taken into account while using MIG generated designs.



UG086\_c1\_32\_083007

Figure 1-45: Finish

The text in the blue color indicates the path of the design output files. Click the blue text to go through the output files. Click the **Finish** button to quit. The Quit Confirmation window shown in Figure 1-46 appears.



UG086\_c1\_33\_091307

Figure 1-46: Quit Confirmation

Click **Yes** to exit or **No** to return to the Finish page.

## Output Files

A MIG-generated design has the following output files and directory:

- A *<component name>\_xmdf.tcl* file, used for the CORE Generator application.
- A *<component name>.vho* file, used for the core to be instantiated, created only when a VHDL design is generated.
- A *<component name>.veo* file, used for the core to be instantiated, created only when a Verilog design is generated.
- A *<component name>* directory.

In the *<component name>* directory, three folders are created:

- docs
- example\_design
- user\_design

Any relevant documents, such as application notes, timing analysis spreadsheets, and user guide are in the docs directory.

The example\_design and user\_design folders contain several other folders and files. They are:

- rtl — Contains all the RTL files (either VHDL or Verilog design files).
- par — Contains the UCF with constraints for the design. Two scripts files are generated:
  - ♦ ise\_flow.bat — The user double-clicks the ise\_flow.bat script file to run the design through synthesis, build, map, and par. This script file sets all the required options. Users should refer to this file for the recommended build options for the design.
  - ♦ create\_ise.bat — The user double-clicks the create\_ise.bat file to create an ISE project. The ISE project thus generated contains the recommended build options for the design. To run the project in GUI mode, the user double-clicks the ISE project file to open up ISE in GUI mode with all project settings.
- synth — Contains the SDC file which has design constraints for Synplify Pro synthesis tool. 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 testbench files that are needed to simulate the design. It also has an executable and a .do file. If sim.exe is double-clicked, the design is automatically simulated using the ModelSim simulator.

There is a simulation\_help.chm file in the sim folder that helps you to understand the simulation environment provided. For the user\_design folder, a synthesizable testbench module is also present in the sim folder.

**Caution! Recommended Build Options.** The ise\_flow.bat file in the par folder of the component name directory contains the recommended build options for the design. Failure to follow the recommended build options could produce unexpected results.

## Create Design for Xilinx Reference Boards

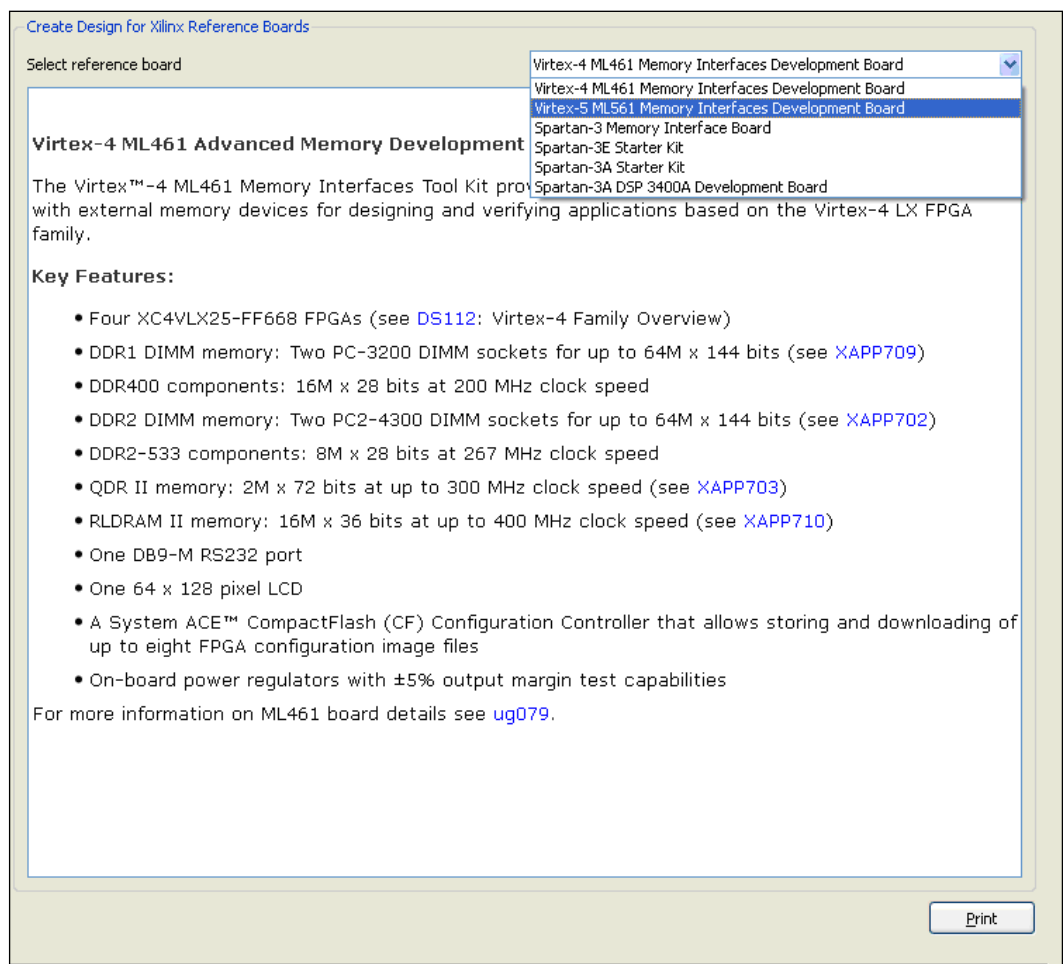
To create a design for the Xilinx Reference Boards, select **Create Design for Xilinx Reference Boards** from the MIG Output Options. It is intended to generate the board files for various Xilinx Reference Boards. Click the **Next** button to move ahead.

The flow is as follows:

1. [Reference Board Designs](#)
2. [Memory Model License](#)
3. [PCB Information](#)
4. [Finish](#)

### Reference Board Designs

This section allows selection of the board for which the designs are to be generated.



UG086\_c1\_34\_090407

Figure 1-47: Create Design for Xilinx Reference Boards

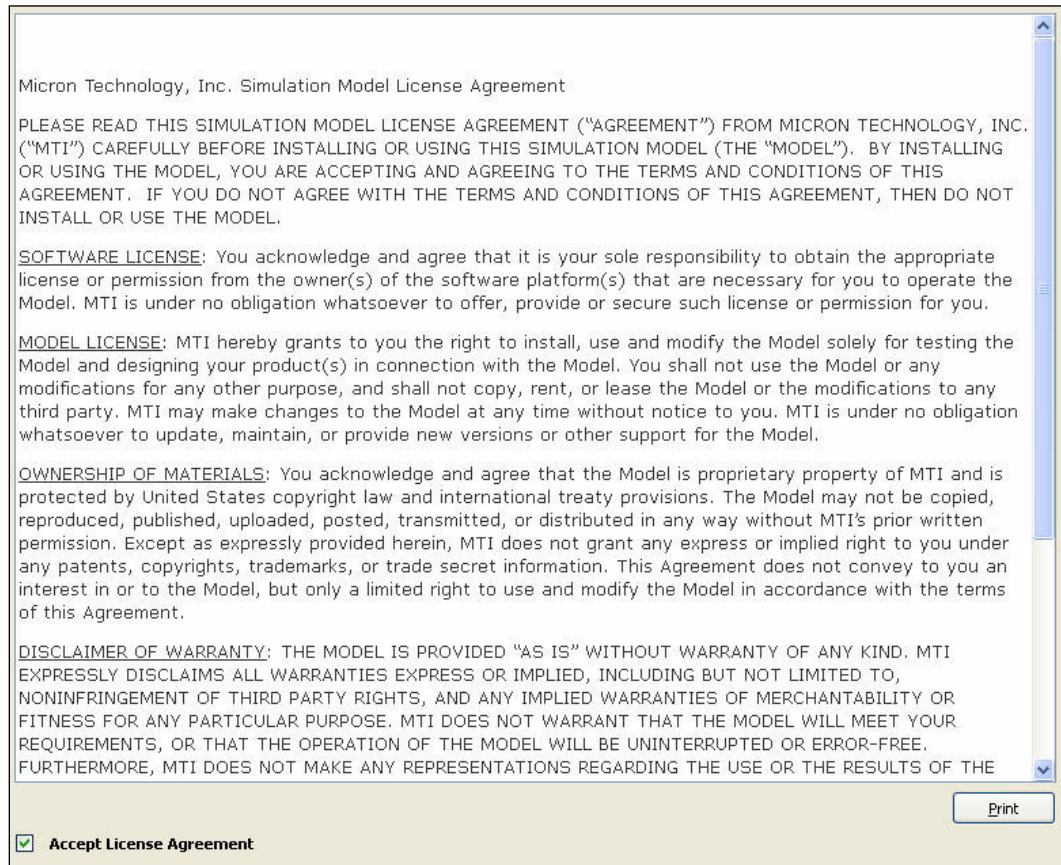
The pull-down menu includes a list of boards. Select the appropriate board. Details about the particular board are displayed in the pane below. After selecting the board, click **Next** to move to next page.



## Memory Model License

MIG outputs a Micron memory model for simulation purpose for memories like DDR SDRAM, DDR2 SDRAM, and RLDRAM II. To generate the board files for the specified Xilinx Reference Board, read the License Agreement carefully and mark the **Accept License Agreement** checkbox to accept it.

If the License Agreement is not accepted, the user cannot generate board files. The **Next** button is disabled unless the License Agreement is accepted.



Micron Technology, Inc. Simulation Model License Agreement

PLEASE READ THIS SIMULATION MODEL LICENSE AGREEMENT ("AGREEMENT") FROM MICRON TECHNOLOGY, INC. ("MTI") CAREFULLY BEFORE INSTALLING OR USING THIS SIMULATION MODEL (THE "MODEL"). BY INSTALLING OR USING THE MODEL, YOU ARE ACCEPTING AND AGREEING TO THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE WITH THE TERMS AND CONDITIONS OF THIS AGREEMENT, THEN DO NOT INSTALL OR USE THE MODEL.

**SOFTWARE LICENSE:** You acknowledge and agree that it is your sole responsibility to obtain the appropriate license or permission from the owner(s) of the software platform(s) that are necessary for you to operate the Model. MTI is under no obligation whatsoever to offer, provide or secure such license or permission for you.

**MODEL LICENSE:** MTI hereby grants to you the right to install, use and modify the Model solely for testing the Model and designing your product(s) in connection with the Model. You shall not use the Model or any modifications for any other purpose, and shall not copy, rent, or lease the Model or the modifications to any third party. MTI may make changes to the Model at any time without notice to you. MTI is under no obligation whatsoever to update, maintain, or provide new versions or other support for the Model.

**OWNERSHIP OF MATERIALS:** You acknowledge and agree that the Model is proprietary property of MTI and is protected by United States copyright law and international treaty provisions. The Model may not be copied, reproduced, published, uploaded, posted, transmitted, or distributed in any way without MTI's prior written permission. Except as expressly provided herein, MTI does not grant any express or implied right to you under any patents, copyrights, trademarks, or trade secret information. This Agreement does not convey to you an interest in or to the Model, but only a limited right to use and modify the Model in accordance with the terms of this Agreement.

**DISCLAIMER OF WARRANTY:** THE MODEL IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. MTI EXPRESSLY DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, NONINFRINGEMENT OF THIRD PARTY RIGHTS, AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. MTI DOES NOT WARRANT THAT THE MODEL WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE MODEL WILL BE UNINTERRUPTED OR ERROR-FREE. FURTHERMORE, MTI DOES NOT MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE

**Accept License Agreement** Print

UG086\_c1\_54\_010108

*Figure 1-48: Memory Model License*

After accepting the agreement, click **Generate** to generate the board files for the specified Xilinx Reference Board. After the successful generation of board files, the PCB Information page is displayed.

## PCB Information

This page displays the PCB-related information to be considered while designing the board that is to use a MIG generated design. Click **Next** to go to the Finish page.



Figure 1-49: PCB Information

UG086\_c1\_51\_123007

Clicking **Next** displays the Finish page.

## Finish

The blue text above the pane shows the path of the output folder. Click the **Back** button to choose a different board. Click the **Close** button to terminate.

Completed. Look at the folder [E:/mig\\_2\\_0\\_test/Wizard\\_tool/DDR\\_V4/verilog/tool\\_test](E:/mig_2_0_test/Wizard_tool/DDR_V4/verilog/tool_test) for board files.

### Virtex-4 ML461 Memory Interface Development Board Files

Following are the bit files, RTL, and UCF files provided for ML461 Memory Interface Development Board designs.

#### DDR2 SDRAM Direct Clocking

S.No.	Data Width	HDL	BL <sup>1</sup> , CL <sup>2</sup> , AL <sup>3</sup>	Synthesis Tool	Memory Part
1	8	Verilog	8, 4, 2	XST	Component "MT47H32M16XX-3"
2	72	VHDL	4, 4, 0	XST	Registered DIMM "MT9HTF6472XX-667"

#### DDR2 SDRAM SERDES Clocking

S.No.	Data Width	HDL	BL <sup>1</sup> , CL <sup>2</sup> , AL <sup>3</sup>	Synthesis Tool	Memory Part
1	8	Verilog	8, 5, 2	XST	Component "MT47H32M16XX-3"
2	72	VHDL	4, 4, 0	XST	Registered DIMM "MT9HTF6472XX-667"

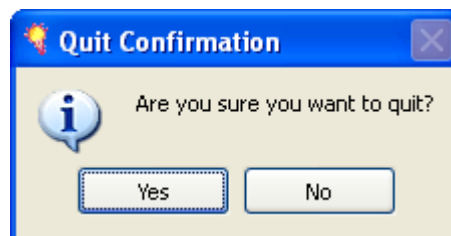
#### DDR SDRAM

Print

UG086\_c1\_35\_083007

Figure 1-50: Finish

Click **Yes** to exit or **No** to return to the Finish page.



UG086\_c1\_33\_091307

Figure 1-51: Quit Confirmation

**Note:** In order to run the simulations in batch mode using ModelSim for the board design files, the `sim.exe` file must be copied to the respective board design's `sim` folder. The `sim.exe` file is provided in the `simulation_executable` folder.

## Verify UCF/Update Design

To verify and update the user constraints file (UCF), select the third option (Verify UCF/Update Design) from the MIG Output Options page. Verify UCF is intended for verification of UCF files that are generated from MIG and later modified. Update Design is intended to update the old UCF files to be compatible to the current RTL design. This feature ensures that the pinout still follows the rules required for the generated design.

Click the **Next** button to move ahead.

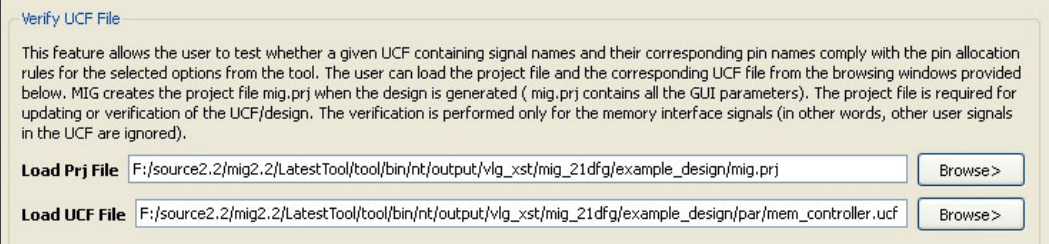
The flow is as follows:

1. [Verify UCF File](#)
2. [Summary](#)
3. [Update Design](#)
4. [Memory Model License](#)
5. [Verification Report](#)
6. [PCB Information](#)
7. [Finish](#)

### Verify UCF File

Provide the input UCF path at the **Load UCF File** box and input the project file path (`mig.prj`) at the Load Prj File Box, or click the **Browse** button to enter the UCF and Prj files through a browser window.

**Note:** Update Design is not supported for the UCF signal names that were modified using the Edit Signal Names option of MIG 1.73.



**Verify UCF File**

This feature allows the user to test whether a given UCF containing signal names and their corresponding pin names comply with the pin allocation rules for the selected options from the tool. The user can load the project file and the corresponding UCF file from the browsing windows provided below. MIG creates the project file `mig.prj` when the design is generated ( `mig.prj` contains all the GUI parameters). The project file is required for updating or verification of the UCF/design. The verification is performed only for the memory interface signals (in other words, other user signals in the UCF are ignored).

**Load Prj File**

**Load UCF File**

UG086\_c1\_37\_022108

Figure 1-52: Verify UCF File

Select the appropriate files. After selecting the files, click **Next** to move ahead.

## Summary

This page provides complete details about the bank selection, Interface parameters, CORE Generator options and FPGA options of the project for which the UCF is to be verified.

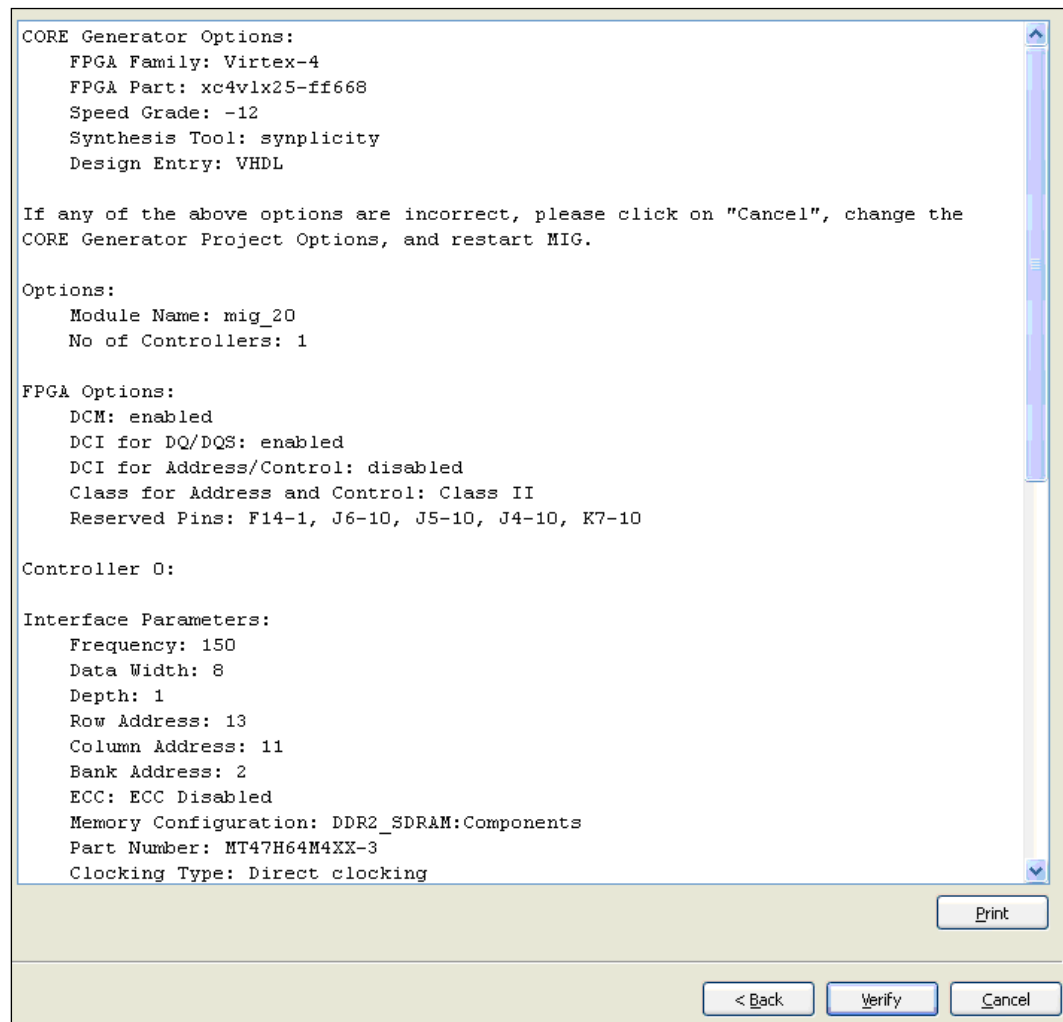


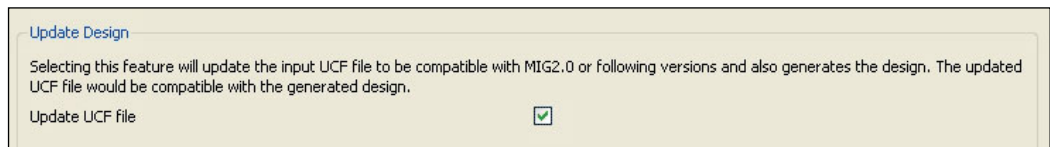
Figure 1-53: Summary Page

UG086\_ct\_38\_091707

Click the **Verify** button to generate the verification report file. After verification, the Update Design page is displayed if the loaded UCF does not contain any required constraints or any changes required to be compatible with the current design. If the loaded UCF is compatible with the current design, the Finish page is displayed.

## Update Design

If the user selects **Update UCF file** and clicks the **Next** button, the License agreement page appears for Micron parts. The Finish page appears for other memory parts.



UG086\_c1\_52\_022108

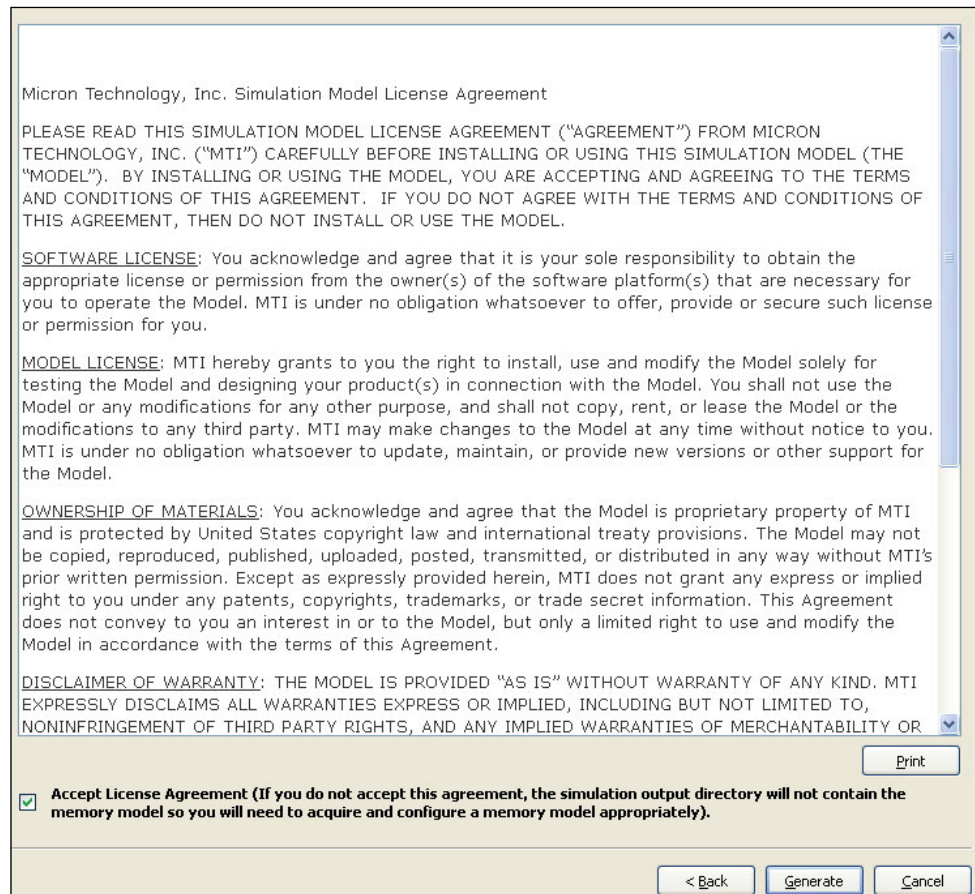
Figure 1-54: Update Design

Click **Next** to move ahead.

## Memory Model License

MIG outputs a Micron memory model for simulation purposes for memories such as DDR SDRAM, DDR2 SDRAM, and RLDRAM II. To get the simulation model in the output folder, click the **Micron License Agreement** check box. Read the License Agreement carefully and mark the **Accept License Agreement** check box to accept it.

If the License Agreement is not agreed to, the simulation model is not output into the output folder.



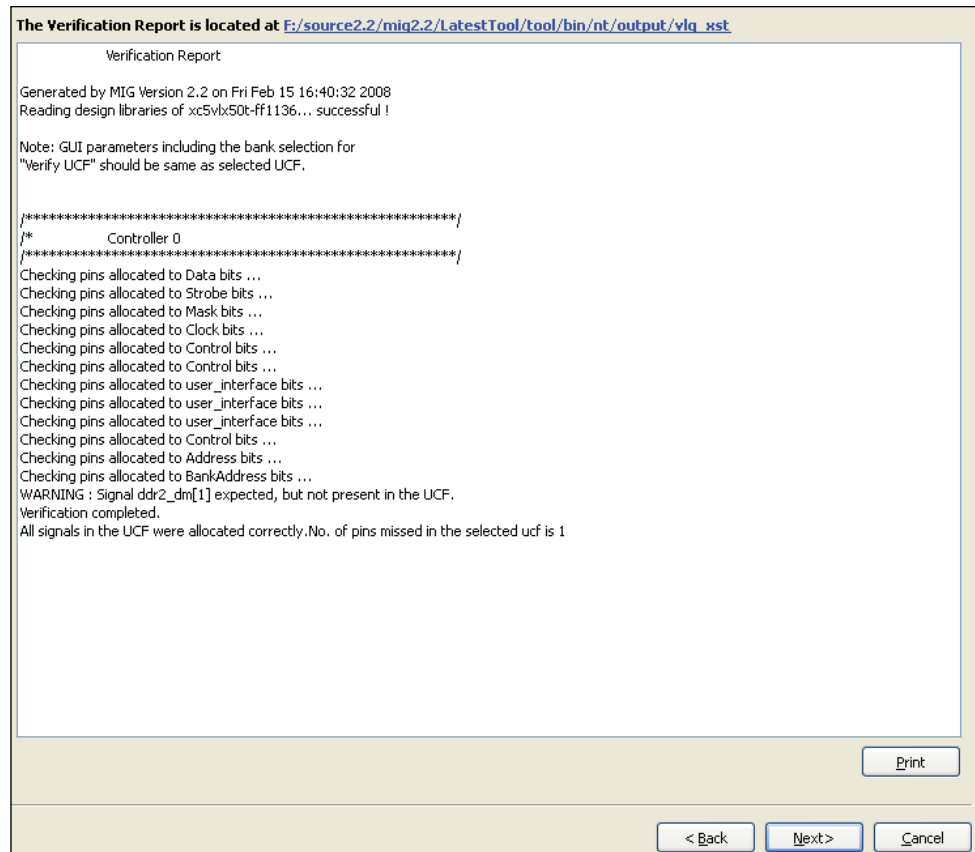
UG086\_c1\_62\_022108

Figure 1-55: Micron License Agreement

Click the **Generate** button to generate the complete design with the loaded Prj settings and modified UCF (the UCF is updated without affecting the pin location constraints) in the updated\_ucf folder.

## Verification Report

This window indicates if the loaded UCF has been verified successfully or provides warnings and errors if the loaded UCF does not follow the pin allocation rules.



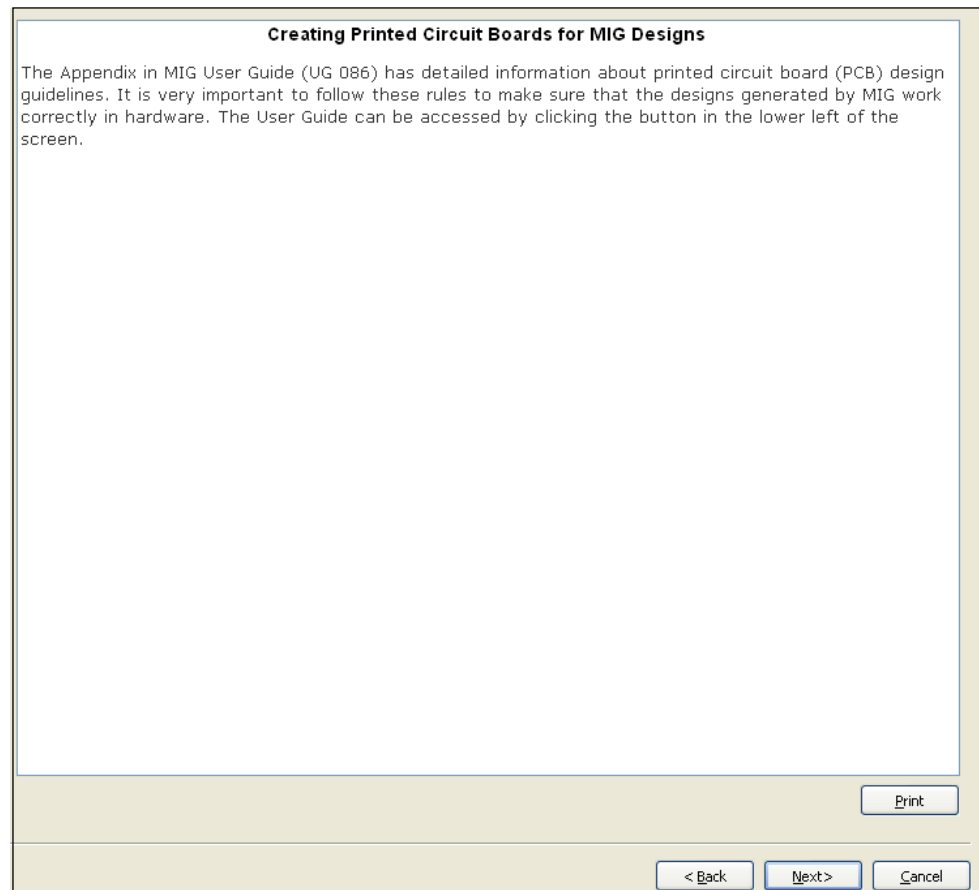
UG086\_c1\_63\_022108

Figure 1-56: Verification Report

Click the **Next** button to move to the PCB information page.

## PCB Information

This page displays the PCB related information to be considered while designing the board that uses MIG generated designs. Click **Next** to go to the Finish page.



UG086\_c1\_64\_022108

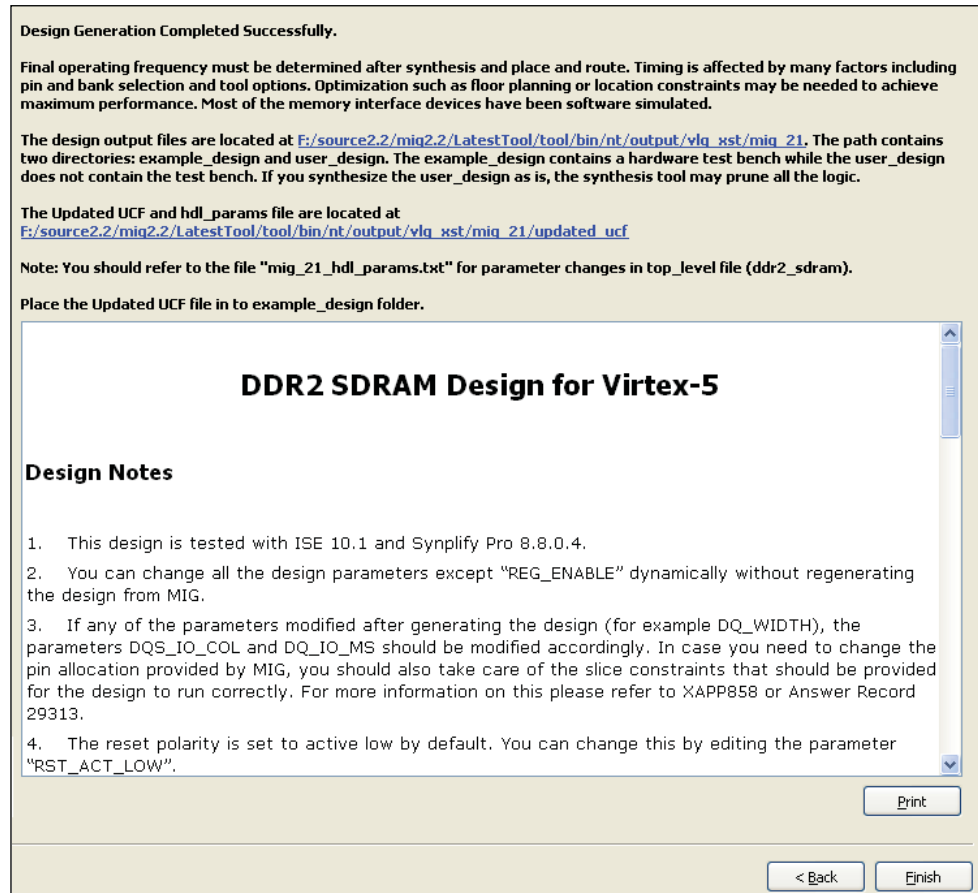
Figure 1-57: PCB Information



## Finish

This window shows if the design was generated successfully. This page provides the design notes that should be taken into account while using MIG generated designs.

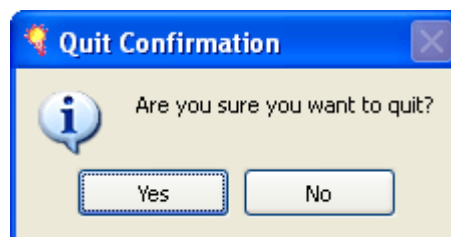
The text in the blue color indicates the path of the design output files. Click the blue text to go through the output files. Click the **Finish** button to quit.



UG086\_c1\_65\_022108

Figure 1-58: Finish

Click **Yes** to exit or **No** to return to the Finish page.



UG086\_c1\_33\_091307

Figure 1-59: Quit Confirmation

## Features Verified

Features verified using Verify UCF option are described as follows:

- Whether all the data bits are allocated in the selected banks.
- The associated groups are allocated in the same bank. For example, data bits corresponding to a DQS in SDRAMs are treated as a group, and data read bits corresponding to a CQ in QDRII SRAMs are treated as a group. All the signals within the same group should be in the same bank.
- 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.

The reference UCF must follow the MIG naming conventions (refer to the UCF generated by MIG). For example, the Virtex-4 FPGA 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.

- **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 <I> are allocated to same pins.”

For example, if `cntrl0_ddr2_dq[0]` and `cntrl0_ddr2_dqs[0]` are allocated to 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.
```

- **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. This Association rule is not applied for data write bits in SIO components.

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.

- **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.
```

- **Absence of signals.** If one or more signal-pin pair is missing and/or commented in the given UCF 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 against the selected inputs."

For example, assume the reference UCF has 8 bits (dq[0:7]), and the data width passed through PRJ is 16 bits. While checking, MIG verifies only 8 bits and reports the other expected bits as follows:

```
WARNING : cntrl0_ddr2_dq[8] is expected, but not present in the UCF.
```

```
WARNING : cntrl0_ddr2_dq[9] is expected, but not present in the UCF.
```

```
WARNING : cntrl0_ddr2_dq[10] is expected, but not present in the
UCF.
```

```
WARNING : cntrl0_ddr2_dq[11] is expected, but not present in the
UCF.
```

```
WARNING : cntrl0_ddr2_dq[12] is expected, but not present in the
UCF.
```

```
WARNING : cntrl0_ddr2_dq[13] is expected, but not present in the
UCF.
```

```
WARNING : cntrl0_ddr2_dq[14] is expected, but not present in the
UCF.
```

```
WARNING : cntrl0_ddr2_dq[15] is expected, but not present in the
UCF.
```

- **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) should not be allocated to bank 6.
```

## Create Preset Configuration

This option outputs pre-verified configurations for the selected FPGA. The preset configurations meet the specified frequency with a reasonable margin. Banks, frequency, memory component, and all the other parameters are already selected. When the preset configuration option is selected and **Next** is clicked, the Preset Configurations page appears.

The flow for preset configuration is as follows:

1. [Create Preset Configuration](#)
2. [Memory License Agreement](#)
3. [Summary](#)
4. [PCB Information](#)
5. [Finish](#)

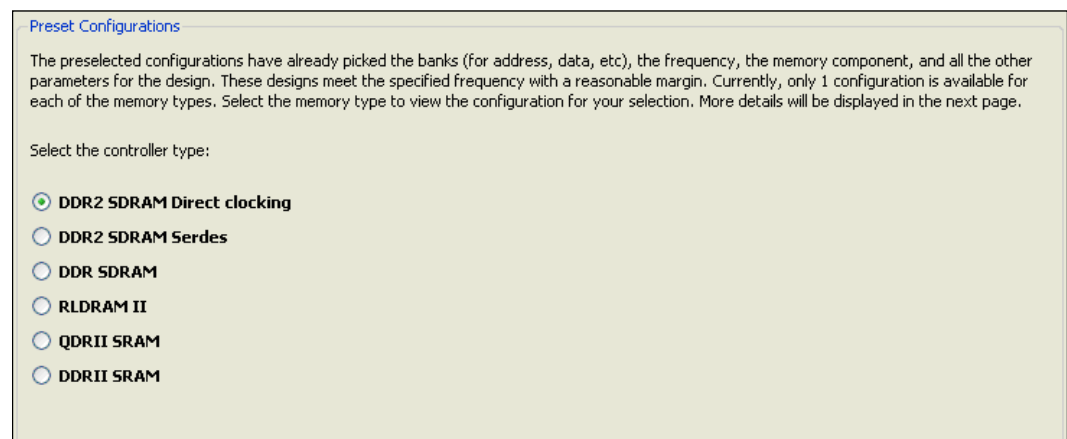


Figure 1-60: Preset Configurations

Select the controller type here to select the memory and click the **Next** button. In case of DDR2 SDRAM Direct clocking, DDR2 SDRAM SerDes, DDR SDRAM, and RLDRAM II memories, the Micron License page is displayed. For other memories, the Summary page is displayed.

### Memory License Agreement

Check or uncheck the check box to accept the License Agreement and then click **Next**.

### Summary

This page gives details about the options set for Preset Configuration. Make sure to check this page and ensure that the preset parameters are good for the design requirements. Refer to the [Summary](#) section for details.

Click the **Generate** button to generate the design files. This displays the PCB information page.

### PCB Information

Refer to the [PCB Information](#) section for details. Clicking **Next** displays the Finish page.

## Finish

Refer to the [Finish](#) section for more information.

## Spartan-3A FPGA DDR2 SDRAM 200 MHz Design

This page is displayed only for Spartan-3A FPGA designs. It provides links to XAPP458 [\[Ref 15\]](#) and the Spartan-3A DDR2 SDRAM 200 MHz reference design.

200 MHz Design support

**Implementing DDR2-400 Memory Interfaces in Spartan-3A FPGA's:**

- DDR2-400 has been validated in hardware
- Documented in application notes with reference design
- Additional bandwidth enables new applications and lower cost implementations of existing applications

**Interface Details:**

- 16-bit data width
- Spartan-3A FPGA - XC3S700A-5FG484C
- Memory Part - MT47H32M16BN-3(CL=3)
- Uses MIG Controller with minor modifications
- All five timing budgets documented and pass

Please refer to the links below for application notes and reference design

- [http://www.xilinx.com/prs\\_rls/2007/silicon\\_spart/0795\\_ddr2.htm](http://www.xilinx.com/prs_rls/2007/silicon_spart/0795_ddr2.htm)
- [http://www.xilinx.com/support/documentation/application\\_notes/xapp458.pdf](http://www.xilinx.com/support/documentation/application_notes/xapp458.pdf)
- [http://www.xilinx.com/support/documentation/application\\_notes/xapp458.zip](http://www.xilinx.com/support/documentation/application_notes/xapp458.zip)

UG086\_c1\_53\_010208

Figure 1-61: Spartan-3A FPGA 200 MHz Design Support

## Using MIG in Batch Mode

To run MIG in batch mode, the XCO and MIG . PRJ files must be created by running MIG in GUI mode through the CORE Generator system.

### XCO File

The XCO file contains the following information:

- Path of the MIG . prj file
- Synthesis tool to be used
- FPGA device information
- HDL to be used

To change these parameters they must be set in the XCO file.

### MIG.prj File

The user can change various parameter values in the PRJ file with valid input data and can regenerate the design. Parameters with a fixed value cannot be changed. [Table 1-1](#) describes the information contained in the PRJ file.

Table 1-1: PRJ File Parameters

Parameter	Description
Controller number	Indicates the most recent controller selected in the GUI before generating the design.
NoOfControllers	Indicates the number of controllers selected. Multicontrollers are supported only for Virtex-4 FPGA DDR2 SDRAM Direct clocking designs.
MemoryDevice	Contains the memory device configuration. For a multicontroller case, this parameter contains the most recent memory device selected.
SelectedPins	If the user reserves some pins, this parameter displays the remaining pins along with the bank number from the selected banks. If the user does not reserve any pins, no pins are displayed (the user can use all the pins).
ReservedPins	Displays the pins that are reserved by the user. If the user does not reserve any pins, no pins are displayed.
DCM	Indicates whether DCM is enabled [1] or disabled [0].
ModuleName	Displays the top-level design name assigned by the user.
dci_inouts_inputs	Indicates whether Digitally Controlled Impedance (DCI) for inputs and inouts is enabled [1] or disabled [0]. If DCI is enabled, input and inout pins have the DCI I/O standards.
dci_outputs	Indicates whether Digitally Controlled Impedance (DCI) for address and control signals is enabled [1] or disabled [0]. If DCI is enabled, address and control pins have the DCI I/O standards.
FPGADevice	Displays the compatible devices selected by the user for the selected target device. If the compatible devices are not selected, nothing is displayed.
Class	Indicates the I/O standard class. It can be either Class I or Class II. They determine the various drive strengths of the signal.
Debug_En	Indicates whether the debug signals are to be port-mapped to the ChipScope modules in design_top.

Table 1-1: PRJ File Parameters (Continued)

Parameter	Description
Controller number	Information related to each controller is between "<Controller number="X">" and "</Controller>". X holds the values from 0 to NoOfControllers – 1. It is different than ControllerNumber. The options selected by the user for each controller are listed below:
MemoryDevice	Gives the selected memory device and the memory type.
Clocking	Denotes the selected clocking type.
CCCheck	Indicates whether the Clock Capable (CC) option is enabled [1] or disabled [0]. If CC is enabled, strobe pins are allocated to the CC pins only.
Frequency	Indicates the frequency selected by the user for that controller.
DataWidth	Data width selected by the user.
Data Mask	Indicates whether Data Mask pin is to be allocated.
DeepMemory	Indicates the depth of the memory. This parameter is supported for Virtex-4 DDR2 SDRAM Direct clocking designs only. The depth of the memory for that controller is increased by multiples of DeepMemory value.
RowAddress	Indicates the row address width, this is the parameter of the Create New Memory Part.
MasterBanks	Indicates the Master Banks selected. (This appears only for the QDRII Virtex-5 FPGA design.)
ColAddress	Indicates the column address width, this is the parameter of the Create New Memory Part.
BankAddress	Indicates the bank address width, this is the parameter of the Create New Memory Part.
TimingParameters	Indicates various timing parameters of the selected Memory component.
ECC	Error Correction Code (ECC) is supported for Virtex-4 DDR2 SDRAM Direct clocking designs only.
WritePipeLine	Represents the pipeline stages. This parameter is supported for Spartan-3, Spartan-3E, Spartan-3A, and Spartan-3A DSP designs only.
BankSelection	Displays the banks selected by the user for that controller. Information about the particular bank is "<Bank Control="0" Address="0" SysClk="1" Dwrite="0" Data="0" name="3" wasso="16" />", where: <ul style="list-style-type: none"> <li>• "0" denotes signals that are not allocated in that bank.</li> <li>• "1" denotes signals that are allocated in that bank.</li> <li>• Control, Address, SysClk, Dwrite, and Data are the different signal groups.</li> <li>• "name" denotes the bank number.</li> <li>• "wasso" denotes the number of pins limited by the user in the particular bank.</li> </ul>

**Notes:**

1. All the above parameters might not be available for all the designs. They vary according to the design.

Load Mode and Extended Mode Register value parameters are listed in “[Mode Register Values](#).” These define specific modes of operation. These mode registers are not supported by all designs. They appear controller-wise.

Table 1-2: **Mode Register Values**

	Description
<mrBurstLength name="Burst Length" >8(011)</mrBurstLength>	Denotes the Burst length selected by the user. Valid values are 2 (001), 4 (010), or 8 (011), depending on the design.
<mrBurstType name="Burst Type" >sequential(0)</mrBurstType>	Gives information about the burst type. Not all designs support this parameter.
<mrCasLatency name="Cas Latency" >4(100)</mrCasLatency>	Supported CAS latencies are 3 (011), 4 (100) and 5 (101). Some designs do not have this concept.
<mrMode name="Mode" >normal(0)</mrMode>	MIG supports normal mode only. The test mode is used only by the manufacturer.
<mrDllReset name="DLL Reset" >no(0)</mrDllReset>	Self-clearing is supported when ‘1’. MIG does not support this option for all designs.
<mrPdMode name="PD Mode" >fast exit(0)</mrPdMode>	Power Down mode determines the performance versus power savings. MIG only supports fast exit mode.
<mrWriteRecovery name="Write Recovery" >5(100)</mrWriteRecovery>	During a WRITE with auto precharge operation, the DDR2 SDRAM delays the internal auto precharge operation by WR clocks. WR supports the following values: 2 (001), 3 (010), 4 (011), 5 (100) and 6 (101). This value varies depending on the user-selected frequency.
<emrDllEnable name="DLL Enable" >Enable-Normal(0)</emrDllEnable>	The DLL should be enabled for normal mode of operation.
<emrOutputDriveStrength name="Output Drive Strength" >Fullstrength(0)</emrOutputDriveStrength>	It selects full drive strength for all outputs. MIG supports full drive strength alone.
<emrRTT name="RTT (nominal) - ODT" >150ohms(10)</emrRTT>	On-Die Termination effective resistance (RTT) has the following values: Disabled (00), 75Ω(01), 150Ω (10), and 50Ω(11). MIG does not support 50Ω(11).
<emrPosted name="Additive Latency (AL)" >2(010)</emrPosted>	Additive Latency (AL) can have values of 0 (000), 1 (001), 2 (010), 3 (011), and 4 (100). MIG only supports AL values of 0, 1, and 2, depending on the design.



Table 1-2: Mode Register Values (Continued)

	Description
<emrOCD name="OCD Operation" >OCD Exit(000)</emrOCD>	Not supported by MIG.
<emrDQS name="DQS# Enable" >Enable(0)</emrDQS>	A 0 enables differential DQS. A 1 enables single DQS. This is applicable for designs supporting both differential and single-ended DQSs. For example, Virtex-4 DDR2 SDRAM designs supports both differential DQS and single-ended DQS.
<emrRDQS name="RDQS Enable" >Disable(0)</emrRDQS>	When enabled, RDQS is identical in function and timing to data strobe DQS during a READ operation. During a WRITE operation, RDQS is ignored by the DDR2 SDRAM. MIG does not support this option, which is disabled in the tool.
<emrOutputs name="Outputs" >Enable(0)</emrOutputs>	This value should always be 0 (enables the outputs). A value of 1 is not supported.

## Running in Batch Mode

The following GUI features are not supported in batch mode:

- Generate board files
- Verify UCF
- Read ucf file in the Reserve Pins option
- Save as option in the Reserve Pins option
- User guide
- Create New Memory Part
- Version info
- Real-time pin allocation

MIG designs can also be generated through the CORE Generator tool in batch mode as follows:

- First set the command prompt to the output path. To generate the MIG design, the following command is executed from the command prompt:

```
coregen -b <xcofilename>.xco -p <project path>
```

Where the <project path> indicates the path of the mig.prj file.

For example,

```
coregen -b test.xco -p D:\MIG_testing\coregen_test\v4_design
```

- After this command is executed, all the outputs are generated in the <Component Name> folder.





## *Section II: Virtex-4 FPGA to Memory Interfaces*

*Chapter 2, "Implementing DDR SDRAM Controllers"*

*Chapter 3, "Implementing DDR2 SDRAM Controllers"*

*Chapter 4, "Implementing QDR II SRAM Controllers"*

*Chapter 5, "Implementing DDR II SRAM Controllers"*

*Chapter 6, "Implementing RLDRAM II Controllers"*



## Implementing DDR SDRAM Controllers

This chapter describes how to implement DDR SDRAM interfaces for Virtex™-4 FPGAs generated by MIG. This design is based on XAPP709 [Ref 20].

### Feature Summary

#### Supported Features

The DDR SDRAM controller design supports the following:

- 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 precharge command given by the user
- Registered DIMMs, unbuffered DIMMs, and SODIMMs
- Different memories (density/speed)
- Auto refresh
- Linear addressing
- VHDL and Verilog
- With and without a testbench
- With and without a DCM

The supported features are described in more detail in “[Architecture](#).”

#### Design Frequency Ranges

Table 2-1: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-10		-11		-12	
	Min	Max	Min	Max	Min	Max
Component	100	165	100	170	100	175
DIMM	100	165	100	170	100	175

## Unsupported Features

- Dual Rank DIMMs
- Deep Memory
- Auto Precharge
- Bank Management
- Multi Controller

## 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.

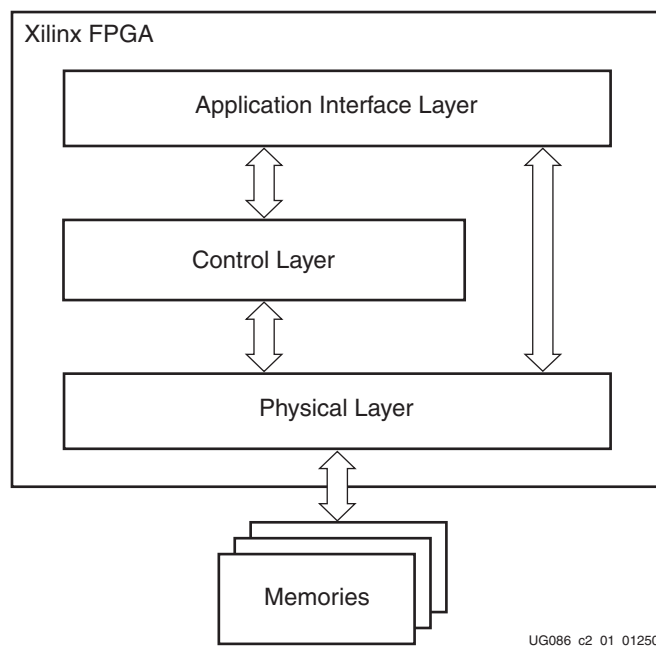


Figure 2-1: Modular Memory Interface Representation

### 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 set 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 from the GUI. The DDR SDRAM controller uses these parameters directly.

The user issues 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 2-10](#) or [Figure 2-12](#)) is decoded in a sequence. The total width of the Read/Write Address FIFO (rd\_wr\_addr\_fifo) is 36 bits. The user writes the column address (least-significant bits), row address, bank address, chip address [31:0], and the command to be issued [34:32]. The 36th bit (APP\_AF\_ADDR[35]) is reserved by the design to manipulate whether or not the row to be accessed is same as that of the previous row. The APP\_AF\_ADDR[35] input is a don't care for the design. The controller takes the row and column address bits based on the selected component. The ["Write Interface"](#) and ["Read Interface"](#) sections provide further details on how to issue the write and read commands, respectively.

[Table 2-2](#) lists the commands that the user can issue through the User interface. If the user issues an invalid command, the state of the controller is undefined. The functionality is not guaranteed when an invalid command is issued.

*Table 2-2: User Commands*

Command	APP_AF_ADDR[34:32]
READ	101
WRITE	100
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 from the GUI. It determines the maximum number of column locations accessed for a given READ or WRITE command.

The DDR SDRAM ddr\_controller module implements the user-selected burst length from MIG.

## 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 from the GUI.

The controller supports CAS latencies of 2, 2.5, and 3.

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 and address 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.

## Unbuffered DIMMs and SODIMMs

DDR SDRAM design supports unbuffered DIMMs and SODIMMs. Unbuffered DIMMs are normal DIMMs where a set of components are used to get a particular configuration. SODIMMs vary from the unbuffered DIMMs only by package type. They are functionally the same.

## Precharge

The PRECHARGE command is issued before the next read or write is issued for a different row, but not if the read or write is in the same row. The PRECHARGE command checks the row address, bank address, and chip selects. The DDR Virtex-4 FPGA controller issues a PRECHARGE command if there is a change in any address where a read or write command is to be issued. The AUTO PRECHARGE command via the A10 column bit is not supported.

## 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 there is a refresh request while during an ongoing read or write burst, the controller issues a REFRESH command after completing the current read or write burst command.

## Linear Addressing

The DDR SDRAM controller supports linear addressing. Linear addressing refers to the way the user provides the address of the memory to be accessed. For Virtex-4 DDR SDRAM controllers, the user provides the address information through the `app_af_addr` signal. As the densities of the memory devices vary, the number of column address bits and row address bits also changes. In any case, the row address bits in the `app_af_addr` signal always start from the next-higher bit, where the column address ends. This feature increases the number of devices that can be supported with the design.

## 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. Higher densities can be created using the "Create new memory part" feature of MIG. The maximum supported column address is 13 bits, the maximum row address is 15 bits, and the maximum bank address is 2 bits. 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 2-3](#) lists the timing parameters for components, and [Table 2-4](#) lists the timing parameters for DIMMs.



Table 2-3: 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 <sub>CK</sub>	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 <sub>MRD</sub>	LOAD MODE Command Cycle Time		10 ns	15 ns	10 ns	15 ns	10 ns	15 ns	10 ns	15 ns
T <sub>RP</sub>	PRECHARGE Command Period		15 ns	20 ns	15 ns	20 ns	15 ns	20 ns	15 ns	20 ns
T <sub>RFC</sub>	REFRESH Time		70 ns	75 ns	70 ns	75 ns	70 ns	75 ns	120 ns	120 ns
T <sub>RCD</sub>	ACTIVE to READ or WRITE Delay		15 ns	20 ns	15 ns	20 ns	15 ns	20 ns	15 ns	20 ns
T <sub>RAS</sub>	ACTIVE to PRECHARGE Command		40 ns	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns
T <sub>RC</sub>	ACTIVE to ACTIVE (Same Bank) Command		55 ns	65 ns	55 ns	65 ns	55 ns	65 ns	55 ns	65 ns
T <sub>WTR</sub>	WRITE to READ Command Delay		2 * T <sub>CK</sub>	1 * T <sub>CK</sub>	2 * T <sub>CK</sub>	1 * T <sub>CK</sub>	2 * T <sub>CK</sub>	1 * T <sub>CK</sub>	2 * T <sub>CK</sub>	1 * T <sub>CK</sub>
T <sub>WR</sub>	WRITE Recovery Time		15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns

Table 2-4: 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 <sub>CK</sub>	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 <sub>MRD</sub>	LOAD MODE Command Cycle Time		10 ns	10 ns	10 ns	10 ns
T <sub>RP</sub>	PRECHARGE Command Period		15 ns	15 ns	15 ns	15 ns
T <sub>RFC</sub>	REFRESH Time		70 ns	70 ns	70 ns	70 ns
T <sub>RCD</sub>	ACTIVE to READ or WRITE Delay		15 ns	15 ns	15 ns	15 ns
T <sub>RAS</sub>	ACTIVE to PRECHARGE Command		40 ns	40 ns	40 ns	40 ns
T <sub>RC</sub>	ACTIVE to ACTIVE (Same Bank) Command		55 ns	55 ns	55 ns	55 ns

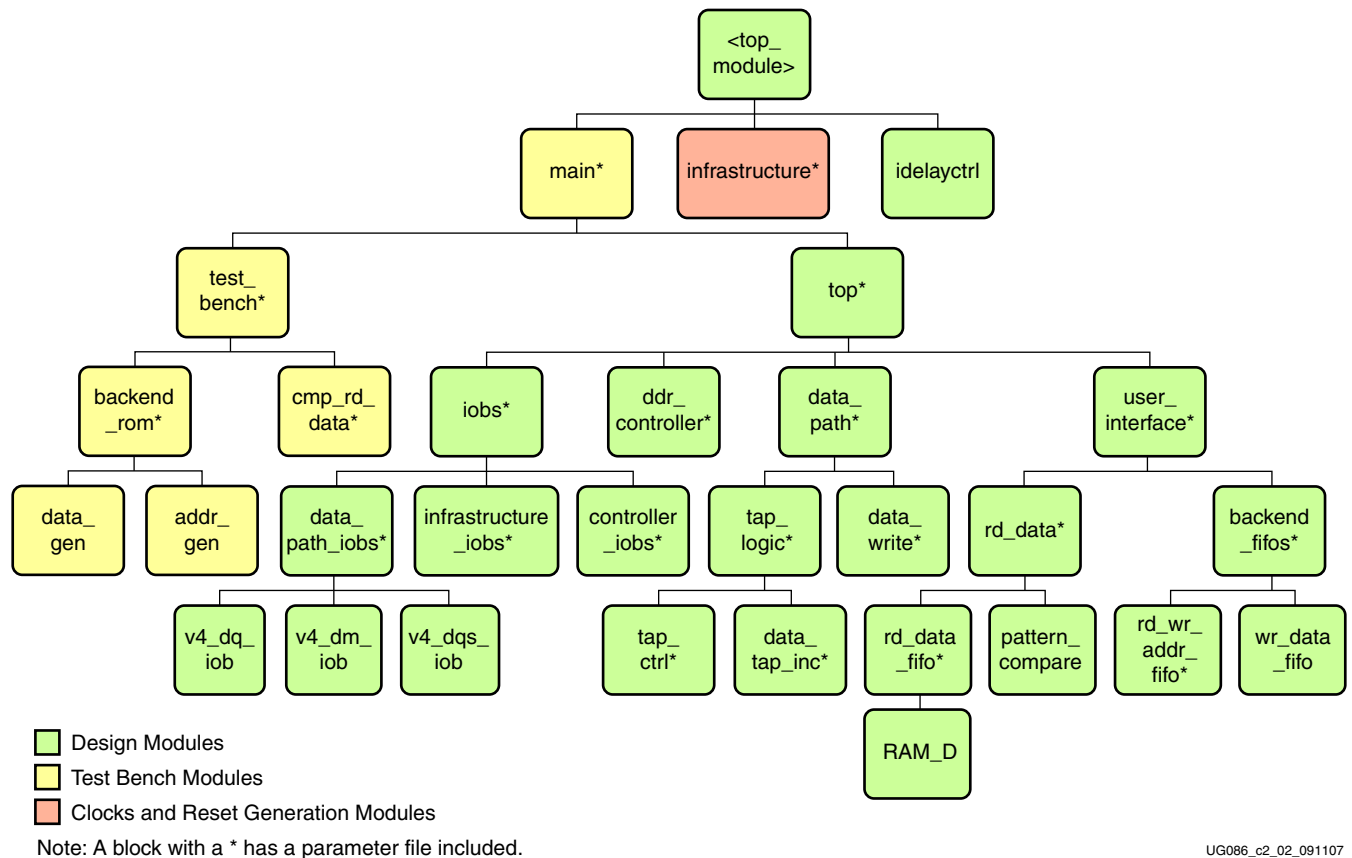
Table 2-4: Timing Parameters for DIMMs (Unbuffered and Registered) (Continued)

Parameter	Description	Micron 128 MB	Micron 256 MB	Micron 512 MB	Micron 1 GB
		-40	-40	-40	-40
T <sub>WTR</sub>	WRITE to READ Command Delay	2 * T <sub>CK</sub>	2 * T <sub>CK</sub>	2 * T <sub>CK</sub>	2 * T <sub>CK</sub>
T <sub>WR</sub>	WRITE Recovery Time	15 ns	15 ns	15 ns	15 ns

**Note:** For the latest timing information, refer to the vendor memory data sheets.

## Hierarchy

Figure 2-2 shows the hierarchical structure of the DDR SDRAM design generated by MIG with a testbench and a DCM. The physical and control layers are clearly separated in this figure. MIG generates the entire DDR SDRAM controller as shown in this hierarchy, including the testbench. MIG also generates a parameter file where all user input parameters or some parameters used internally by the design are defined.



UG086\_c2\_02\_091107

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

The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks and reset generation modules

There is a parameter file generated with the design that has all the user input and design parameters selected from MIG.

MIG can generate four different DDR SDRAM designs:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

When the testbench is not generated by MIG, the top-level module has the user interface signals. The list of user interface signals is provided in [Table 2-7](#).

Design clocks and resets are generated in the infrastructure module. The DCM clock is instantiated in the infrastructure module for designs with a DCM. The inputs to this module are the differential design clock and a 200 MHz differential clock for the IDELAYCTRL module. A user reset is also input to this module. Using the input clocks and reset signals, the system clocks and the system reset are generated in this module, which is used in the design.

The DCM primitive is not instantiated in this module if the **Use DCM** option is unchecked. So, the system operates on the user-provided clocks. The system reset is generated in the infrastructure module using the DCM\_LOCK input signal.

[Figure 2-3](#) shows a DDR SDRAM controller block diagram representation of the top-level module for a design with a DCM and a testbench. SYS\_CLK\_P and SYS\_CLK\_N are differential input system clocks. The DCM clock is instantiated in the infrastructure module that generates the required design clocks. CLK200\_P and CLK200\_N are used for the idelay\_ctrl element. SYS\_RESET\_IN\_N is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal. Memory device signals are prepended with the controller number. For example, DDR\_RAS\_N appears as *cntrl0\_DDR\_RAS\_N*.

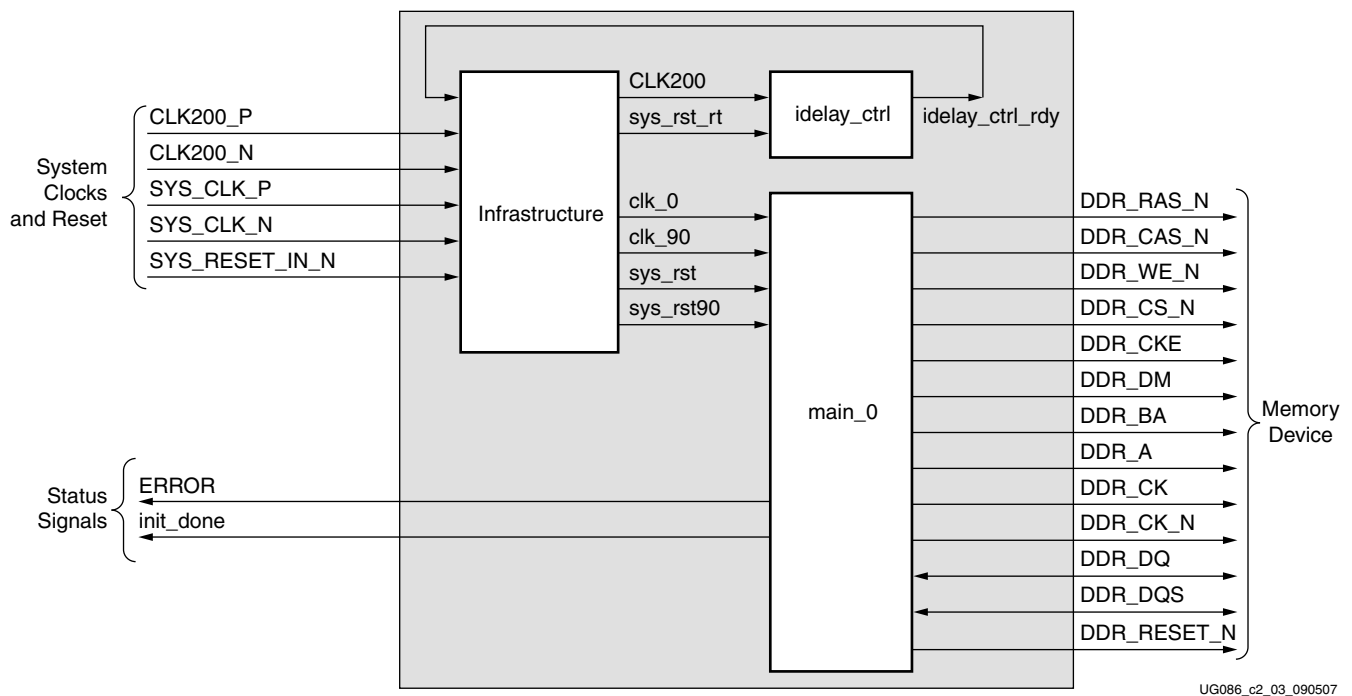


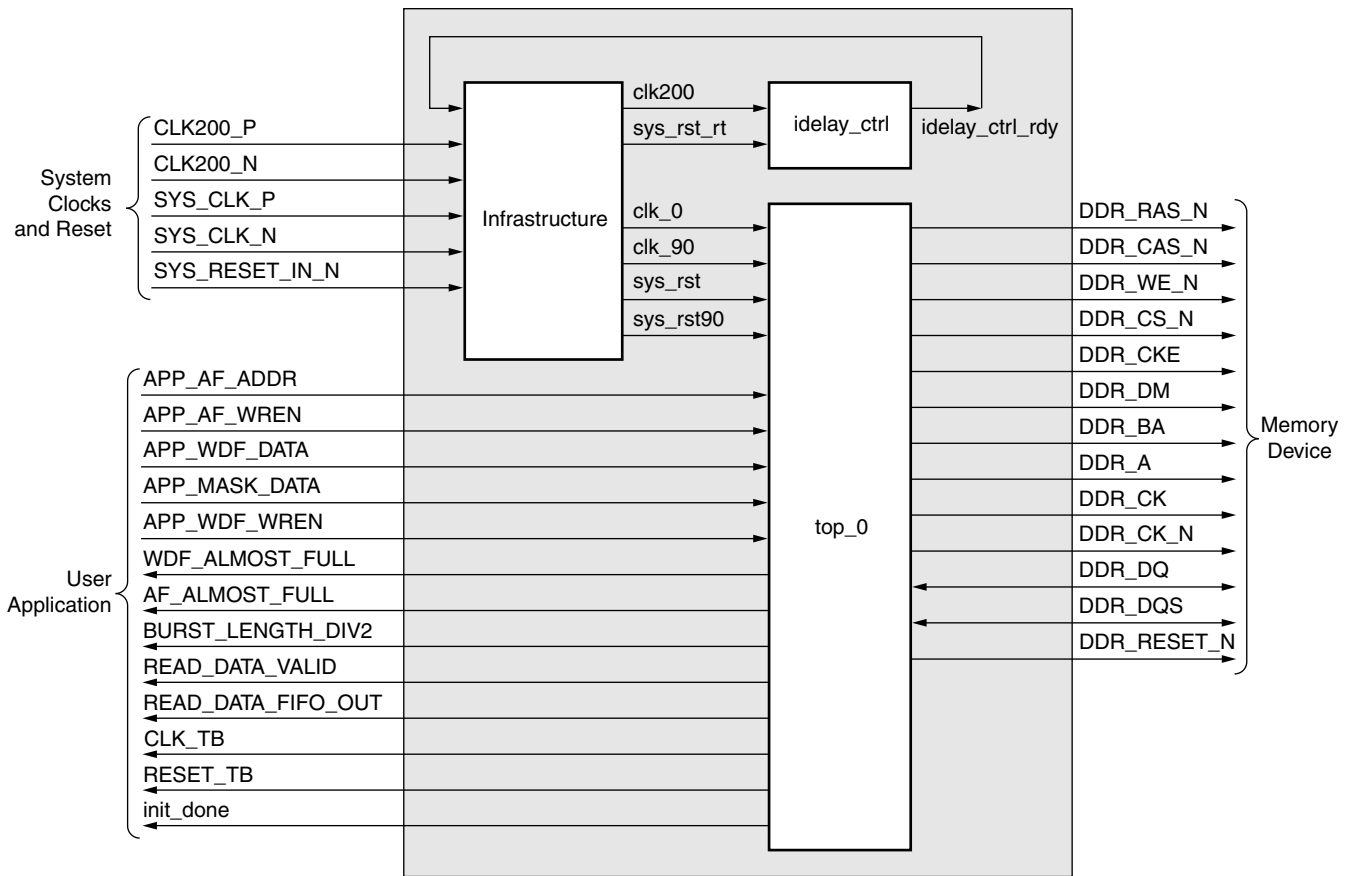
Figure 2-3: Top-Level Block Diagram of the DDR SDRAM Design with a DCM and a Testbench

The error output signal indicates whether the case passes or fails. The testbench module does writes and reads, and also compares the read data with the written data. The error signal is driven High on data mismatches.

The `init_done` signal indicates the completion of initialization and calibration of the design.

All the signals listed under the Memory Device category do not necessarily appear in the top level port list. The port list varies according to the memory type selected, such as a component or a registered DIMM. For example, a component does not have the `ddr_reset_n` signal.

Figure 2-4 shows a block diagram representation of the top-level module for a design with a DCM but without a testbench.



UG086\_c2\_04\_090507

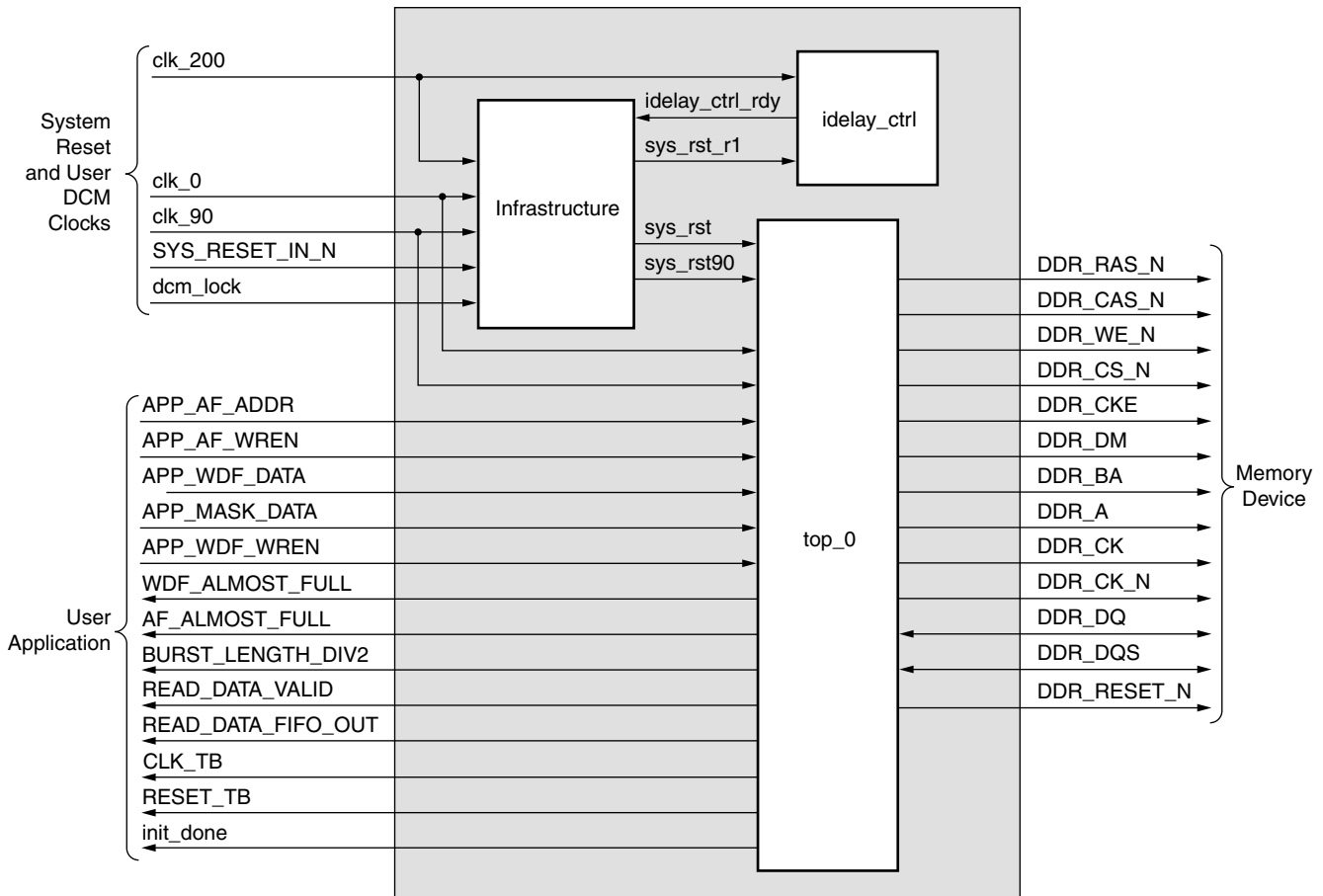
Figure 2-4: Top-Level Block Diagram of the DDR SDRAM Design with a DCM but without a Testbench

The DCM clock module is instantiated in the infrastructure module. Using the differential `SYS_CLK_P` and `SYS_CLK_N` signals, the internal DCM generates all the required clocks for the design. `CLK200_P` and `CLK200_N` are used by the `idelay_ctrl` element. `SYS_RESET_IN_N` is the active-Low system reset signal. All design resets are generated using the input reset signal gated by the `dcm_lock` signal.

The `init_done` signal indicates the completion of initialization and calibration of the design.

The application’s user interface signals are listed in Figure 2-4. The design provides the `clk_tb` and `reset_tb` signals to the user to synchronize with the design.

Figure 2-5 shows a block diagram representation of the top-level module for a design without a DCM or a testbench. There is no DCM instantiated in the infrastructure module. All the clocks and dcm\_lock should be given as inputs from the user interface. Resets are generated using the SYS\_RESET\_IN\_N signal gated by the dcm\_lock signal in the infrastructure module. Clk200 is used by the idelay\_ctrl element. All the clocks should be single-ended. The user application must have a DCM primitive instantiated in the design. The init\_done signal indicates the completion of initialization and calibration of the design. The user interface signals are also listed in the <top\_module> module. The design provides the clk\_tb and reset\_tb signals to the user to synchronize with the design.



UG086\_c2\_05\_090507

Figure 2-5: Top-Level Block Diagram of the DDR SDRAM Design without a DCM or a Testbench

Figure 2-6 shows a block diagram representation of the top-level module for a design with a testbench but without a DCM. The user should provide all the clocks and the dcm\_lock signal. These clocks should be single-ended. SYS\_RESET\_IN\_N is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

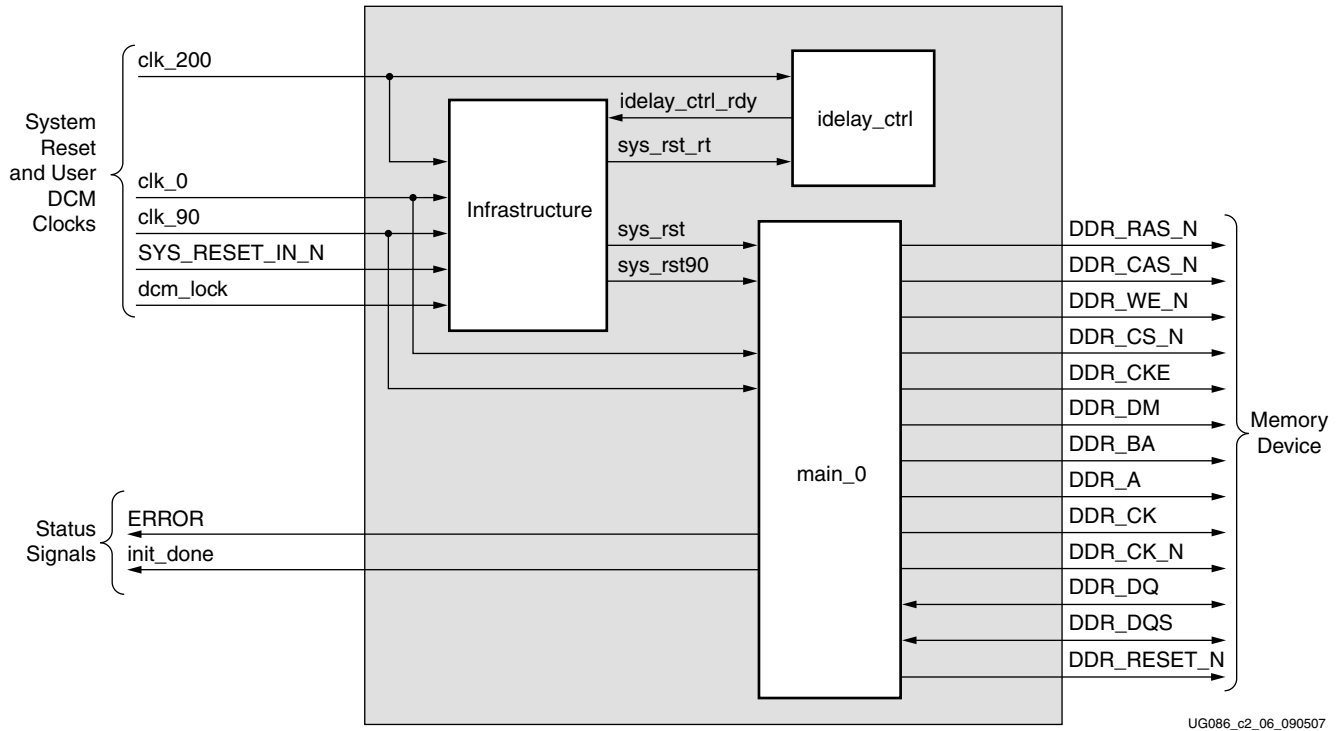
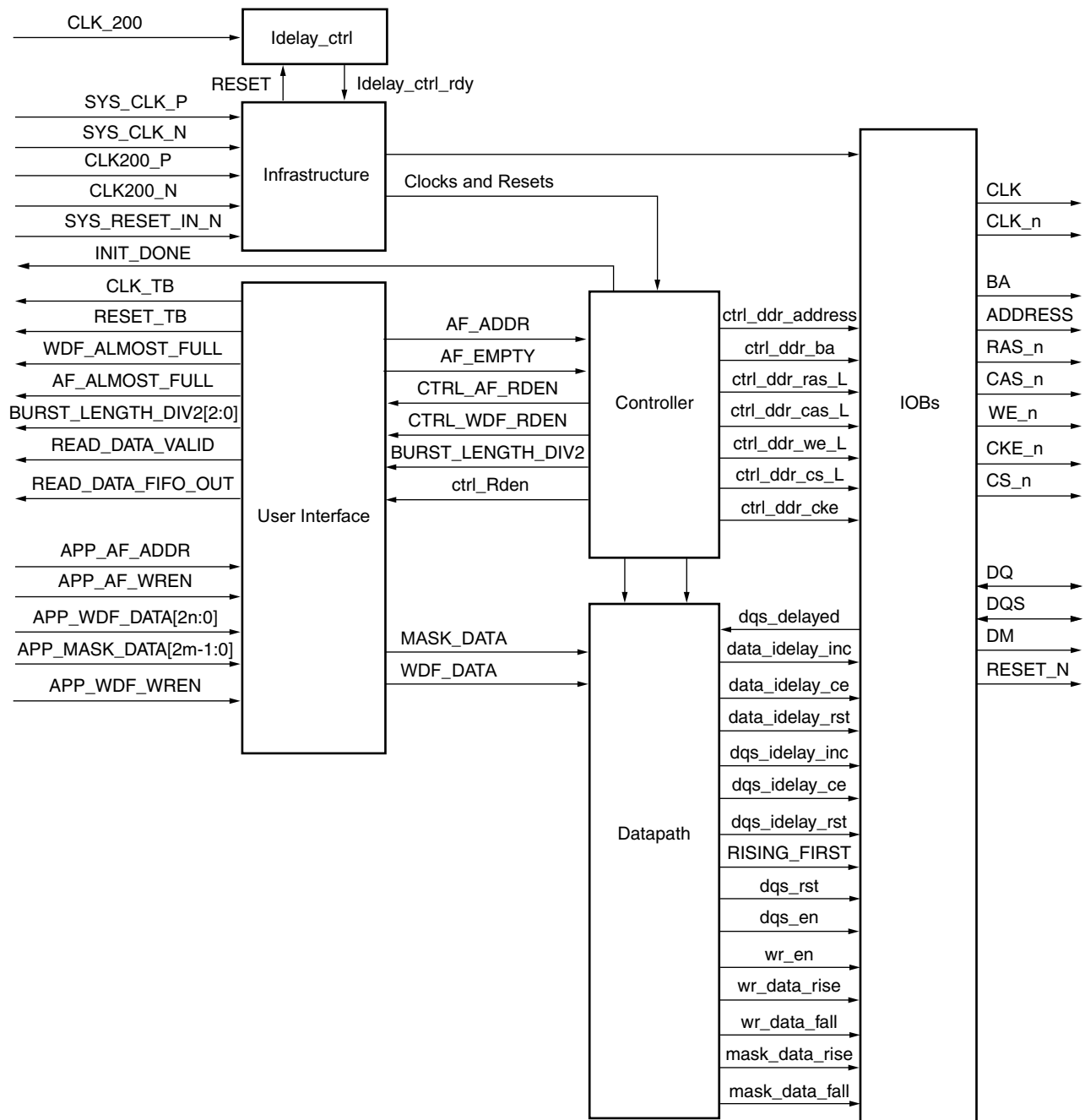


Figure 2-6: Top-Level Block Diagram of the DDR SDRAM Design with a Testbench but without a DCM

The error output signal indicates whether the case passes or fails. The testbench module does writes and reads, and also compares the read data with the written data. The ERROR signal is driven High on data mismatches. The init\_done signal indicates the completion of initialization and calibration of the design.

Figure 2-7 shows the expanded block diagram of the design. The top module is expanded to show various internal blocks. The functions of these blocks are explained in the subsections following the figure.



UG086\_c2\_07\_090507

Figure 2-7: Expanded 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 is suspended until the `ref_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.

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 Initialization and Calibration

DDR memory is initialized through a specified sequence as shown in [Figure 2-8](#). The controller starts the memory initialization at power up itself. 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 `sel_done` port in `tap_logic` module indicates the completion of DQS to FPGA calibration. The number of



taps is then used to delay the data during normal reads to register the valid data in the FPGA. XAPP701 [Ref 17] provides more information about the calibration architecture.

Following the strobe detection, the controller does a read enable calibration. This calibration is used to determine the delay from read command to read data at rd\_data\_fifo. The delay between read command and read data is affected by the CAS latency parameters, the PCB traces, and the I/O buffer delays. Read enable calibration is used to generate a write enable to rd\_data\_fifo so that valid data is registered. Controller writes a known fixed pattern and reads back the data from memory. The read data is compared against the known fixed pattern. The comp\_done port in rd\_data module indicates the completion of the read enable calibration.

The init\_done port indicates the completion of both DQS to FPGA calibration and read enable calibration. After initialization and calibration is done, the controller can start issuing user commands to the memory.

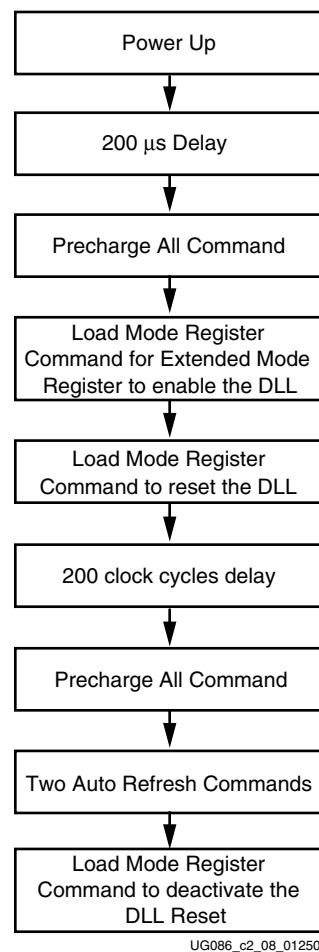


Figure 2-8: DDR Memory Initialization Sequence

## DDR SDRAM System and User Interface Signals

Table 2-5 describes the DDR SDRAM system interface signals for designs with the DCM. 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\_RESET\_IN\_N resets all the logic.

Table 2-5: DDR SDRAM System Interface Signals for Designs with DCM

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_RESET_IN_N	Input	Active-Low reset to the DDR SDRAM controller.
CLK200_P, CLK200_N	Input	Differential clock used in the idelay_ctrl logic.

Table 2-6 shows the system interface signals for designs without the DCM. The clk\_0, clk\_90, and clk\_200 signals are the single-ended input clocks. The clk\_90 signal must have a phase difference of 90° with respect to clk\_0. The clk\_200 signal is the clock used for the IDELAYCTRL primitives in Virtex-4 FPGAs.

Table 2-6: System Interface Signals for Designs without the DCM

Signal	Direction	Description
clk_0	Input	The DDR SDRAM controller and memory operate on this clock.
SYS_RESET_IN_N	Input	Active-Low reset to the DDR SDRAM controller. This signal is used to generate a synchronous system reset.
clk_90	Input	90° phase-shifted clock with the same frequency as clk0.
clk_200	Input	200 MHz input differential clock for the IDELAYCTRL primitive of the Virtex-4 FPGA.
dcm_lock	Input	The status signal indicating whether the DCM is locked or not. It is used to generate the synchronous system reset.

Table 2-7 describes the DDR SDRAM user interface signals for designs without the testbench.

Table 2-7: DDR SDRAM User Interface Signals for Designs without the Testbench Case

Signal Name	Direction	Description
CLK_TB	Output	All user interface signals must be synchronized with respect to CLK_TB.
RESET_TB	Output	Active-High system reset for the user interface.
BURST_LENGTH_DIV2[2:0]	Output	Indicates the number of bursts that can be written to or read from the memory. 001: burst length = 2 010: burst length = 4 100: burst length = 8

Table 2-7: DDR SDRAM User Interface Signals for Designs without the Testbench Case (Continued)

Signal Name	Direction	Description
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 [2n-1:0]	Output	Read data from memory, where $n$ is the data width of the interface. 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 5 more locations into the FIFO in designs generated with a testbench and 14 more locations in designs without a testbench.
AF_ALMOST_FULL	Output	ALMOST FULL status of the Read Address FIFO. The user can issue eight more locations into the FIFO after AF_ALMOST_FULL is asserted.
APP_AF_ADDR[35:0]	Input	Memory address and command. Bit 35 is used internally by the controller. The controller ignores this bit from the user interface. Bits [34:32] are used for dynamic commands as follows: 001: Auto Refresh 010: Precharge 100: Write 101: Read  Bits [31:0] form the memory chip select, bank address, row address, and column address. The positioning of the chip, bank, row, and column addresses changes based on the memory configuration.
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.
APP_MASK_DATA[2m-1:0]	Input	User mask data, where $m$ indicates the data mask width of the interface. The mask data is twice the mask width of the interface. The mask data is written into the Write Data FIFO along with the write data.
APP_WDF_DATA[2n-1:0]	Input	User write data to the memory, where $n$ indicates the data width of the interface. The user write data is twice the data width of the interface. 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_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.

**Notes:**

1. All user interface signal names are prepended with a controller number, for example, cntrl0\_APP\_WDF\_DATA. DDR SDRAM devices currently support only one controller.

Table 2-8 describes the status signals that are available to the user.

Table 2-8: DDR SDRAM Design Status Signals

Signal Name	Direction	Description
init_done	Output	This signal indicates the completion of initialization and calibration of the design.

## User Interface Accesses

The user backend logic communicates with the memory controller through a FIFO-based user interface. This interface consists of three related buses:

- A Command/ Address FIFO bus, which accepts write/read commands as well as the corresponding memory address from the user
- A Write Data FIFO bus, which accepts the corresponding write data when the user issues a write command on the Command/Address bus
- A Read bus on which the corresponding read data for an issued read command is returned

The user interface has the following timing and signaling restriction:

- When issuing a write command, the first write data word must be written to the Write Data FIFO no more than one clock cycle after the write command is issued. This restriction arises from the fact that the controller assumes write data is available when it receives the write command from the user.

## Write Interface

Figure 2-9 shows the user interface block diagram for write operations.

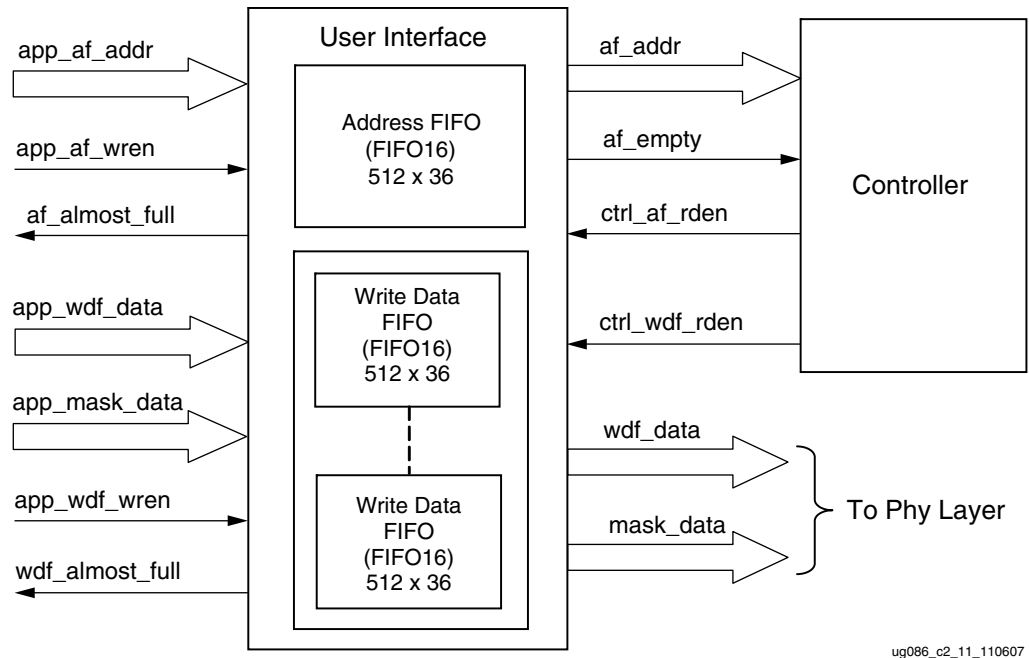
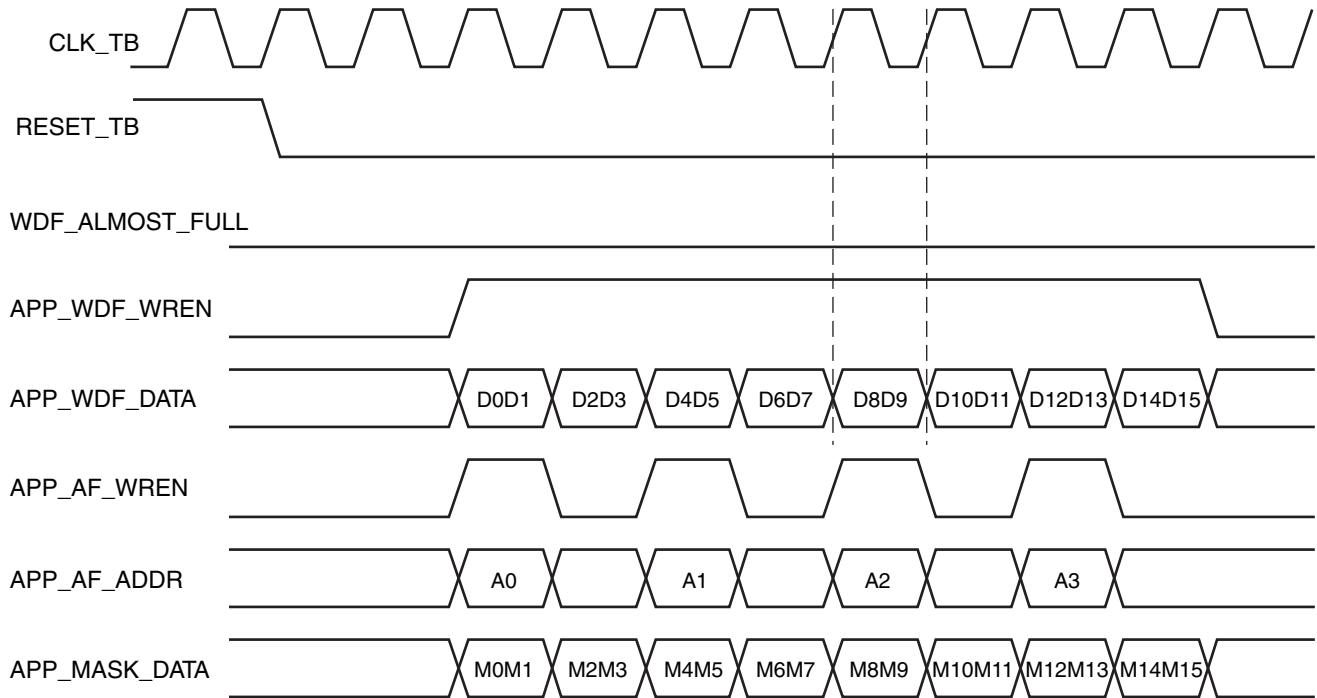


Figure 2-9: User Interface Block Diagram for Write Operations

The following steps describe the architecture of the Address and Write Data FIFOs and show how to perform a write burst operation to DDR SDRAM from the user interface.

1. The user interface consists of an Address FIFO and a Write Data FIFO. These FIFOs are constructed using Virtex-4 FIFO16 primitives with a 512 x 36 configuration. The 36-bit architecture comprises one 32-bit port and one 4-bit port. For Write Data FIFOs, the 32-bit port is used for data bits and the 4-bit port is used for mask bits.
2. The Common Address FIFO is used for both write and read commands, and comprises a command part and an address part. Command bits discriminate between write and read commands.
3. User interface data width `app_wdf_data` is twice that of the memory data width. For an 8-bit memory width, the user interface is 16 bits consisting of rise data and fall data. For every 8 bits of data, there is a mask bit. For 72-bit memory data, the user interface data width `app_wdf_data` is 144 bits, and the mask data `app_mask_data` is 18 bits.
4. The minimum configuration of the Write Data FIFO is 512 x 36 for a memory data width of 8 bits. For an 8-bit memory data width, the least significant 16 bits of the data port is used for write data. The controller internally pads all zeros for the most-significant 16 bits.
5. Depending on the memory data width, MIG instantiates multiple FIFO16s to gain the required width. For designs using 8-bit data width, one FIFO16 is instantiated; for 72-bit data width, a total of five FIFO16s are instantiated. The bit architecture comprises 16 bits of rising-edge data, 2 bits of rising-edge mask, 16 bits of falling-edge data, and 2 bits of falling-edge mask, which are all stored in a FIFO16. MIG routes the `app_wdf_data` and `app_mask_data` to FIFO16s accordingly.
6. The user can initiate a write to memory by writing to the Address FIFO and the Write Data FIFO when the FIFO Full flags are deasserted. Status signal `af_almost_full` is asserted when Address FIFO is full, and similarly `wdf_almost_full` is asserted when Write Data FIFO is full.
7. Both the Address FIFO and Write Data FIFO Full flags are deasserted with power-on.
8. The user should assert the Address FIFO write-enable signal `app_af_wren` along with address `app_af_addr` to store the write address and write command into the Address FIFO.
9. The user should assert the Data FIFO write-enable signal `app_wdf_wren` along with write data `app_wdf_data` and mask data `app_mask_data` to store the write data and mask data into the Write Data FIFO. The user should provide both rise and fall data together for each write to the Data FIFO.
10. The controller reads the Address FIFO by issuing the `ctrl_af_rden` signal. The controller reads the Write Data FIFO by issuing the `ctrl_wdf_rden` signal after the Address FIFO is read. It decodes the command part after the Address FIFO is read.
11. The write command timing diagram in [Figure 2-10](#) is derived from the MIG-generated test bench. As shown (burst length of 4), each write to the Address FIFO must be coupled with *two* writes to the Data FIFO. Similarly, for a burst length of 8, every write to the Address FIFO must be coupled with *four* writes to the Data FIFO. Failure to follow this rule can cause unpredictable behavior.



ug086\_c2\_09\_072006

Figure 2-10: DDR SDRAM Write Burst for Four Bursts (BL = 4)

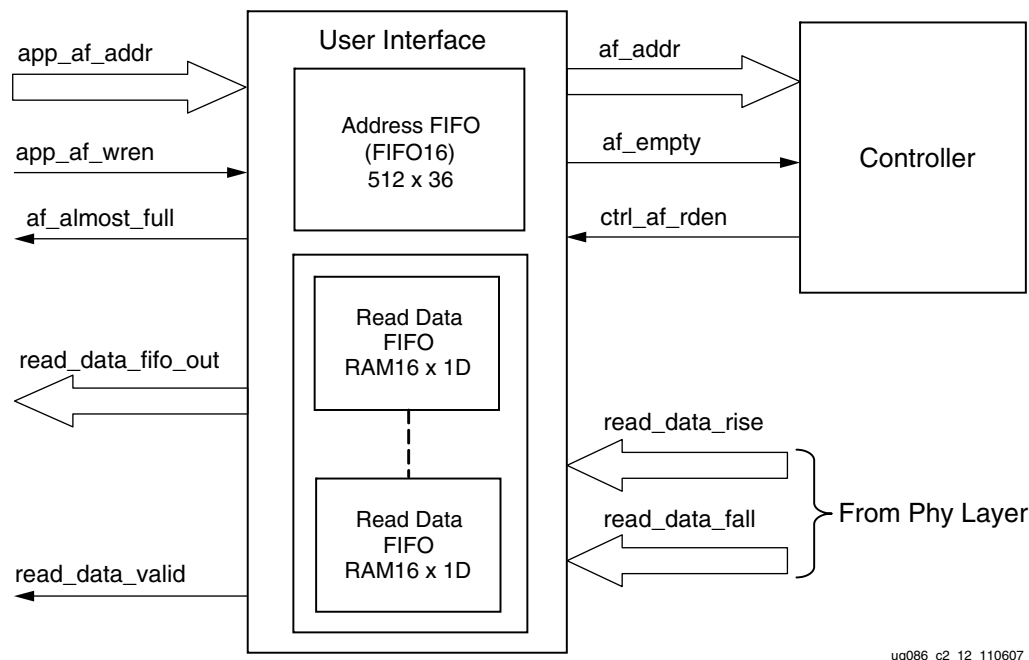
### Correlation between the Address and Data FIFOs

There is a worst case two-cycle latency from the time the address is loaded into the address FIFO on APP\_AF\_ADDR[35:0] to the time the controller decodes the address. Because of this latency, it is not necessary to provide the address on the last clock where data is entered into the data FIFO. If the address is written before the last data phase, the overall efficiency and performance increases because it eliminates or reduces the two-cycle latency. However, if the address is written before data is input into the data FIFO, a FIFO empty condition might result because the Data FIFO does not contain valid data.

Based on these considerations, Xilinx recommends entering the address into the address FIFO between the first data phase and the next-to-last data phase. For a burst of four or eight, this means the address can be asserted one clock before the first data phase. This implementation increases efficiency by reducing the one clock latency and guarantees that valid data is available in the Data FIFO.

## Read Interface

Figure 2-11 shows a block diagram of the read interface.



ug086\_c2\_12\_110607

Figure 2-11: User Interface Block Diagram for Read Operation

The following steps describe the architecture of the Read Data FIFOs and show how to perform a burst read operation from DDR SDRAM from the user interface.

1. The read user interface consists of an Address FIFO and a Read Data FIFO. The Address FIFO is common to both read and write operations. These FIFOs are constructed using Virtex-4 Distributed RAMs with a 16 x 1 configuration. MIG instantiates a number of RAM16Ds depending on the data width. For example, for 8-bit data width, MIG instantiates a total of 16 RAM16Ds, 8 for rising-edge data and 8 for falling-edge data. Similarly, for 72-bit data width, MIG instantiates a total of 144 RAM16Ds, 72 for rising-edge data and 72 for falling-edge data.
2. The user can initiate a read to memory by writing to the Address FIFO when the FIFO Full flag `af_almost_full` is deasserted.
3. To write the read address and read command into the Address FIFO, the user should issue the Address FIFO write-enable signal `app_af_wren` along with read address `app_af_addr`.
4. The controller reads the Address FIFO containing the address and command. After decoding the command, the controller generates the appropriate control signals to memory.
5. Prior to the actual read and write commands, the design calibrates the latency (number of clock cycles) from the time the read command is issued to the time data is received. Using this pre-calibrated delay information, the controller generates the write-enable signals to the Read Data FIFOs.
6. The `read_data_valid` signal is asserted when data is available in the Read Data FIFOs.
7. Figure 2-12 shows a user interface timing diagram for a burst length of 4, CAS latency of 3 at 175 MHz, and a `Trcd` value of the memory part at 20 ns. The read latency is calculated from the point when the Read command is given by the user to the point

when the data is available with the read\_data\_valid signal. The minimum latency in this case is 26 clocks, where no precharge is required, no auto-refresh request is pending, the user commands are issued after initialization is completed, and the first command issued is a Read command. The controller executes the commands only after initialization is done as indicated by the init\_done signal.

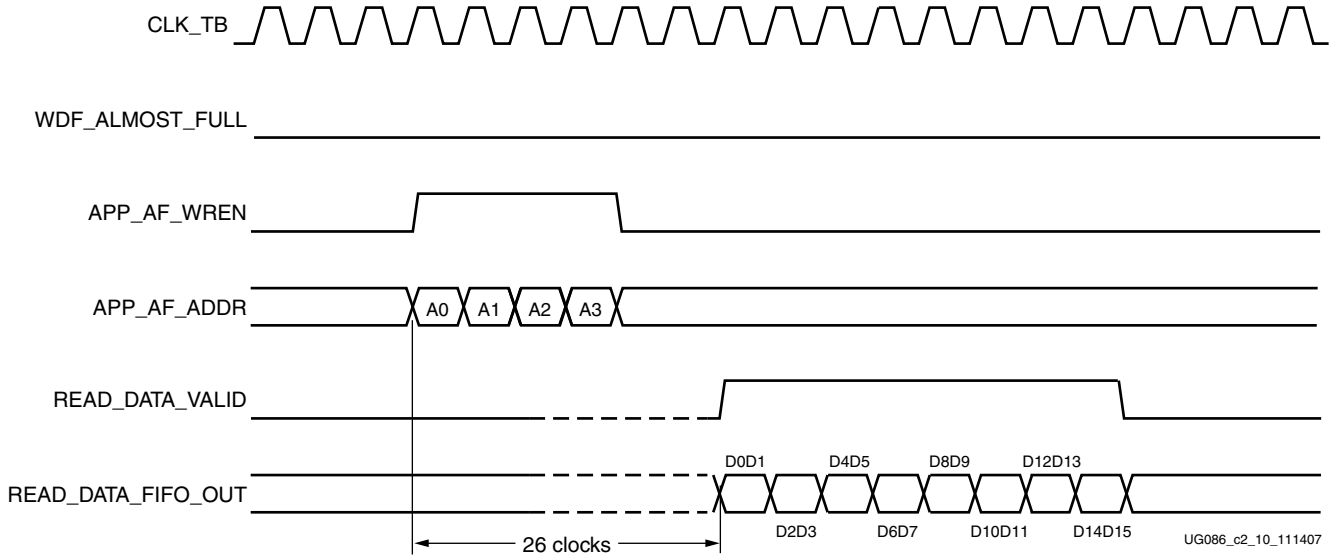


Figure 2-12: DDR SDRAM Read Burst for Four Bursts (BL = 4)

8. After the address and command are loaded into the Address FIFO, it takes 26 clock cycles minimum for CL = 3 at a frequency of 175 MHz for the controller to assert the read\_data\_valid signal.
9. Read data is available only when the read\_data\_valid signal is asserted. The user should access the read data on every positive edge of the read\_data\_valid signal.

The read latency for the case where (1) CL = 3, (2) the read is written to an empty address/command FIFO, (3) the read targets an unopened bank/row, and (4) the frequency is 175 MHz, is broken down as indicated in Table 2-9.

Table 2-9: Read Command to Read Data Clock Cycles

Parameter	Number of Clock Cycles
Read Command to Empty Signal Deassertion	7 clocks
Empty to Active Command	5.5 clocks
Active to Read Command	4 clocks
Memory Read to Valid	9.5 clocks
<b>Total:</b>	<b>26 clocks</b>

In general, read latency varies based on the following parameters:

- Number of commands already in the FIFO pipeline before the read command is issued
- Whether an ACTIVATE command needs to be issued to open the new bank/row
- Whether a PRECHARGE command needs to be issued to close a previously opened bank



- Specific timing parameters for the memory, such as  $T_{RAS}$  and  $T_{RCD}$  in conjunction with the bus clock frequency
- Commands can be interrupted, and banks/rows can forcibly be closed when the periodic AUTO REFRESH command is issued
- CAS latency
- If the user issues the commands before initialization is complete, the latency cannot be determined.
- Board-level and chip-level (for both memory and FPGA) propagation delays

Table 2-10 shows the list of signals allocated in a group from bank selection check boxes. See Chapter 11, “Implementing DDR SDRAM Controllers,” for more factors.

Table 2-10: DDR SDRAM Signal Allocations

Bank Selected	Signals Allocated in the Group
Address	Memory address, memory control, and memory clock signals
Data	Data, data mask, and data strobes
System Control	System reset from the user interface and status signals
System_Clock	System clocks from the user interface

MIG allows selection of banks for different classes of memory signals. When a particular bank is checked for an address, MIG allocates the memory address, the memory control, and the memory clocks in that bank. When a bank is checked for data, MIG allocates the data, the data mask, and the data strobes in that bank. When a bank is checked for system control, MIG allocates the system reset and status signals in that bank. When a bank is checked for system clocks, MIG allocates the system clock signals in that bank.

## Simulating the DDR SDRAM Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains the external testbench, memory model, do file and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more about the files in the `sim` folder and to simulate the design, refer to the `simulation_help.chm` file in `sim` folder.

### Changing the Refresh Rate

Change the global ``define` (for Verilog) or constant (for VHDL) variable `MAX_REF_CNT` in `mymodule_parameters_0.v` (or `.vhd`) so that  $MAX\_REF\_CNT = (\text{refresh interval in clock periods}) = (\text{refresh interval}) / (\text{clock period})$ . For example, for a refresh rate of  $3.9 \mu\text{s}$  with a memory bus running at 200 MHz:

$$MAX\_REF\_CNT = 3.9 \mu\text{s} / (\text{clock period}) = 3.9 \mu\text{s} / 5 \text{ ns} = 780 \text{ (decimal)} = 0x30C$$

If the above value exceeds  $2^{MAX\_REF\_WIDTH} - 1$ , the value of `MAX_REF_WIDTH` must be increased accordingly in `parameters_0.v` (or `.vhd`) to increase the width of the counter used to track the refresh interval.

## Supported Devices

The design generated out of MIG is independent of the memory package, hence the package part of the memory component is replaced with XX or XXX, where XX or XXX to indicate a don't care condition. The tables below list the components (Table 2-11) and DIMMs (Table 2-12 through Table 2-14) supported by MIG for DDR SDRAM. In supported devices, XX in the memory component column denotes either single or two alphanumeric characters. For example, MT46V32M4XX-75 can be either MT46V32M4P-75 or MT46V32M4BN-75. An X in the DIMM columns (for Unbuffered, Registered, and SO DIMMs) denotes a single alphanumeric character. For example, MT9VDDF3272X-40B can be either MT9VDDF3272G-40B or MT9VDDF3272Y-40B. Similarly MT4VDDT1664AX-40B can be either MT4VDDT1664AG-40B or MT4VDDT1664AY-40B.

Table 2-11: Supported Components for DDR SDRAM

Components	Packages (XX)	Components	Packages (XX)
MT46V32M4XX-75	P,TG	MT46V32M4XX-5B	-
MT46V64M4XX-75	FG,P,TG	MT46V64M4XX-5B	BG,FG,P,TG
MT46V128M4XX-75	BN,FN,P,TG	MT46V128M4XX-5B	BN,FN,P,TG
MT46V256M4XX-75	P,TG	MT46V256M4XX-5B	P,TG
MT46V16M8XX-75	P,TG	MT46V16M8XX-5B	TG,P
MT46V32M8XX-75	FG,P,TG	MT46V32M8XX-5B	BG,FG,P,TG
MT46V64M8XX-75	BN,FN,P,TG	MT46V64M8XX-5B	BN,FN,P,TG
MT46V128M8XX-75	P,TG	MT46V128M8XX-5B	-
MT46V8M16XX-75	P,TG	MT46V8M16XX-5B	TG,P
MT46V16M16XX-75	BG,FG,P,TG	MT46V16M16XX-5B	BG,FG,P,TG
MT46V32M16XX-75	-	MT46V32M16XX-5B	BN,FN,P,TG
MT46V64M16XX-75	P,TG	MT46V64M16XX-5B	-

Table 2-12: Supported Unbuffered DIMMs for DDR SDRAM

Unbuffered DIMMs	Packages (X)	Unbuffered DIMMs	Packages (X)
MT4VDDT1664AX-40B	G,Y	MT8VDDT3264AX-40B	G,Y
MT4VDDT3264AX-40B	G,Y	MT9VDDT3272AX-40B	Y

Table 2-13: Supported Registered DIMMs for DDR SDRAM

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9VDDF3272X-40B	G,Y	MT18VDDF6472X-40B	D,G,Y
MT9VDDF6472X-40B	G,Y	MT18VDDF12872X-40B	DY,G,Y

Table 2-14: Supported SODIMMs for DDR SDRAM

SODIMMs	Packages (X)	SODIMMs	Packages (X)
MT4VDDT3264HX-40B	G,Y	MT9VDDT3272HX-40B	-
MT4VDDT1664HX-40B	Y	MT9VDDT6472HX-40B	G,Y
MT8VDDT3264HX-40B	-	MT9VDDT12872HX-40B	-
MT8VDDT6464HX-40B	DG,DY,G,Y		

## Hardware Tested Configurations

The frequencies shown in [Table 2-15](#) were achieved on the Virtex-4 FPGA ML461 Memory Interfaces Development Board under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based on a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 64-bit wide interface.

**Table 2-15: Hardware Tested Configurations**

Synthesis Tools		XST and Synplicity
HDL		Verilog and VHDL
FPGA Device		XC4VLX25-FF668-11
Burst Lengths		2, 4, 8
CAS Latency (CL)		2, 2.5, 3
16-bit Design		Tested on 16-bit Component "MT46V32M16XX-5B"
72-bit Design		Tested on 72-bit DIMM "MT18VDDF6472X-40B"
CL =2	Achieved Frequency Range for Component	110 MHz to 170 MHz
	Achieved Frequency Range for DIMM	110 MHz to 150 MHz
CL=2.5	Achieved Frequency Range for Component	110 MHz to 230 MHz
	Achieved Frequency Range for DIMM	110 MHz to 170 MHz
CL=3	Achieved Frequency Range for Component	110 MHz to 250 MHz
	Achieved Frequency Range for DIMM	110 MHz to 230 MHz



## Implementing DDR2 SDRAM Controllers

This chapter describes how to implement DDR2 SDRAM interfaces for Virtex-4 FPGAs generated by MIG. MIG supports two implementations of DDR2 SDRAM interfaces: Direct clocking and SerDes clocking. The Direct clocking interface supports frequencies up to 240 MHz. This design is based on XAPP702 [Ref 18]. The SerDes clocking design supports frequencies up to 300 MHz and is based on XAPP721 [Ref 22].

### 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.

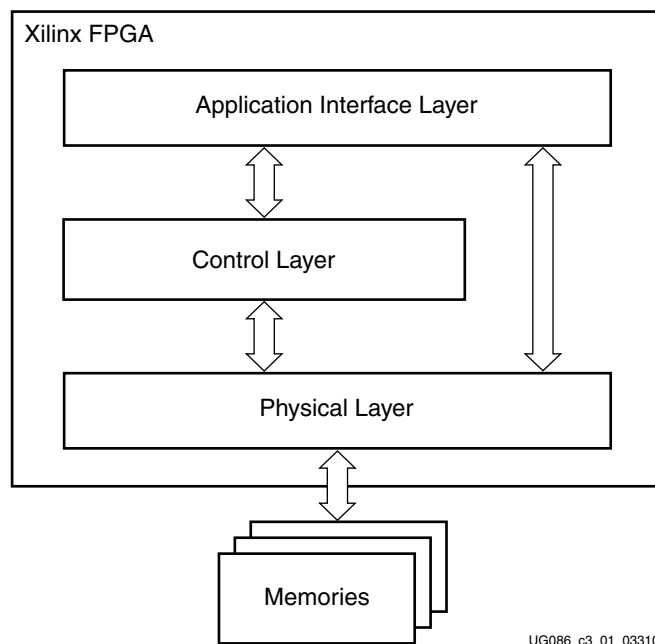


Figure 3-1: Modular Memory Interface Representation

# 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 the following:

- Burst lengths of four and eight
- Sequential and interleaved burst types
- CAS latencies of 3, 4, and 5
- Additive latencies of 0, 1, and 2
- Differential and single-ended DQS
- On-Die Termination (ODT)
- Up to four deep memories
- Memory components
- Registered DIMMs (up to 240 MHz)
- Unbuffered DIMMs (up to 200 MHz)
- Unbuffered SODIMMs (up to 200 MHz)
- Different memories (density/speed)
- Byte-wise data masking
- Precharge and auto refresh
- Linear addressing
- ECC support
- Verilog and VHDL
- With and without a testbench
- With and without a DCM
- Multicontrollers (up to eight)

The supported features are described in more detail in [“Architecture.”](#)

### Design Frequency Ranges

Table 3-1: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-10		-11		-12	
	Min	Max	Min	Max	Min	Max
Component	125	220	125	230	125	240
UDIMM/SODIMM	125	200	125	200	125	200
RDIMM	125	220	125	230	125	240
Deep Memory / Dual Rank DIMM	125	150	125	150	125	150

## Unsupported Features

The DDR2 SDRAM controller design does not support:

- Additive latencies of 3 and 4
- 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

The DDR2 SDRAM controller supports burst lengths of four and eight. The burst length can be selected through the “Set mode register(s)” option from the GUI. For a design without a testbench (user design), the user has to provide bursts of the input data based on the chosen burst length. Bits M2:M0 of the Mode Register define the burst length, and bit M3 indicates the burst type (see the Micron data sheet). Read and write accesses to the DDR2 SDRAM are burst-oriented. It determines the maximum number of column locations accessed for a given READ or WRITE command.

#### CAS Latency

The DDR2 SDRAM controller supports CAS latencies (CLs) of three and four. CL can be selected in the “Set mode register(s)” option from the GUI. The CAS latency is implemented in the `ddr2_controller` module. During data write operations, the generation of the `ctrl_Dqs_En` and `ctrl_Dqs_Rst` signals varies according to the CL in the `ddr2_controller` module. During data read operations, the generation of the `ctrl_RdEn` signal varies according to the CL in the `ddr2_controller` module. Bits M4:M6 of the Mode Register define the CL (see the Micron data sheet). CL is the delay in clock cycles between the registration of a READ command and the availability of the first bit of output data.

#### Additive Latency

DDR2 SDRAM devices support a feature called posted CAS additive latency (AL). The DDR2 SDRAM supports additive latencies of 0, 1, and 2. AL can be selected in the “Set mode register(s)” option from the GUI. Additive latency is implemented in the `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. 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 (see the Micron data sheet).

#### Registered DIMMs

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

### Unbuffered DIMMs and SODIMMs

The DDR2 SDRAM design supports unbuffered DIMMs and SODIMMs. Unbuffered DIMMs are normal DIMMs, where a set of components are used to get a particular configuration. SODIMMs differ from the unbuffered DIMMs only by the package type. Otherwise they are functionally the same.

### Multicontrollers

MIG supports multicontrollers for DDR2 SDRAMs. A maximum of eight controllers can be selected by the user from the tool. In multicontroller designs, MIG supports the same frequency for all the controllers.

### Different Memories (Density/Speed)

The DDR2 SDRAM controller supports different densities. For DDR2 components shown in MIG, densities vary from 256 Mb to 2 Gb, and DIMM densities vary from 128 Mb to 4 Gb. Higher densities can be created using the “Create new memory part” feature of MIG. The supported maximum column address is 13, the maximum row address is 15, and the maximum bank address is 3. The design can decode write and read addresses from the user in the DDR2 SDRAM controller module. The user address consists of column, row, bank, chip address, and user command.

Table 3-2 and Table 3-3 list sample timing sheets for Micron components and DIMMs, respectively.

**Table 3-2: 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 3-3: 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



Table 3-3: Timing Parameters for DIMMs (Continued)

Parameter	Description	MT4HTF		MT8HTF		MT16HTF		MT9HTF		MT18HTF	
		-53E	-40E	-53E	-40E	-53E	-40E	-53E	-40E	-53E	-40E
T <sub>RFC</sub>	REFRESH to ACTIVE or REFRESH to REFRESH command interval	128 MB	75	256 MB	75	512 MB	75	256 MB	75	512 MB	75
		75		75		75		75		75	
		256 MB	105	512 MB	105	1 GB	105	512 MB	105	1 GB	105
		105		105		105		105		105	
		512 MB	127.5	1 GB	127.5	2 GB	127.5	1 GB	127.5	2 GB	127.5
		127.5		127.5		127.5		127.5		127.5	
T <sub>RCD</sub>	ACTIVE to READ or WRITE delay	15	15	15	15	15	15	15	15	15	15
T <sub>RAS</sub>	ACTIVE to PRECHARGE command	40	40	40	40	40	40	40	40	40	40
T <sub>RC</sub>	ACTIVE to ACTIVE (same bank) command	55	55	55	55	55	55	55	55	55	55
T <sub>RTP</sub>	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 <sub>WTR</sub>	WRITE to READ command delay	7.5	10	7.5	10	7.5	10	7.5	10	7.5	10
T <sub>WR</sub>	WRITE recovery time	15	15	15	15	15	15	15	15	15	15

**Note:** For the latest timing information, refer to the vendor memory data sheets.

### Data Masking

The DDR2 SDRAM design supports data masking per byte. Masking per nibble is not supported due to the limitation of the internal block RAM based FIFOs. So, the masking of data can be done on a per byte basis. The mask data is stored in the Data FIFO along with the actual data.

### Precharge

The PRECHARGE command is used to close the open row in a bank if there is a command to be issued in the same bank. The PRECHARGE command checks the row address, bank address, and chip address, and the DDR2 Virtex<sup>TM</sup>-4 controller issues a PRECHARGE command if there is a change in any of the addresses where a read or write command is to be issued. The auto precharge function is not supported.

### Auto Refresh

The DDR2 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 there is a refresh request while there is an ongoing read or write burst, the controller issues a refresh command after completing the current read or write burst command.

### Linear Addressing

The DDR2 SDRAM controller supports linear addressing. Linear addressing refers to the way the user provides the address of the memory to be accessed. For Virtex-4 FPGA DDR2 SDRAM controllers, the user provides address information through the app\_af\_addr bus. As the densities of the memory devices vary, the number of column address bits and row

address bits also change. In any case, the row address bits in the `app_af_addr` bus always start from the next higher bit, where the column address ends. This feature increases the number of devices that can be supported with the design.

### On-Die Termination

The DDR2 SDRAM controller supports on-die termination (ODT). Through the “Set mode register(s)” option from the GUI, the user can disable ODT or can choose 75, 150, or 50. ODT can turn the termination on and off as needed to improve signal integrity in the system. Because DDR2 supports the deep memory maximum of four, a maximum of four ODTs is supported. 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

The MIG DDR2 SDRAM controller supports depths up to 4. Through the “Depth” option, the user can select various deep values. For deep memory implementations, MIG generates chip selects, CKE signals, and ODT signals for each memory. The clock widths (CK and CK\_N) are a multiple factor of the deep configuration chosen in MIG. This feature increases the depth of the memory. For example, if the user selects a 256 Mb component and deep memory = 4 from MIG, the tool generates a memory interface for a 1 Gb design.

Deep memory logic is implemented in the `ddr2_` controller module. With deep memories, DDR2 SDRAMs are initialized one after the other to avoid loading the address and control buses, and the calibration is done on the last memory. Apart from initialization, the DDR2 SDRAM controller module also demultiplexes the column, row, and bank addresses from the user address. The module also decodes the chip selects and rank addresses for components and DIMMs.

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

The DDR2 SDRAM controller supports ECC. ECC is supported for the following data widths:

- 40-bit (32-bit data and a 0 prepended to 7-bit parity)

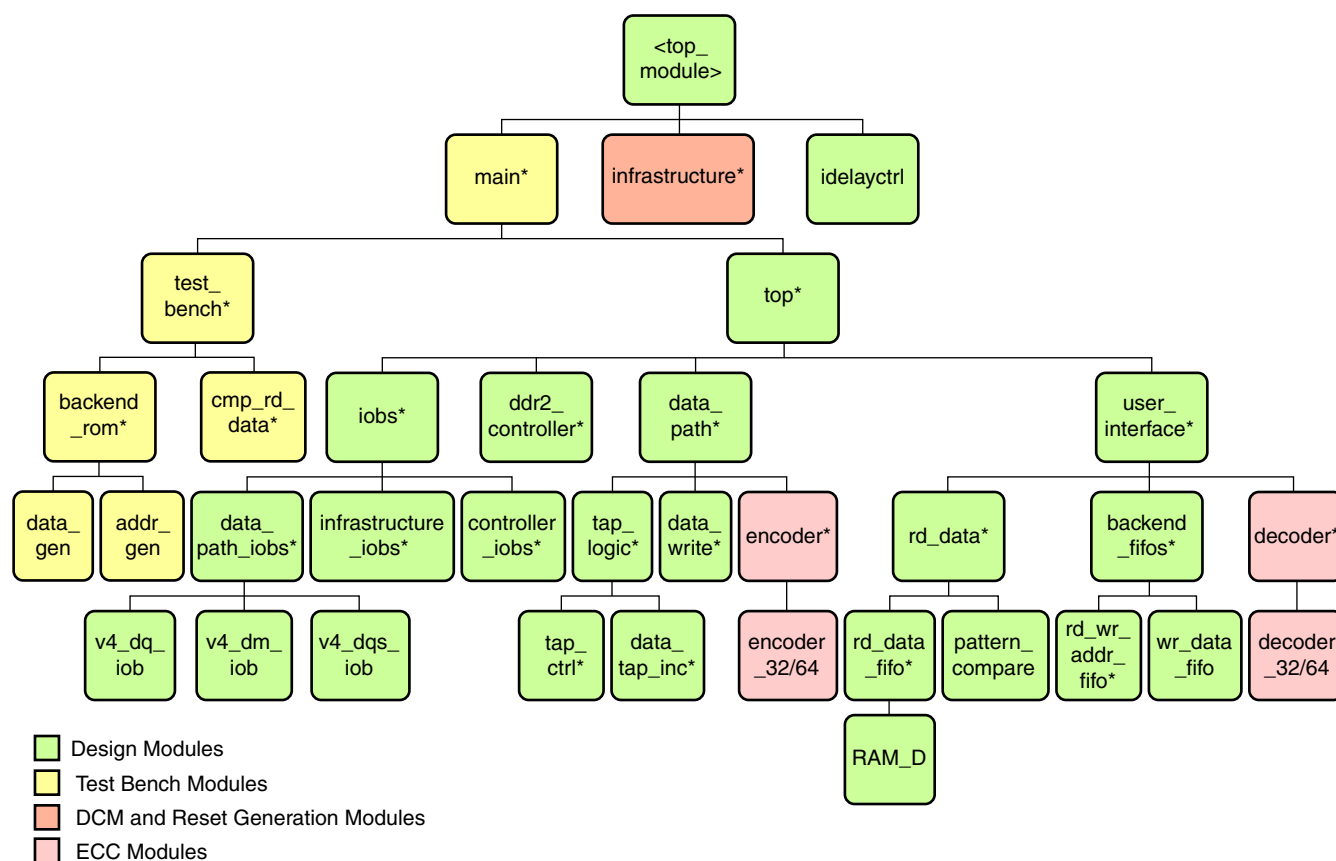
- 72-bit (64-bit data and 8-bit parity)
- 144-bit (128-bit data and 16-bit parity)

The user can completely disable the ECC or can generate the design for the above data widths by choosing either the Unpipeline mode or the Pipeline mode from the GUI.

ECC is based on XAPP645 [Ref 16]. 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. The Pipeline mode improves the frequency performance at the cost of an extra pipeline stage.

## Hierarchy

Figure 3-2 shows the hierarchical structure of the DDR2 SDRAM controller.



UG086\_c3\_03\_091107

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

Figure 3-2 shows the hierarchical structure of the DDR2 SDRAM design generated by MIG with a testbench and a DCM. The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks and reset generation modules

There is a parameter file generated with the design that has all the user input and design parameters selected from MIG.

MIG can generate four different DDR2 SDRAM designs:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

A design without a testbench (`user_design`) does not have testbench modules. The `<top_module>` module has the user interface signals for designs without a testbench. The list of user interface signals is provided in [Table 3-6, page 119](#).

Design clocks and resets are generated in the infrastructure module. The DCM clock is instantiated in the infrastructure module for designs with a DCM. The inputs to this module are the differential design clock and a 200 MHz differential clock for the `IDELAYCTRL` module. A user reset is also input to this module. Using the input clocks and reset signals, the system clocks and the system reset are generated in this module, which is used in the design.

The DCM primitive is not instantiated in this module if the **Use DCM** option is unchecked. So, the system operates on the user-provided clocks. The system reset is generated in the infrastructure module using the `DCM_LOCK` input signal.

For ECC enabled designs, the corresponding shaded modules are present in the design. ECC data is generated from these modules.

Figure 3-3 shows a top-level block diagram of a DDR2 SDRAM design with a DCM and a testbench. SYS\_CLK\_P and SYS\_CLK\_N are differential input system clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. CLK200\_P and CLK200\_N are used for the idelay\_ctrl element. SYS\_RESET\_IN\_N is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal. The ERROR output signal indicates whether the case passes or fails. The testbench module does writes and reads, and also compares the read data with the written data. The ERROR signal is driven High on data mismatches. The INIT\_DONE signal indicates the completion of initialization and calibration of the design. Memory device signals are prepended with the controller number. For example, for a single controller design, the DDR2\_RAS\_N signal appears as *cntrl0\_DDR2\_RAS\_N*. Similarly, for a four-controller design with controllers 0, 1, 2, and 3, the controller 3 DDR2\_RAS\_N signal appears as *cntrl3\_DDR2\_RAS\_N*.

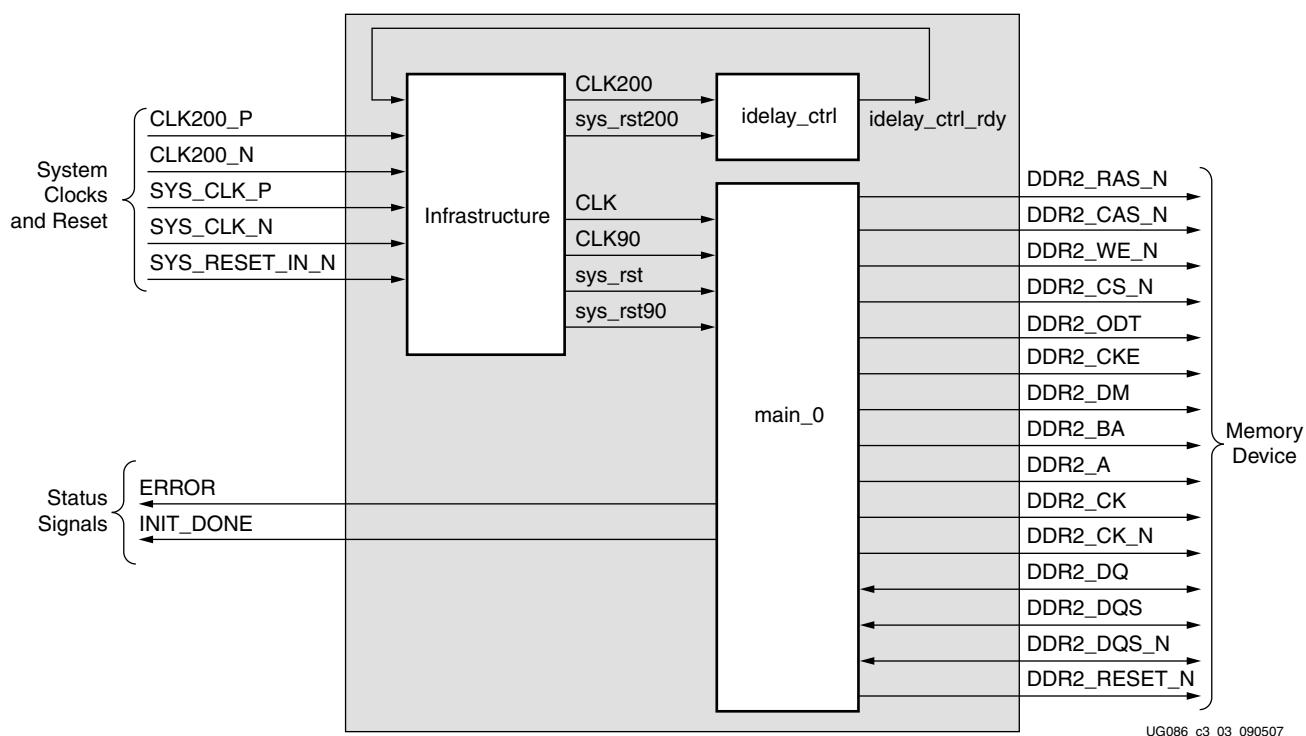
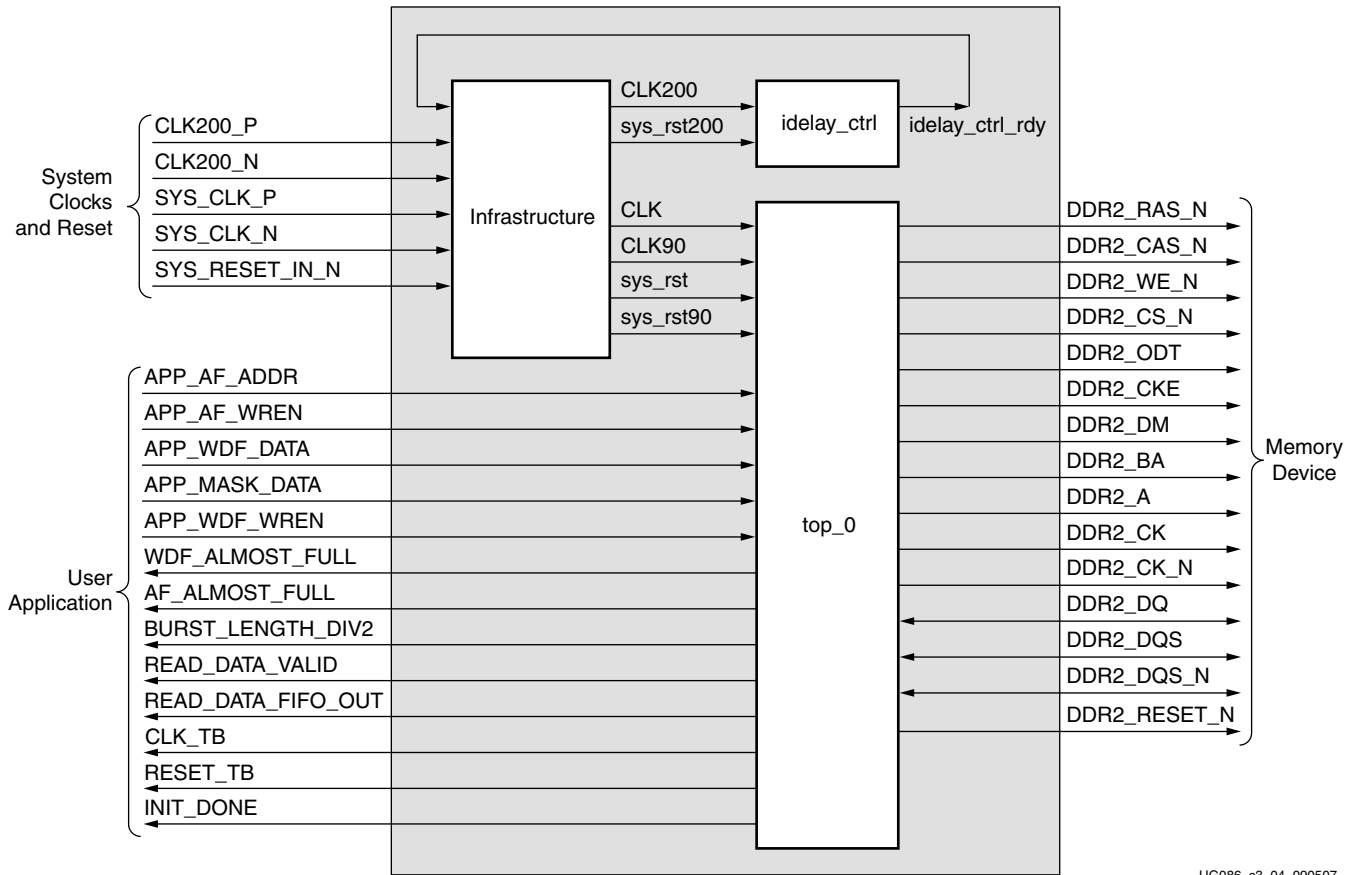


Figure 3-3: Top-Level Block Diagram of the DDR2 SDRAM Design with a DCM and a Testbench

All Memory Device ports do not necessarily appear for all MIG-generated designs. For example, port DDR2\_RESET\_N appears in the port list for Registered DIMM designs only. Similarly, DDR2\_DQS\_N does not appear for single-ended DQS designs. Port DDR2\_DM appears only for parts that contain a data mask; a few RDIMMs have no data mask, and DDR2\_DM does not appear in the port list for them.

Figure 3-4 shows a top-level block diagram of a DDR2 SDRAM design with a DCM but without a testbench. SYS\_CLK\_P and SYS\_CLK\_N are differential input system clocks. A DCM is instantiated in the infrastructure module that generates the required design clocks. CLK200\_P and CLK200\_N are used for the idelay\_ctrl element. SYS\_RESET\_IN\_N is an active-Low system reset signal. All design resets are gated by the dcm\_lock signal. The user has to drive the user application signals. The design provides the CLK\_TB and RESET\_TB signals to the user to synchronize with the design. The INIT\_DONE signal indicates the completion of initialization and calibration of the design.



UG086\_c3\_04\_090507

Figure 3-4: Top-Level Block Diagram of the DDR2 SDRAM Design with a DCM but without a Testbench

Figure 3-5 shows a top-level block diagram of a DDR2 SDRAM design without a DCM or a testbench. The user should provide all the clocks and the `dcm_lock` signal. These clocks should be single-ended. `SYS_RESET_IN_N` is an active-Low system reset signal. All design resets are gated by the `dcm_lock` signal. The user application must have a DCM primitive instantiated in the design. All user clocks should be driven through BUFs. The user has to drive the user application signals. The design provides the `CLK_TB` and `RESET_TB` signals to the user to synchronize with the design. The `INIT_DONE` signal indicates the completion of initialization and calibration of the design.

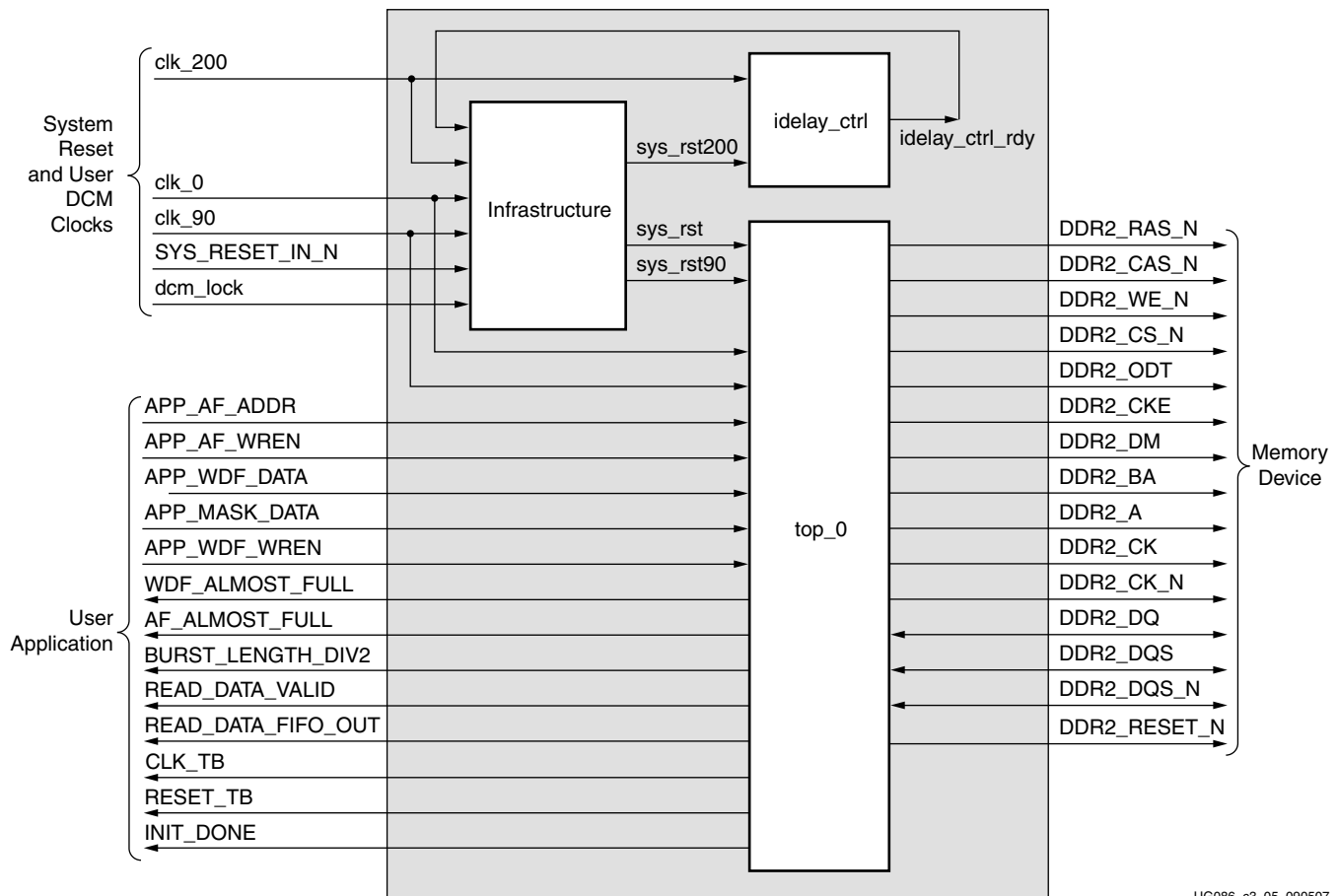
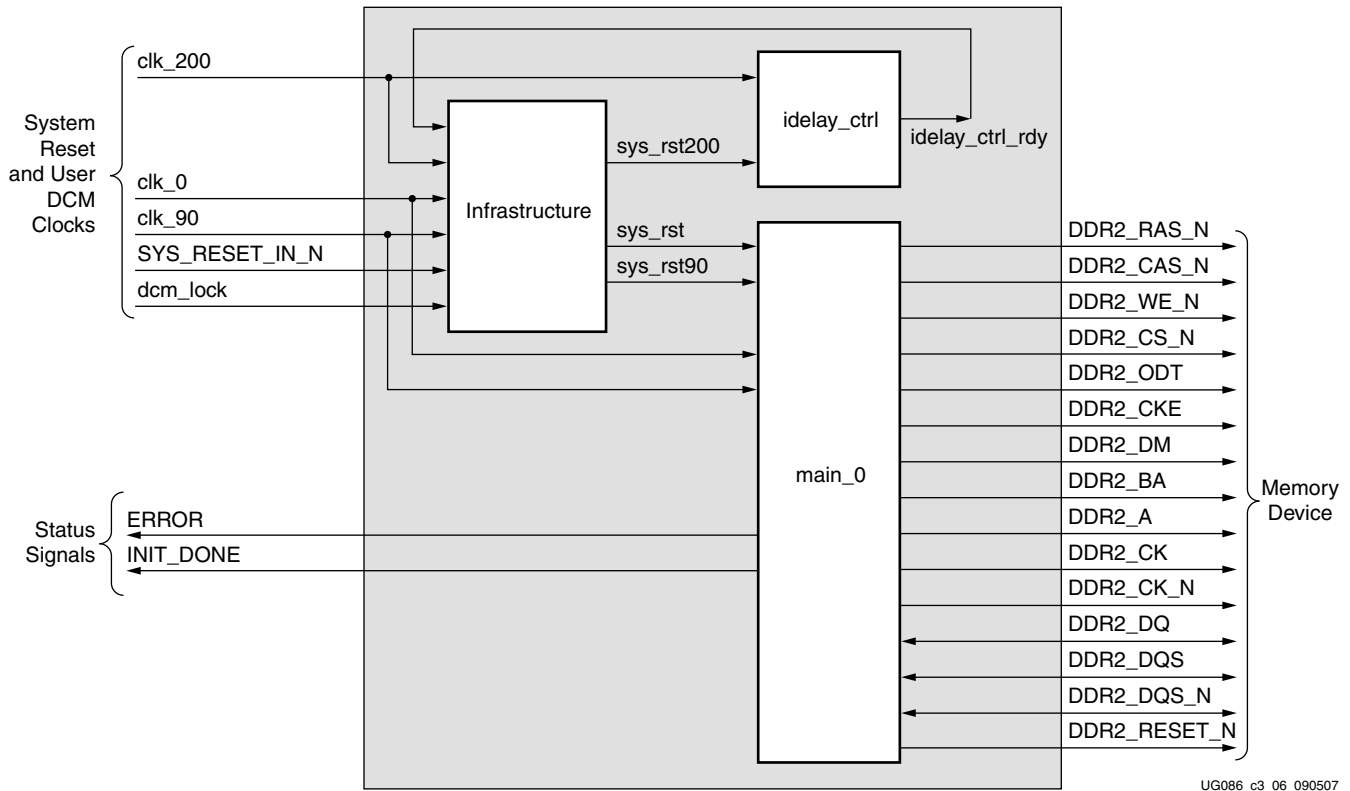


Figure 3-5: Top-Level Block Diagram of the DDR2 SDRAM Design without a DCM or a Testbench

Figure 3-6 shows a top-level block diagram of a DDR2 SDRAM design with a testbench but without a DCM. The user should provide all the clocks and the dcm\_lock signal. These clocks should be single-ended. SYS\_RESET\_IN\_N is an active-Low system reset signal. All design resets are gated by the dcm\_lock signal. The user application must have a DCM primitive instantiated in the design. All user clocks should be driven through BUFGs. The ERROR output signal indicates whether the case passes or fails. The testbench module does writes and reads, and also compares the read data with the written data. The ERROR signal is driven High on data mismatches. The INIT\_DONE signal indicates the completion of initialization and calibration of the design.



UG086\_c3\_06\_090507

Figure 3-6: Top-Level Block Diagram of the DDR2 SDRAM Design with a Testbench but without a DCM



## DDR2 Controller Submodules

Figure 3-7 is a detailed block diagram of the DDR2 SDRAM controller. The five blocks shown are the sub-blocks of the top module. User backend signals are provided by the tool for designs with a testbench. The user has to drive these signals for designs without a testbench. The functions of these blocks are explained in the subsections following the figure.

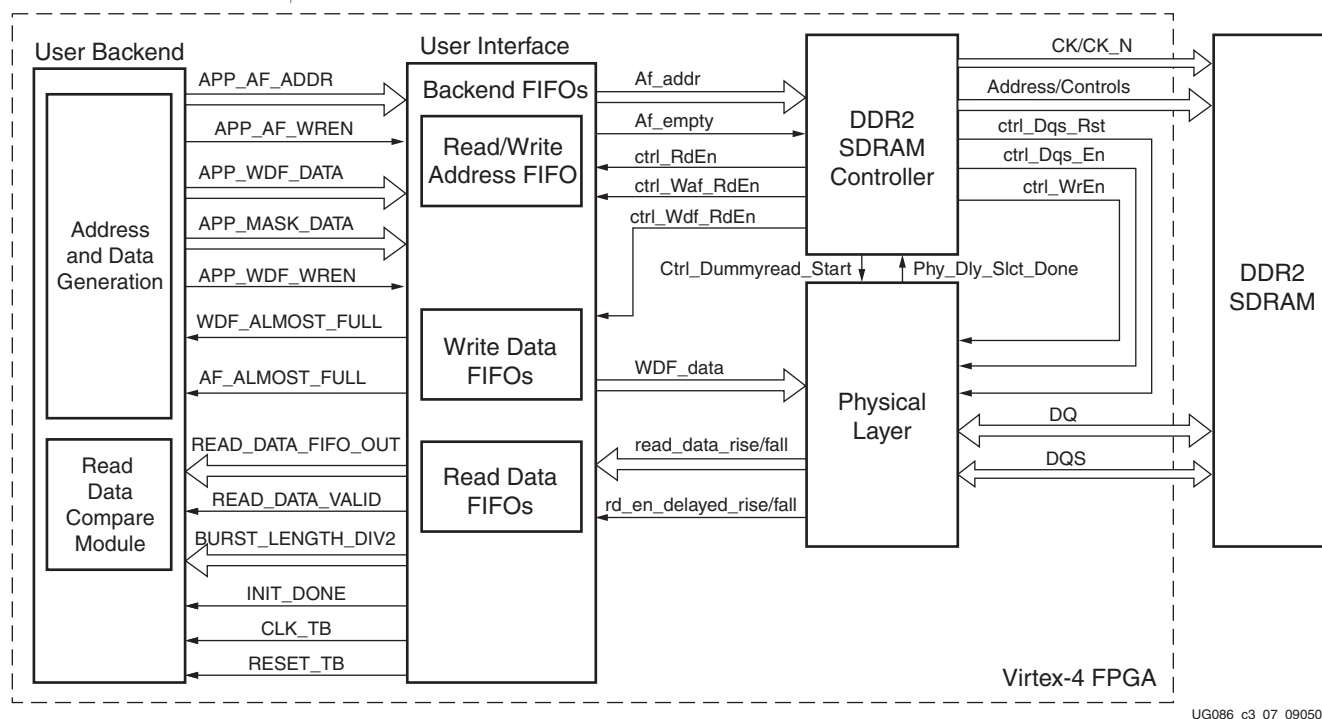


Figure 3-7: DDR2 Memory Controller Block Diagram

### 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 and delays the data to center-align with the FPGA clock. After the calibration is done, the controller issues a dummy write and pattern read commands to get the delay between the read command and IOB output data.

The delay, calculated in number of clocks, is used as a write-enable signal to the read data FIFOs. For deep designs, the DQ calibration and pattern calibration are done only on the last memory. For example, for four deep designs, the fourth memory is used for calibration. There is no reason to use the fourth memory because the controller retains the last chip select during initialization of memory. Thus the same chip select is used for calibration. XAPP701 [Ref 17] provides more details about the calibration architecture.

### 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 or the pattern\_compare4 module. During the pattern\_read command, the controller generates the ctrl\_rden signal, which is delayed in the pattern\_compare 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\_compare 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.

## DDR2 SDRAM Initialization and Calibration

DDR2 memory is initialized through a specified sequence as per both Micron and JEDEC specifications. The controller starts the memory initialization at power-up. Following the initialization, the relationship between the data and the FPGA clock is calculated using the tap\_logic. The controller issues a dummy write command and a dummy read command to the memory and compares read data with the fixed pattern. During dummy reads, the tap\_logic module calibrates and delays the data to center-align with the FPGA clock.

The sel\_done port in the tap\_logic module indicates the completion of the per-bit calibration. After the per-bit calibration is done, the controller does a read enable calibration. This calibration is used to determine the delay from read command to read data at rd\_data\_fifo. The delay between read command and read data is affected by the CAS latency and additive latency parameters, the PCB traces, and the I/O buffer delays. This in turn is used to generate a write enable to rd\_data\_fifo so that valid data is registered. The controller writes a known fixed pattern and reads back the data. The read data is compared against the known fixed pattern. The comp\_done port in the rd\_data module indicates the completion of the read enable calibration.

The init\_done port indicates the completion of both per-bit calibration and read enable calibration. After initialization and calibration is done, the controller can start issuing user commands to the memory.

## DDR2 SDRAM System and User Interface Signals

Table 3-4 describes the DDR2 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 provided to the design. They must have a phase difference of 180° with respect to each other. Similarly, CLK200\_P and CLK\_200N are 200 MHz differential clocks for the IDELAYCTRL module. SYS\_RESET\_IN\_N resets all the logic.

Table 3-4: DDR2 SDRAM Controller System Interface Signals (with a DCM)

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_IN_N	Input	Active-Low reset to the DDR2 controller.

Table 3-5 shows the system interface signals for designs without a DCM. clk\_0, clk\_90, and clk\_200 are single-ended input clocks. The clk\_90 signal must have a phase difference of 90° with respect to clk\_0. The clk\_200 signal is the clock used for the IDELAYCTRL primitives in Virtex-4 FPGAs.

Table 3-5: DDR2 SDRAM Controller System Interface Signals (without a DCM)

Signal	Direction	Description
clk_0	Input	The DDR2 SDRAM controller and memory operates on this clock.
SYS_RESET_IN_N	Input	Active-Low reset to the DDR2 SDRAM controller. This signal is used to generate the synchronous system reset.
clk_90	Input	90° phase-shifted clock with the same frequency as clk_0.
clk_200	Input	200 MHz input differential clock for the IDELAYCTRL primitive of Virtex-4 FPGAs.
dcm_lock	Input	This status signal indicates whether the DCM is locked or not. It is used to generate the synchronous system reset.

Table 3-6 describes the DDR2 SDRAM user interface signals.

Table 3-6: DDR2 SDRAM Controller User Interface Signals

Signal Name <sup>(1)</sup>	Direction	Description
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_DIV2[2:0]	Output	This signal determines the data burst length for each write address. 010: burst length = 4 100: burst length = 8
WDF_ALMOST_FULL	Output	This signal indicates the ALMOST_FULL status of the Write Data FIFO. When this signal is asserted, the user can write 5 more locations into the FIFO in designs generated with a testbench and 14 more locations in designs without a testbench.
APP_WDF_DATA[2n-1:0]	Input	User write data to the memory, where $n$ indicates the data width of the interface. The user data is twice the data width of the interface. 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 interface. The mask data is twice the mask width of the interface. The most-significant bits contain the rising-edge mask data and the least-significant bits contain the falling-edge mask data. These signals are not present when the memory part does not have mask support (for example, certain Registered DIMMs) or when the Data Mask option is not selected in the MIG GUI.
APP_WDF_WREN	Input	Write Enable signal to the Write Data FIFO.
AF_ALMOST_FULL	Output	This signal indicates the ALMOST_FULL status of the Address FIFO. The user can issue eight more locations into the FIFO after AF_ALMOST_FULL is asserted.

Table 3-6: DDR2 SDRAM Controller User Interface Signals (Continued)

Signal Name <sup>(1)</sup>	Direction	Description
APP_AF_ADDR[35:0]	Input	The user address consists of a memory address and dynamic commands. Bits [31:0] are the memory read/write address. Bits [31:0] form the memory chip select, bank address, row address, and column address.  Bit 35 is reserved for internal use of the controller. Bits [34:32] represent the following dynamic commands:  001: Auto Refresh 010: Precharge 100: Write 101: Read  Other combinations are invalid. Functionality of the controller is unpredictable for unimplemented commands.
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 <i>n</i> indicates the data width of the interface. 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.
INIT_DONE	Output	This signal indicates the completion of initialization and calibration of the design.

**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 MIG. 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.

### User Interface Accesses

The user backend logic communicates with the memory controller through a FIFO-based user interface. This interface consists of three related buses:

- A Command/Address FIFO bus, which accepts write/read commands as well as the corresponding memory address from the user
- A Write Data FIFO bus, which accepts the corresponding write data when the user issues a write command on the Command/Address bus
- A Read bus on which the corresponding read data for an issued read command is returned

The user interface has the following timing and signaling restrictions:

- Commands and write data cannot be written by the user until calibration is complete (as indicated by INIT\_DONE). In addition, the following interface signals need to be held Low until calibration is complete: APP\_AF\_WREN, APP\_WDF\_WREN, APP\_WDF\_DATA, and APP\_MASK\_DATA. Failure to hold these signals Low causes errors during calibration. This restriction arises from the fact that the Write Data FIFO is used during calibration to hold the training patterns for the various stages of calibration.

- When issuing a write command, the first write data word must be written to the Write Data FIFO no more than two clock cycles after the write command is issued. This restriction arises from the fact that the controller assumes write data is available when it receives the write command from the user.

## Write Interface

Figure 3-8 shows the user interface block diagram for write operations.

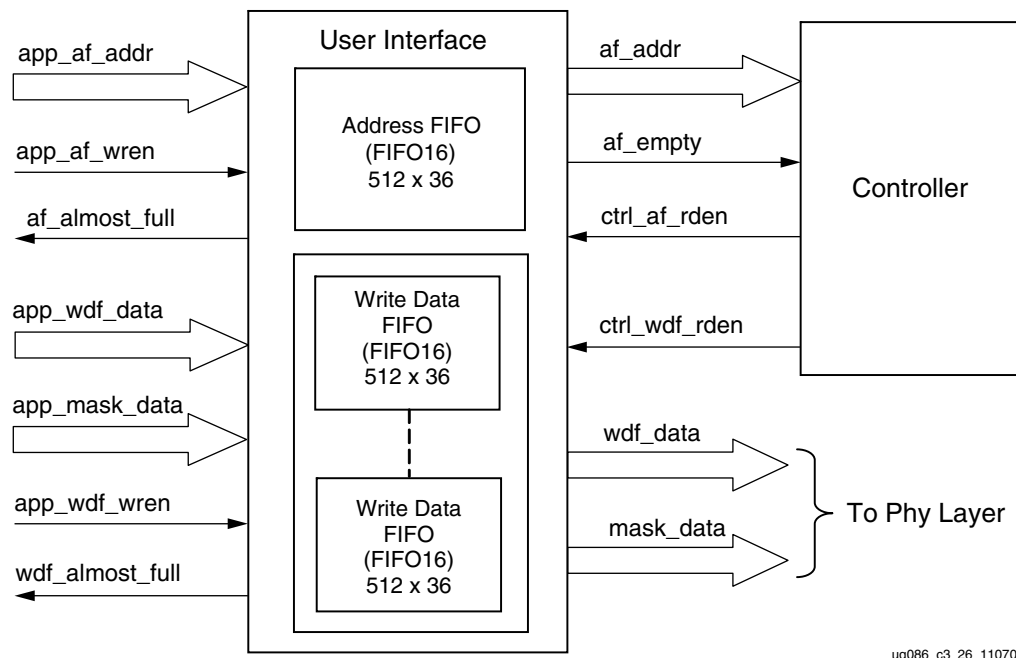


Figure 3-8: User Interface Block Diagram for Write Operations

The following steps describe the architecture of the Address and Write Data FIFOs and show how to perform a write burst operation to DDR2 SDRAM from the user interface.

1. The user interface consists of an Address FIFO and a Write Data FIFO. These FIFOs are constructed using Virtex-4 FIFO16 primitives with a 512 x 36 configuration. The 36-bit architecture comprises one 32-bit port and one 4-bit port. For Write Data FIFOs, the 32-bit port is used for data bits and the 4-bit port is used for mask bits. Mask bits are available only when supported by the memory part *and* when Data Mask is enabled in the MIG GUI. Some memory parts, such as Registered DIMMs of x4 parts, do not support mask bits.
2. The Common Address FIFO is used for both write and read commands, and comprises a command part and an address part. Command bits discriminate between write and read commands.
3. User interface data width app\_wdf\_data is twice that of the memory data width. For an 8-bit memory width, the user interface is 16 bits consisting of rise data and fall data. For every 8 bits of data, there is a mask bit. For 72-bit memory data, the user interface data width app\_wdf\_data is 144 bits, and the mask data app\_mask\_data is 18 bits.
4. The minimum configuration of the Write Data FIFO is 512 x 36 for a memory data width of 8 bits. For an 8-bit memory data width, the least-significant 16 bits of the data port are used for write data and the least-significant two bits of the 4-bit port are used

for mask bits. The controller internally pads all zeros for the most-significant 16 bits of the 32-bit port and the most-significant two bits of the 4-bit port.

5. Depending on the memory data width, MIG instantiates multiple FIFO16s to gain the required width. For designs using 8-bit data width, one FIFO16 is instantiated; for 72-bit data width, a total of five FIFO16s are instantiated. The bit architecture comprises 16 bits of rising-edge data, 2 bits of rising-edge mask, 16 bits of falling-edge data, and 2 bits of falling-edge mask, which are all stored in a FIFO16. MIG routes the `app_wdf_data` and `app_mask_data` to FIFO16s accordingly.
6. The user can initiate a write to memory by writing to the Address FIFO and the Write Data FIFO when the FIFO Full flags are deasserted and after the `init_done` signal is asserted. Status signal `af_almost_full` is asserted when Address FIFO is full, and similarly `wdf_almost_full` is asserted when Write Data FIFO is full.
7. Both the Address FIFO and Write Data FIFO Full flags are deasserted with power-on.
8. The user should assert the Address FIFO write-enable signal `app_af_wren` along with address `app_af_addr` to store the write address and write command into the Address FIFO.
9. The user should assert the Data FIFO write-enable signal `app_wdf_wren` along with write data `app_wdf_data` and mask data `app_mask_data` to store the write data and mask data into the Write Data FIFO. The user should provide both rise and fall data together for each write to the Data FIFO.
10. The controller reads the Address FIFO by issuing the `ctrl_af_rden` signal. The controller reads the Write Data FIFO by issuing the `ctrl_wdf_rden` signal after the Address FIFO is read. It decodes the command part after the Address FIFO is read.

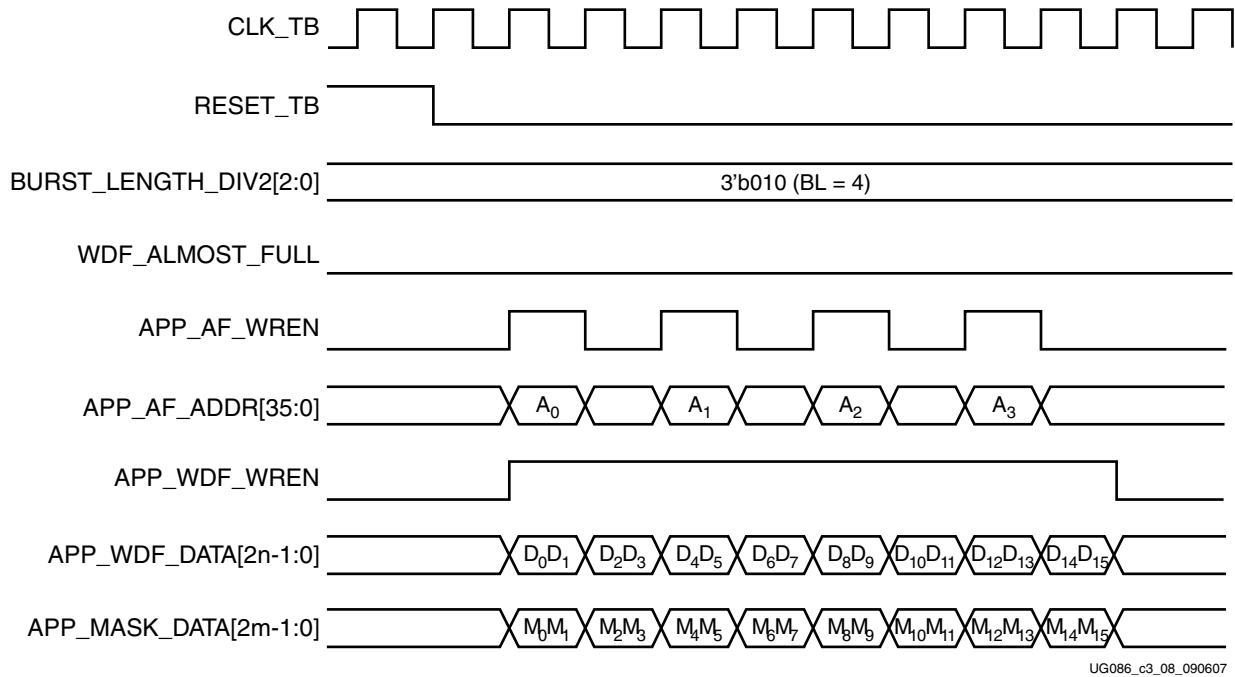


Figure 3-9: DDR2 SDRAM Write Burst (BL = 4) for Four Bursts

11. The write command timing diagram in Figure 3-9 is derived from the MIG-generated test bench. As shown (burst length of 4), each write to the Address FIFO must be coupled with *two* writes to the Data FIFO. Similarly, for a burst length of 8, every write

to the Address FIFO must be coupled with *four* writes to the Data FIFO. Failure to follow this rule can cause unpredictable behavior.

**Note:** The user can start filling the Write Data FIFO two clocks after the Address FIFO is written, because there is a two-clock latency between the command fetch and reading the Data FIFO. Using the terms shown in [Figure 3-10](#), therefore, the user can assert the A0 address two clocks before D0D1.

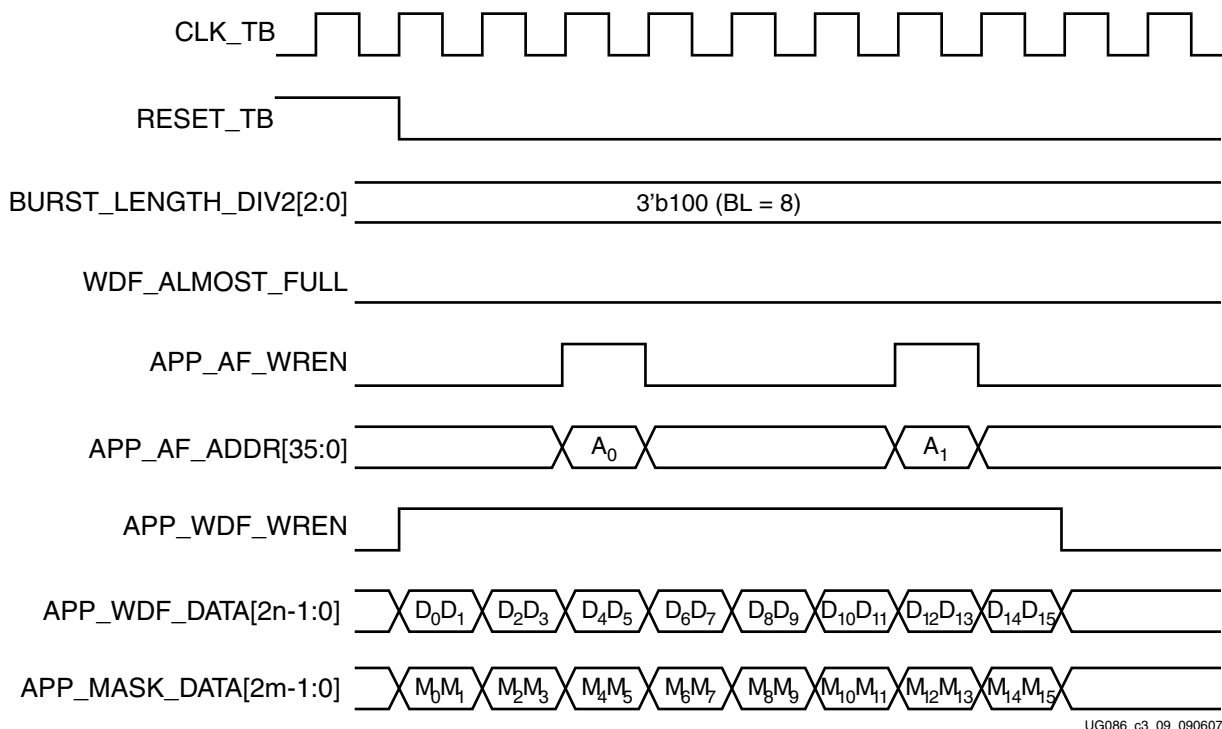


Figure 3-10: DDR2 SDRAM Write Burst (BL = 8) for Two Bursts

- The write command timing diagram in [Figure 3-10](#) is derived from the MIG-generated test bench. As shown (burst length of 8), each write to the Address FIFO must be coupled with *four* writes to the Data FIFO. Because the controller first reads the address and command together, the address need not coincide with the last data. After the command is analyzed (nearly two clocks later for a worst-case timing scenario), the controller sequentially reads the data in four clocks. Thus, there are six clocks from the time the address is read to the time the last data is read.

## Correlation between the Address and Data FIFOs

There is a worst-case two-cycle latency from the time the address is loaded into the address FIFO on APP\_AF\_ADDR[35:0] to the time the controller decodes the address. Because of this latency, it is not necessary to provide the address on the last clock where data is entered into the data FIFO. If the address is written before the last data phase, the overall efficiency and performance increases because it eliminates or reduces the two-cycle latency. However, if the address is written before data is input into the data FIFO, a FIFO empty condition might result because the Data FIFO does not contain valid data.

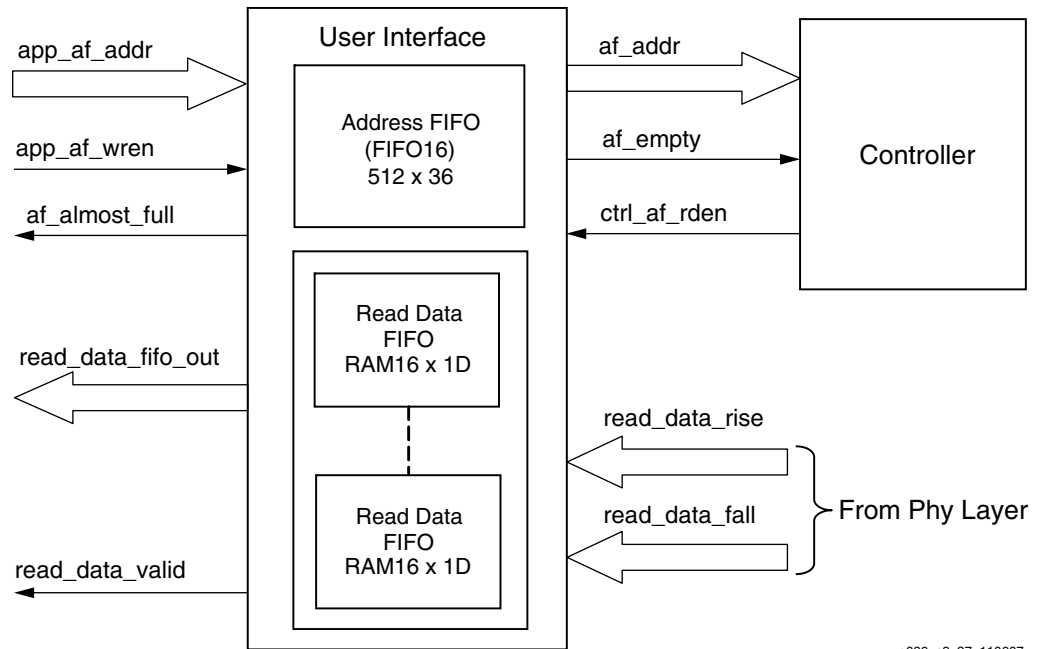
Based on these considerations, Xilinx recommends entering the address into the address FIFO between the first data phase and the next-to-last data phase. For a burst of four or eight, this means the Address can be asserted two clocks before the first data phase. This



implementation increases efficiency by reducing the two clock latency and guarantees that valid data is available in the Data FIFO.

## Read Interface

Figure 3-11 shows a block diagram of the read interface.



ug086\_c3\_27\_110607

Figure 3-11: User Interface Block Diagram for Read Operation

The following steps describe the architecture of the Read Data FIFOs and show how to perform a burst read operation from DDR SDRAM from the user interface.

1. The read user interface consists of an Address FIFO and a Read Data FIFO. The Address FIFO is common to both read and write operations. The Read Data FIFOs are constructed using Virtex-4 Distributed RAMs with a 16 x 1 configuration. MIG instantiates a number of RAM16Ds depending on the data width. For example, for 8-bit data width, MIG instantiates a total of 16 RAM16Ds, 8 for rising-edge data and 8 for falling-edge data. Similarly, for 72-bit data width, MIG instantiates a total of 144 RAM16Ds, 72 for rising-edge data and 72 for falling-edge data.
2. The user can initiate a read to memory by writing to the Address FIFO when the FIFO Full flag `af_almost_full` is deasserted and after `init_done` is asserted.
3. To write the read address and read command into the Address FIFO, the user should issue the Address FIFO write-enable signal `app_af_wren` along with read address `app_af_addr`.
4. The controller reads the Address FIFO containing the address and command. After decoding the command, the controller generates the appropriate control signals to memory.
5. Prior to the actual read and write commands, the design calibrates the latency (number of clock cycles) from the time the read command is issued to the time data is received. Using this pre-calibrated delay information, the controller generates the write-enable signals to the Read Data FIFOs.
6. The `read_data_valid` signal is asserted when data is available in the Read Data FIFOs.



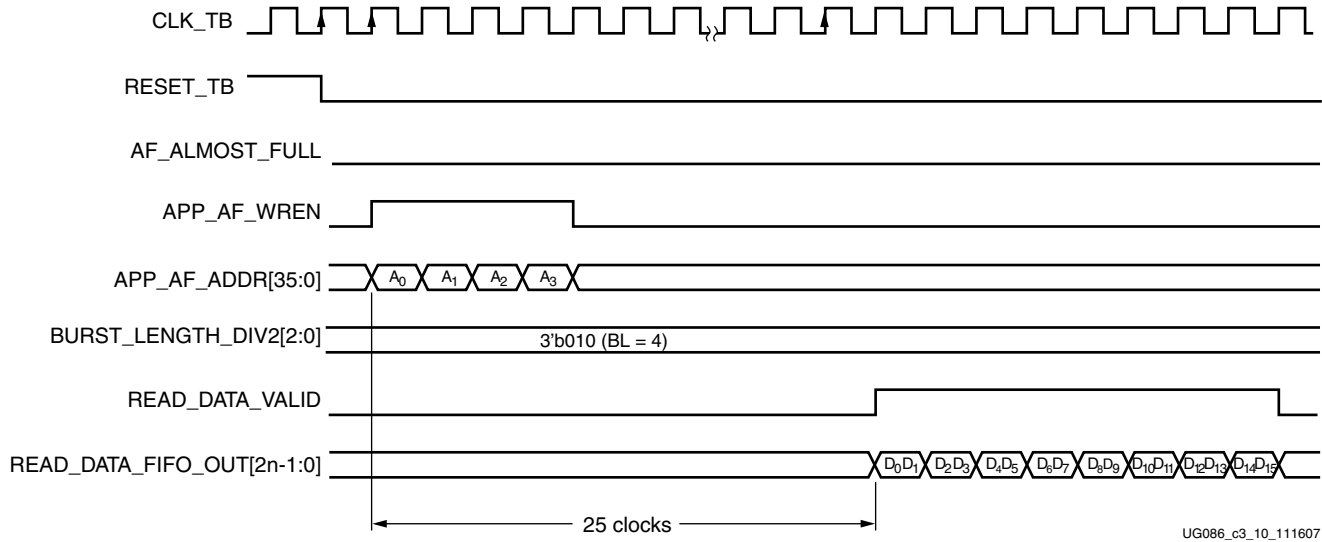


Figure 3-12: DDR2 SDRAM Read Burst (BL = 4) for Four Bursts

7. Figure 3-12 shows the user interface timing diagram for a burst length of 4, and Figure 3-13 shows the user interface timing diagram for a burst length of 8. Both the cases shown here are for a CAS latency of 3 at 200 MHz. The read latency is calculated from the point when the read command is given by the user to the point when the data is available with the read\_data\_valid signal. The minimum latency in this case is 25 clocks, where no precharge is required, no auto-refresh request is pending, the user commands are issued after initialization is completed, and the first command issued is a Read command. Controller executes the commands only after initialization is done as indicated by the init\_done signal.
8. After the address and command are loaded into the Address FIFO, it takes 25 clock cycles minimum for the controller to assert the read\_data\_valid signal.
9. Read data is available only when the read\_data\_valid signal is asserted. The user should access the read data on every positive edge of the read\_data\_valid signal.

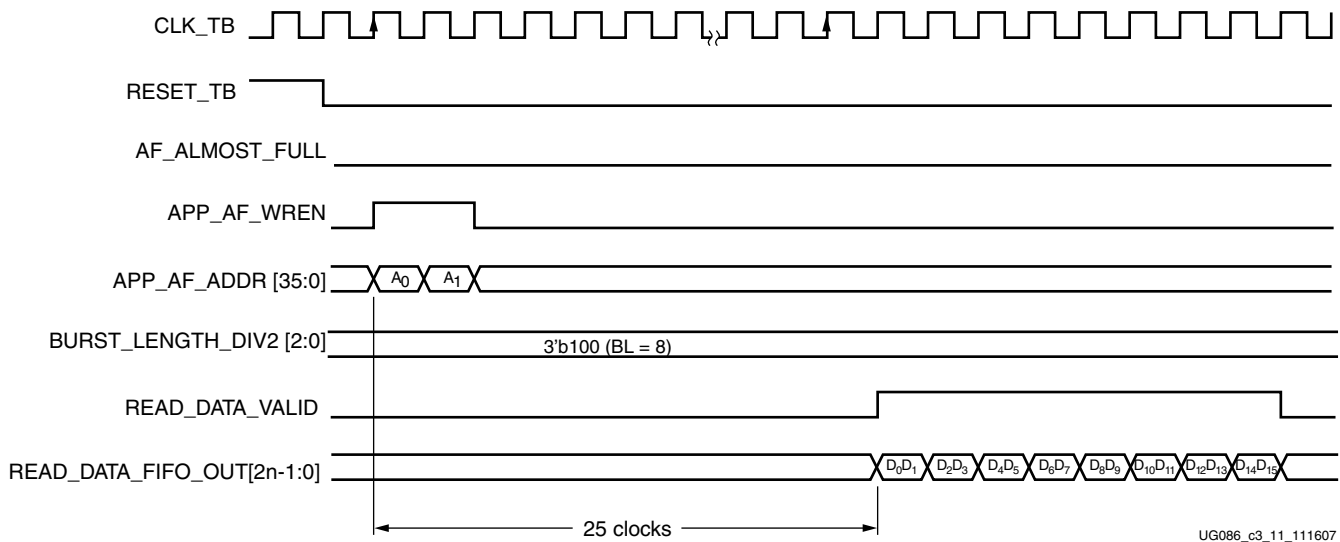


Figure 3-13: DDR2 SDRAM Read Burst (BL = 8) for Two Bursts

The 25 clocks from the read command to the read data, as shown in [Figure 3-12](#) and [Figure 3-13](#), are broken up as indicated in [Table 3-7](#).

**Table 3-7: Read Command to Read Data Clock Cycles**

Parameter	Number of Clocks
Read Address to Empty Deassert	7 clocks
Empty to Active Command	5.5 clocks
Active to Read Command	3 clocks
Memory Read Command to Read Data Valid	9.5 clocks
<b>Total:</b>	<b>25 clocks</b>

In general, read latency varies based on the following parameters:

- CAS latency (CL) and additive latency (AL)
- The number of commands already in the FIFO pipeline before the read command is issued
- Whether an ACTIVATE command needs to be issued to open the new bank/row
- Whether a PRECHARGE command needs to be issued to close a previously opened bank
- Specific timing parameters for the memory, such as  $T_{RAS}$  and  $T_{RCD}$  in conjunction with the bus clock frequency
- Commands might be interrupted, and banks/rows might be forcibly closed when the periodic AUTO REFRESH command is issued
- If the user issues the commands before initialization is complete, the latency cannot be determined
- Board-level and chip-level (for both memory and FPGA) propagation delays

## User to Controller Interface

[Table 3-8](#) lists the signals between the User interface and the controller.

**Table 3-8: 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. The mapping of these address bits is: [31:0]: Memory Address (CS, Bank, Row, Column) [34:32]: Dynamic Command Request [35]: Reserved	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

Table 3-8: List of Signals Between User Interface and Controller (Continued)

Port Name	Port Width	Port Description	Notes
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. <a href="#">Figure 3-15</a> shows the timing waveform for burst length of eight 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 two clock cycles for a burst length of four and four clock cycles for a burst length of eight. <a href="#">Figure 3-15</a> 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 four, 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 (af\_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

```
[column_address + row_address - 1:column_address]
```

#### Bank Address

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

#### Chip Select

```
[column_address + row_address + bank_address + chip_address - 1:column_address + row_address + bank_address]
```

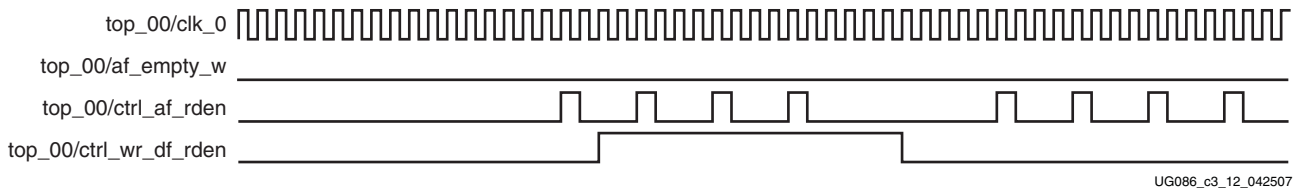
### Dynamic Command Request

[Table 3-9](#) 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 3-9: Optional Commands

Command	Description
001	Auto Refresh
010	Precharge
100	Write
101	Read

Figure 3-14 describes four consecutive writes followed by four consecutive reads with a burst length of 8.



UG086\_c3\_12\_042507

Figure 3-14: Consecutive Writes Followed by Consecutive Reads with Burst Length of 8  
Controller to Physical Layer Interface

Table 3-10 lists the signals between the controller and the physical layer.

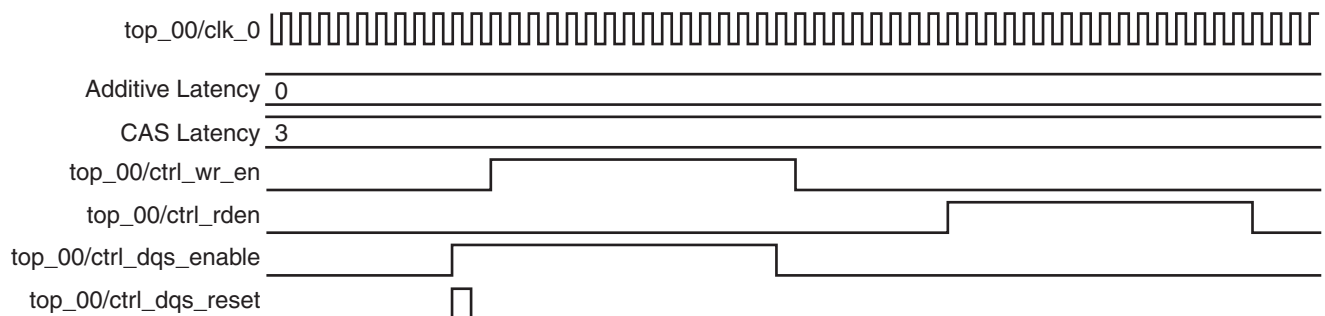
Table 3-10: Signals Between the Controller and Physical Layer

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 deasserted when the 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 3-15 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 3-10: 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 3-15 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 3-15 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 3-15 describes the timing waveform for control signals from the controller to the physical layer.



UG086\_c3\_13\_042507

Figure 3-15: Timing Waveform for Control Signals from the Controller to the Physical Layer

## Deep Memory Configurations

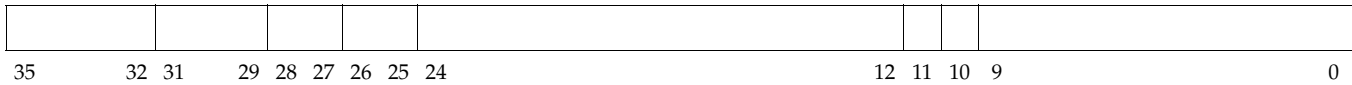
The following examples provide user address formats for different densities of components and DIMMs in deep memory designs. These are examples only, not associated with any specific memory part number from memory data sheets.

### 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:



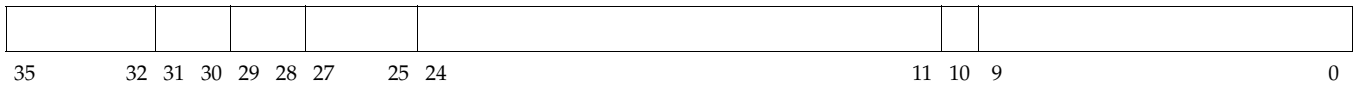
A10-A0	Column address
A23-A11	Row address
A25-A24	Bank address
A27 -A26	Rank + deep address
A31-A28	Assigned to all zeros
A34-A32	Dynamic commands
A35	Reserved for internal use



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
A21-A9	Row address
A23-A22	Bank address
A24	Rank + deep address
A31-A25	Assigned to all zeros
A34-A32	Dynamic commands
A35	Reserved for internal use

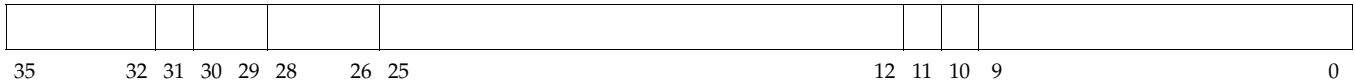




Case 2: (8 GB)

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:



A10-A0	Column address
A24-A11	Row address
A27-A25	Bank address
A29-A28	Rank + deep address
A31-A30	Assigned to zeros
A34 - A32	Dynamic commands
A35	Reserved for internal use

Table 3-11 is an example showing the pin mapping for x4 registered DIMMs between the memory data sheet and the user constraint file (UCF).

Table 3-11: Pin Mapping for x4 DIMMs

Memory Data Sheet	MIG 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]

MIG allows banks to be selected for different classes of memory signals. When a particular bank is checked for an address, MIG allocates the memory address, the memory control, and the memory clocks in that bank. When a bank is checked for data, MIG allocates the data, the data mask, and the data strobes in that bank. When a bank is checked for system control, MIG allocates the system reset and status signals in that bank. When a bank is checked for system clocks, MIG allocates the system clock signals in that bank.

Table 3-12 shows the list of signals allocated in a group from bank selection check boxes.

Table 3-12: Direct Clocking DDR2 SDRAM Signal Allocations

Bank Selected	Signals Allocated in the Group
Address	Memory address, memory control, and memory clock signals
Data	Data, data mask, and data strobes
System Control	System reset from user interface and status signals
System_Clock	System clocks from the user interface

## Simulating the DDR2 SDRAM Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains the external testbench, memory model, `.do` file, and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more details about the files in `sim` folder and to simulate the design, see the `simulation_help.chm` file in `sim` folder.

## Changing the Refresh Rate

Change the global ``define` (for Verilog) or constant (for VHDL) variable `MAX_REF_CNT` in `mymodule_parameters_0.v` (or `.vhd`) so that  $MAX\_REF\_CNT = (\text{refresh interval in clock periods}) = (\text{refresh interval}) / (\text{clock period})$ . For example, for a refresh rate of  $3.9 \mu\text{s}$  with a memory bus running at 200 MHz:

$$MAX\_REF\_CNT = 3.9 \mu\text{s} / (\text{clock period}) = 3.9 \mu\text{s} / 5 \text{ ns} = 780 \text{ (decimal)} = 0x30C$$

If the above value exceeds  $2^{MAX\_REF\_WIDTH} - 1$ , the value of `MAX_REF_WIDTH` must be increased accordingly in `parameters_0.v` (or `.vhd`) to increase the width of the counter used to track the refresh interval.

## Supported Devices

The design generated out of MIG is independent of memory package, hence the package part of the memory component is replaced with `XX` or `XXX`, where `XX` or `XXX` indicates a don't care condition. The tables below list the components (Table 3-13) and DIMMs (Table 3-14 through Table 3-16) supported by the tool for DDR2 Direct clocking designs. In supported devices, an `X` in the components column (for Components and Unbuffered DIMMs) denotes a single alphanumeric character. For example `MT47H128M4XX-3` can be either `MT47H128M4BP-3` or `MT47H128M4B6-3`. Similarly `MT16HTF25664AX-40E` can be either `MT16HTF25664AY-40E` or `MT16HTF25664AG-40E`. An `XX` for Registered DIMMs denotes a single or two alphanumeric characters. For example, `MT9HTF3272XX-667` can be either `MT9HTF3272Y-667` or `MT9HTF3272DY-667`. An `XXX` for Registered DIMMs denotes two or three alphanumeric characters. For example, `MT18HTF12872XXX-667` can be either `MT18HTF12872DY-667` or `MT18HTF12872PDY-667`.

Table 3-13: Supported Components for DDR2 SDRAM

Components	Packages (XX)	Components	Packages (XX)
MT47H64M4XX-3	BP	MT47H128M8XX-3	BT,HQ
MT47H64M4XX-37E	BP	MT47H128M8XX-37E	BT,HQ
MT47H64M4XX-5E	BP	MT47H128M8XX-5E	BT,HQ
MT47H128M4XX-3	B6,CB,GB	MT47H256M8XX-3	HG
MT47H128M4XX-37E	B6,CB,GB	MT47H256M8XX-37E	HG
MT47H128M4XX-5E	B6,CB,GB	MT47H256M8XX-5E	HG
MT47H256M4XX-3	BT,HQ	MT47H16M16XX-3	BG
MT47H256M4XX-37E	BT,HQ	MT47H16M16XX-37E	BG
MT47H256M4XX-5E	BT,HQ	MT47H16M16XX-5E	BG
MT47H512M4XX-3	HG	MT47H32M16XX-3	BN,CC,FN,GC
MT47H512M4XX-37E	HG	MT47H32M16XX-37E	BN,CC,FN,GC
MT47H512M4XX-5E	HG	MT47H32M16XX-5E	BN,CC,FN,GC
MT47H32M8XX-3	BP	MT47H64M16XX-3	BT,HR
MT47H32M8XX-37E	BP	MT47H64M16XX-37E	BT,HR
MT47H32M8XX-5E	BP	MT47H64M16XX-5E	BT,HR
MT47H64M8XX-3	B6,CB,F6,GB	MT47H128M16XX-3	HG
MT47H64M8XX-37E	B6,CB,F6,GB	MT47H128M16XX-37E	HG
MT47H64M8XX-5E	B6,CB,F6,GB	MT47H128M16XX-5E	--

Table 3-14: Supported Registered DIMMs for DDR2 SDRAM

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9HTF3272XX-667	--	MT18HTF25672XX-667	PDY,PY,Y
MT9HTF3272XX-53E	Y	MT18HTF25672XX-53E	PDY,PY,Y
MT9HTF3272XX-40E	Y	MT18HTF25672XX-40E	DY,PDY,Y
MT9HTF6472XX-667	PY,Y	MT18HTF6472XXX-667	--
MT9HTF6472XX-53E	Y	MT18HTF6472XXX-53E	DY,Y
MT9HTF6472XX-40E	Y	MT18HTF6472XXX-40E	DY,Y
MT9HTF12872XX-667	PY	MT18HTF12872XXX-667	DY,PDY,PY,Y
MT9HTF12872XX-53E	PY,Y	MT18HTF12872XXX-53E	DY,MY,NDY, NY,PY,Y
MT9HTF12872XX-40E	Y	MT18HTF12872XXX-40E	DY,PY,Y
MT18HTF6472G-53E	--	MT18HTF25672XXX-667	PDY,PY,Y
MT18HTF6472XX-667	--	MT18HTF25672XXX-53E	PDY,PY,Y
MT18HTF6472XX-53E	DY,Y	MT18HTF25672XXX-40E	DY,PDY,Y
MT18HTF6472XX-40E	DY,Y	MT36HTJ51272XX-667	--
MT18HTF12872XX-667	DY,PDY,PY,Y	MT36HTJ51272XX-53E	Y
MT18HTF12872XX-53E	DY,MY,NDY, NY,PY,Y	MT36HTJ51272XX-40E	Y
MT18HTF12872XX-40E	DY,PY,Y	--	--

**Table 3-15: Supported Unbuffered DIMMs for DDR2 SDRAM**

Unbuffered DIMMs	Unbuffered DIMMs
MT4HTF1664AY-667	MT8HTF12864AY-667
MT4HTF1664AY-40E	MT8HTF12864AY-40E
MT4HTF3264AY-667	MT9HTF3272AY-667
MT4HTF3264AY-40E	MT9HTF3272AY-40E
MT4HTF6464AY-667	MT9HTF6472AY-667
MT4HTF6464AY-40E	MT16HTF25664AX-40E
MT8HTF6464AY-667	MT18HTF6472AY-40E
MT8HTF6464AY-53E	MT18HTF12872AY-40E
MT8HTF6464AY-40E	MT18HTF25672AY-40E

**Table 3-16: Supported SODIMMs for DDR2 SDRAM**

SODIMMs	SODIMMs
MT4HTF1664HY-667	MT8HTF3264HY-53E
MT4HTF1664HY-53E	MT8HTF3264HY-40E
MT4HTF1664HY-40E	MT8HTF6464HY-667
MT4HTF3264HY-667	MT8HTF6464HY-53E
MT4HTF3264HY-53E	MT8HTF6464HY-40E
MT4HTF3264HY-40E	MT8HTF3264HDY-40E
MT8HTF3264HY-667	MT8HTF6464HDY-40E

## Hardware Tested Configurations

The frequencies shown in [Table 3-17](#) were achieved on the Virtex-4 FPGA ML461 Memory Interfaces Development Board under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based on a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 64-bit wide interface.

**Table 3-17: Hardware Tested Configurations**

Synthesis Tools	XST and Synplicity
HDL	Verilog and VHDL
FPGA Device	XC4VLX25-FF668-11
Burst Lengths	4, 8
CAS Latency (CL)	3, 4
Additive Latency	0, 1, 2
8-bit Design	Tested on 16-bit Component "MT47H32M16XX-3"
72-bit Design	Tested on 72-bit DIMM "MT9HTF6472XX-667"
ECC with Pipelined Mode	72-bit Registered DIMM design

Table 3-17: Hardware Tested Configurations (Continued)

Synthesis Tools	XST and Synplicity
Frequency Range	110 MHz to 270 MHz for CL = 3
	110 MHz to 300 MHz for CL = 4 or 5

## 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 the frequency of the interface using the ISERDES. The write data and strobe transmitted by the FPGA use the OSERDES. The OSERDES 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 and Interleaved burst types
- CAS latencies of 4 and 5
- Different memories (density/speed)
- Components
- Additive latencies 0, 1, and 2
- Verilog and VHDL
- Differential and single-ended DQS
- Linear addressing
- Without a testbench
- On Die Termination (ODT)
- DIMMs (registered DIMMs up to 300 MHz and unbuffered DIMMs up to 266 MHz)

The supported features are described in more detail in [“Architecture.”](#)

## Design Frequency Ranges

Table 3-18: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-10		-11		-12	
	Min	Max	Min	Max	Min	Max
Component	200	230	200	266	200	300
Registered DIMM	200	230	200	266	200	300
Unbuffered DIMM	200	230	200	266	200	266

## Unsupported Features

The DDR2 SDRAM controller design does not support:

- CAS latency of 3
- Additive latencies of 3 and 4
- Redundant DQS (RDQS)
- Auto precharge
- Deep memories
- ECC support
- Without a DCM
- Multicontroller

## Architecture

### Implemented Features

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

#### Burst Length

The DDR2 SDRAM controller supports burst lengths of four and eight. The burst length can be selected through the “Set mode register(s)” option in MIG. For a design without a testbench (user design), the user has to provide bursts of the input data based on the chosen burst length. Bits M2:M0 of the Mode Register define the burst length, and bit M3 indicates the burst type (see the Micron data sheet). Read and write accesses to the DDR2 SDRAM are burst-oriented. It determines the maximum number of column locations accessed for a given READ or WRITE command.

#### CAS Latency

The DDR2 SDRAM controller supports CAS latencies (CLs) of four and five. CL can be selected in the “Set mode register(s)” option from the GUI. The CAS latency is implemented in the `ddr2_controller` module. During data write operations, the generation of the `ctrl_WrEn`, `ctrl_WrEn_Dis`, and `ctrl_Odd_Latency` signals varies according to the CL in the `ddr2_controller` module. During data read operations, the generation of the `ctrl_RdEn_div0` signal varies according to the CL in the `ddr2_controller` module. Bits M4:M6 of the Mode Register define the CL (see the Micron data sheet). CL is the delay in clock cycles between the registration of a READ command and the availability of the first bit of output data.



### Additive Latency

DDR2 SDRAM devices support a feature called posted CAS additive latency (AL). The DDR2 SDRAM supports additive latencies of 0, 1, and 2. AL can be selected in the “Set mode register(s)” option. Additive latency is implemented in the ddr2\_controller module. The ddr2\_controller module issues READ/WRITE commands prior to  $t_{\text{RCD}}$  (minimum) depending on the user-selected AL value in the Extended Mode Register. This feature allows the READ command to be issued prior to  $t_{\text{RCD}}$  (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 (see the Micron data sheet).

### Registered DIMMs

DDR2 SDRAM supports registered DIMMs. This feature is implemented in the ddr2\_controller module. For registered DIMMs, the address and command signals are registered at the DIMM and therefore have one additional clock latency than unbuffered DIMMs.

### Unbuffered DIMMs and SODIMMs

The DDR2 SDRAM design supports unbuffered DIMMs and SODIMMs. Unbuffered DIMMs do not have registers at the DIMM for address and command signals. SODIMMs differ from the unbuffered DIMMs only by the package type; otherwise they are functionally the same.

### Different Memories (Density/Speed)

The DDR2 SDRAM controller supports different densities. For DDR2 components shown in MIG, densities vary from 256 Mb to 1 Gb, and DIMM densities vary from 128 Mb to 4 Gb. The user can select various configurations using the “Create new memory part” feature of MIG. The supported maximum column address is 13, the maximum row address is 15, and the maximum bank address is 3. The design can decode write and read addresses from the user in the DDR2 SDRAM controller module. The user address consists of column, row, bank, chip address, and user command.

[Table 3-19](#) and [Table 3-20](#) list sample timing sheets for Micron components and DIMMs, respectively.

**Table 3-19: Timing Parameters for Components**

Parameter	Description	Micron 256 Mb		Micron 512 Mb		Micron 1 Gb		Units
		-37E	-3	-37E	-3	-37E	-3	
$T_{\text{MRD}}$	LOAD MODE command cycle time	2	2	2	2	2	2	$T_{\text{CK}}$
$T_{\text{RP}}$	PRECHARGE command period	15	15	15	15	15	15	ns
$T_{\text{RFC}}$	REFRESH to ACTIVE or REFRESH to REFRESH command interval	75	75	105	105	127.5	127.5	ns
$T_{\text{RCD}}$	ACTIVE to READ or WRITE delay	15	15	15	15	15	15	ns
$T_{\text{RAS}}$	ACTIVE to PRECHARGE command	40	40	40	40	40	40	ns
$T_{\text{RC}}$	ACTIVE to ACTIVE (same bank) command	55	55	55	55	55	55	ns
$T_{\text{RTP}}$	READ to PRECHARGE command delay	7.5	7.5	7.5	7.5	7.5	7.5	ns

Table 3-19: Timing Parameters for Components (Continued)

Parameter	Description	Micron 256 Mb		Micron 512 Mb		Micron 1 Gb		Units
		-37E	-3	-37E	-3	-37E	-3	
$T_{WTR}$	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

Table 3-20: 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 ns	2 ns	2 ns	2 ns	2 ns	2 ns	2 ns	2 ns	2 ns	2 ns
$T_{RP}$	PRECHARGE command period	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns
$T_{RFC}$	REFRESH time	128 MB 75 ns	75 ns	256 MB 75 ns	75 ns	512 MB 75 ns	75 ns	256 MB 75 ns	75 ns	512 MB 75 ns	75 ns
		256 MB 105 ns	105 ns	512 MB 105 ns	105 ns	1 GB 105 ns	105 ns	512 MB 105 ns	105 ns	1 GB 105 ns	105 ns
		512 MB 127.5 ns	127.5 ns	1 GB 127.5 ns	127.5 ns	2 GB 127.5 ns	127.5 ns	1 GB 127.5 ns	127.5 ns	2 GB 127.5 ns	127.5 ns
$T_{RCD}$	ACTIVE to READ or WRITE delay	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns
$T_{RAS}$	ACTIVE to PRECHARGE command	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns	40 ns
$T_{RC}$	ACTIVE to ACTIVE command (same bank)	55 ns	55 ns	55 ns	55 ns	55 ns	55 ns	55 ns	55 ns	55 ns	55 ns
$T_{RTP}$	READ to PRECHARGE command delay	7.5 ns	7.5 ns	7.5 ns	7.5 ns	7.5 ns	7.5 ns	7.5 ns	7.5 ns	7.5 ns	7.5 ns
$T_{WTR}$	WRITE to READ command delay	7.5 ns	10 ns	7.5 ns	10 ns	7.5 ns	10 ns	7.5 ns	10 ns	7.5 ns	10 ns
$T_{WR}$	WRITE recovery time	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns	15 ns

**Notes:**

- For the latest timing information, refer to the vendor memory data sheets.

### Data Masking

The DDR2 SDRAM design supports data masking per byte. Masking per nibble is not supported due to the limitation of the internal block RAM based FIFOs. So, the masking of data can be done on per byte basis. The mask data is stored in the Data FIFO along with the actual data.

### Precharge

The PRECHARGE command is used to close the open row in a bank if there is a command to be issued to a different row in the same bank. The PRECHARGE command checks the row address, bank address, and chip address, and the DDR2 Virtex-4 controller issues a PRECHARGE command if there is a change in any address where a read or write command is to be issued. The auto-precharge function is not supported.

### Auto Refresh

The DDR2 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 there is a refresh request while there is an ongoing read or write burst, the controller issues a REFRESH command after completing the current read or write burst command.

### Linear Addressing

The DDR2 SDRAM controller supports linear addressing. Linear addressing refers to the way the user provides the address of the memory to be accessed. For Virtex-4 DDR2 SDRAM controllers, the user provides the address information through the `app_af_addr` signal. As the densities of the memory devices vary, the number of column address bits and row address bits also change. In any case, the row address bits in the `app_af_addr` signal always start from the next higher bit, where the column address ends. This feature increases the number of devices that can be supported with the design.

### On-Die Termination

The DDR2 SDRAM controller supports on-die termination (ODT). Through the “Set mode register(s)” option from the GUI, the user can disable ODT or can choose 75, 150, or 50. ODT can turn the termination on and off as needed to improve the signal integrity in the system. ODT is only enabled on writes to DDR2 memory. It is disabled on read operations.

## Hierarchy

Figure 3-16 shows the hierarchical structure of the DDR2 SDRAM controller.

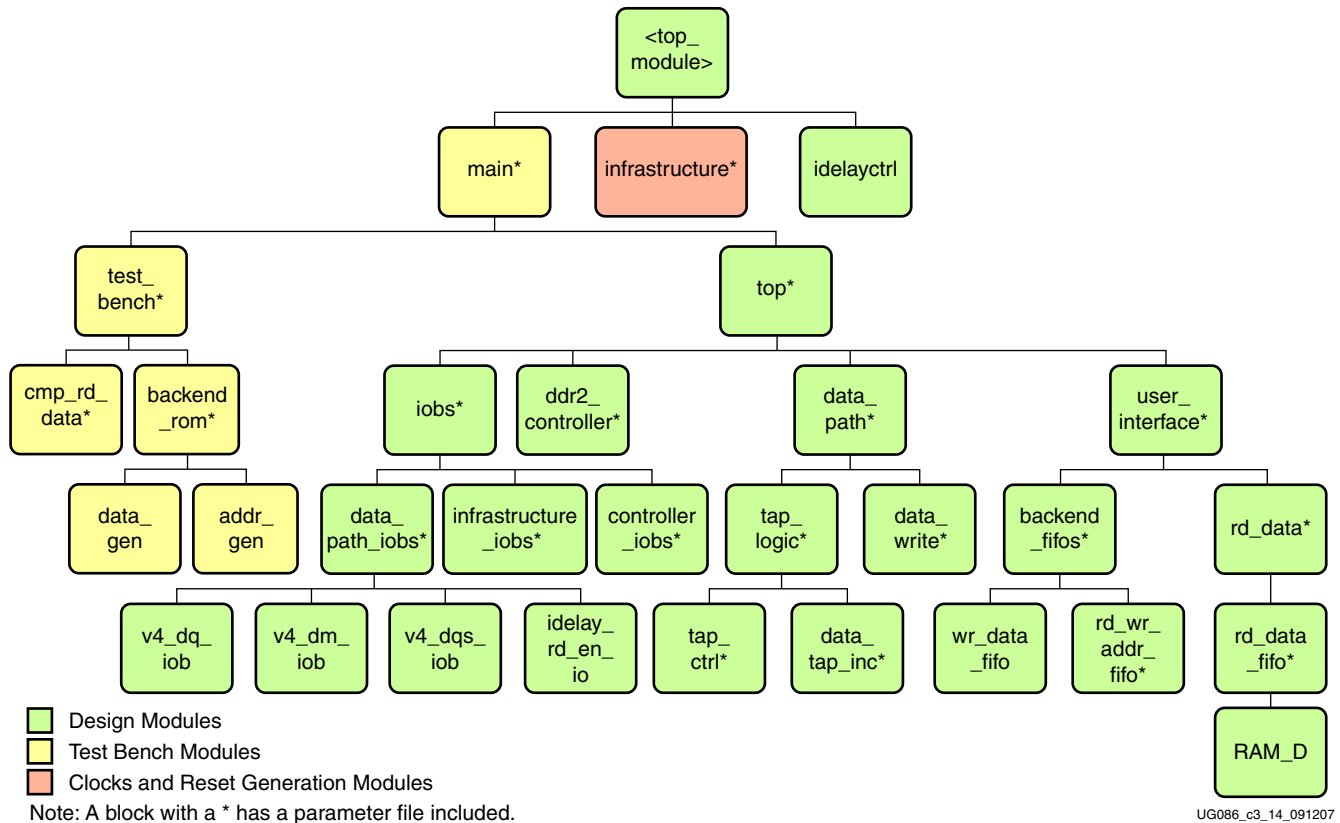


Figure 3-16: Hierarchical Structure of the DDR2 SDRAM Design (SerDes Clocking)

Figure 3-16 shows the hierarchical structure of the DDR2 SDRAM design generated by MIG with a testbench and a DCM. The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks and reset generation modules

There is a parameter file generated with the design that has all the user input and design parameters selected from MIG.

MIG can generate two different DDR2 SDRAM designs:

- With a testbench and a DCM
- Without a testbench and with a DCM

A design without a testbench (user\_design) does not have testbench modules. The <top\_module> module has the user interface signals for designs without a testbench. The list of user interface signals is provided in Table 3-22.

Design clocks and resets are generated by using the DCM in the infrastructure module. The inputs to this module are the differential design clock and a 200 MHz differential clock for the IDELAYCTRL module. A user reset is also input to this module. Using the input clocks and reset signals, the system clocks and the system reset are generated in this module, which is used in the design.

Figure 3-17 shows a top-level block diagram of a DDR2 SDRAM design with a DCM and a testbench. SYS\_CLK\_P and SYS\_CLK\_N are differential input system clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. CLK200\_P and CLK200\_N are used for the idelay\_ctrl element. SYS\_RESET\_IN\_N is an active-Low system reset signal. All design resets are gated by the dcm\_lock signal. The ERROR output signal indicates whether a read passes or fails. The testbench module issues writes and reads, and also compares the read data with the written data. The ERROR signal is driven High on data mismatches. The INIT\_COMPLETE signal indicates the completion of initialization and calibration of the design. Memory device signals are prepended with the controller number. For example, the DDR2\_RAS\_N signal appears as *ctrl0\_DDR2\_RAS\_N*.

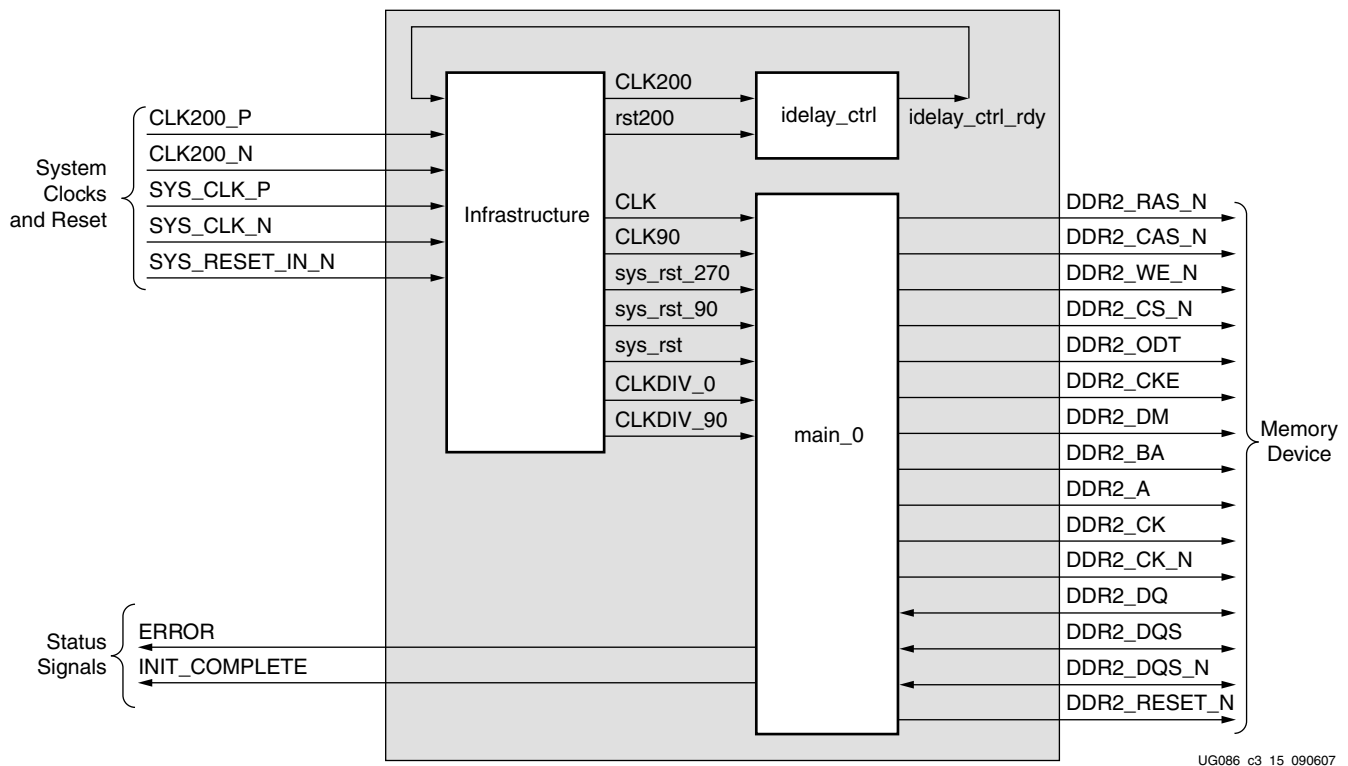
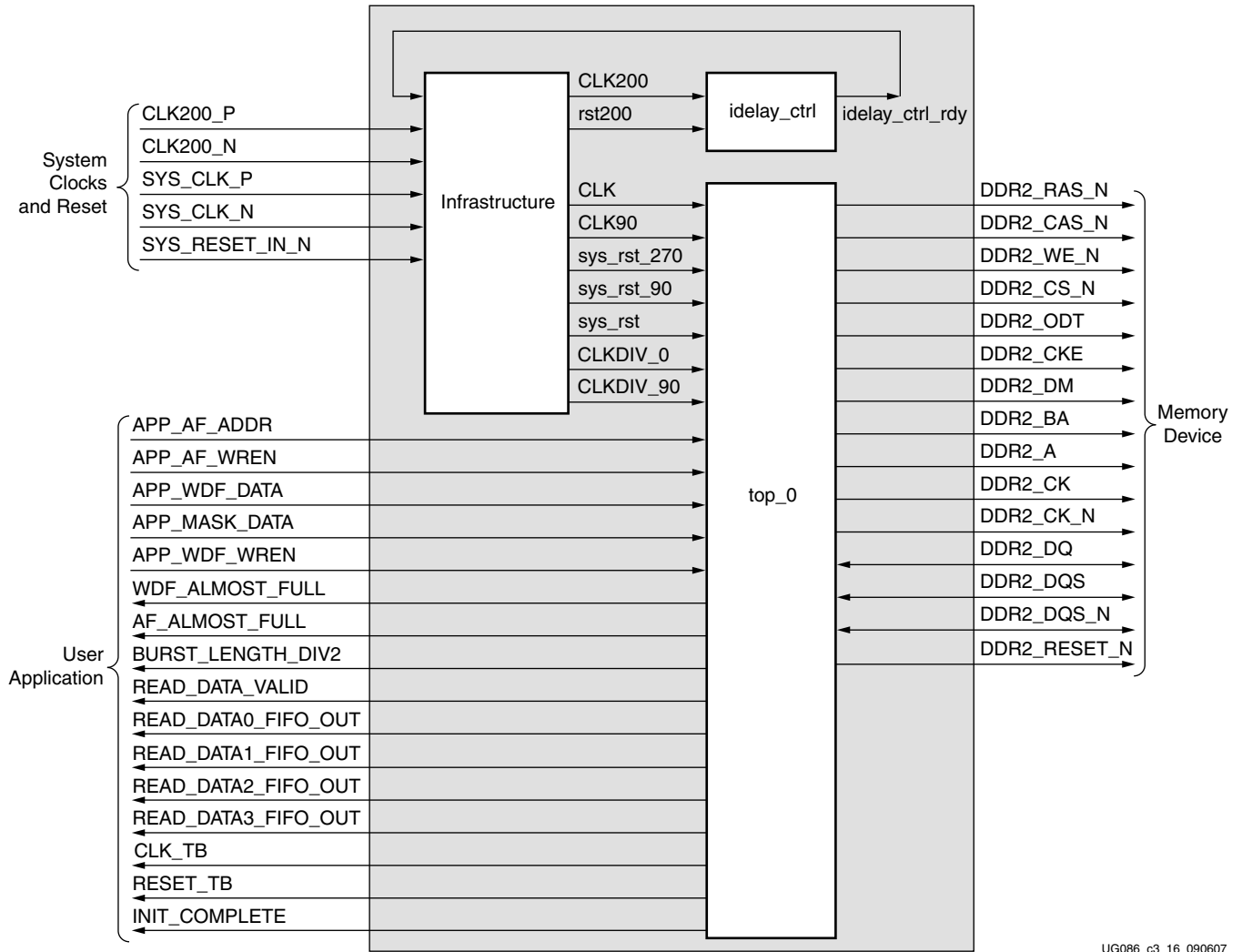


Figure 3-17: Top-Level Block Diagram of the DDR2 SDRAM Design with a DCM and a Testbench

All Memory Device ports do not necessarily appear for all MIG-generated designs. For example, port DDR2\_RESET\_N appears in the port list for Registered DIMM designs only. Similarly, DDR2\_DQS\_N does not appear for single-ended DQS designs. Port DDR2\_DM appears only for parts that contain a data mask; a few RDIMMs have no data mask, and DDR2\_DM does not appear in the port list for them.

Figure 3-18 shows a top-level block diagram of a DDR2 SDRAM design with a DCM but without a testbench. SYS\_CLK\_P and SYS\_CLK\_N are differential input system clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. CLK200\_P and CLK200\_N are used for the idelay\_ctrl element. SYS\_RESET\_IN\_N is an active-Low system reset signal. All design resets are gated by the dcm\_lock signal. The user has to drive the user application signals. The design provides the clk\_tb and reset\_tb signals to the user to synchronize with the design. The INIT\_COMPLETE signal indicates the completion of initialization and calibration of the design.



UG086\_c3\_16\_090607

Figure 3-18: Top-Level Block Diagram of the DDR2 SDRAM Design with a DCM but without a Testbench

## DDR2 Controller Submodules

Figure 3-19 is a detailed block diagram of the DDR2 SDRAM controller. The five blocks shown are the sub-blocks of the top module. The user backend signals are provided by the tool for designs with a testbench. The user has to drive these signals for designs without a testbench. The functions of these blocks are explained in the subsections following Figure 3-19.

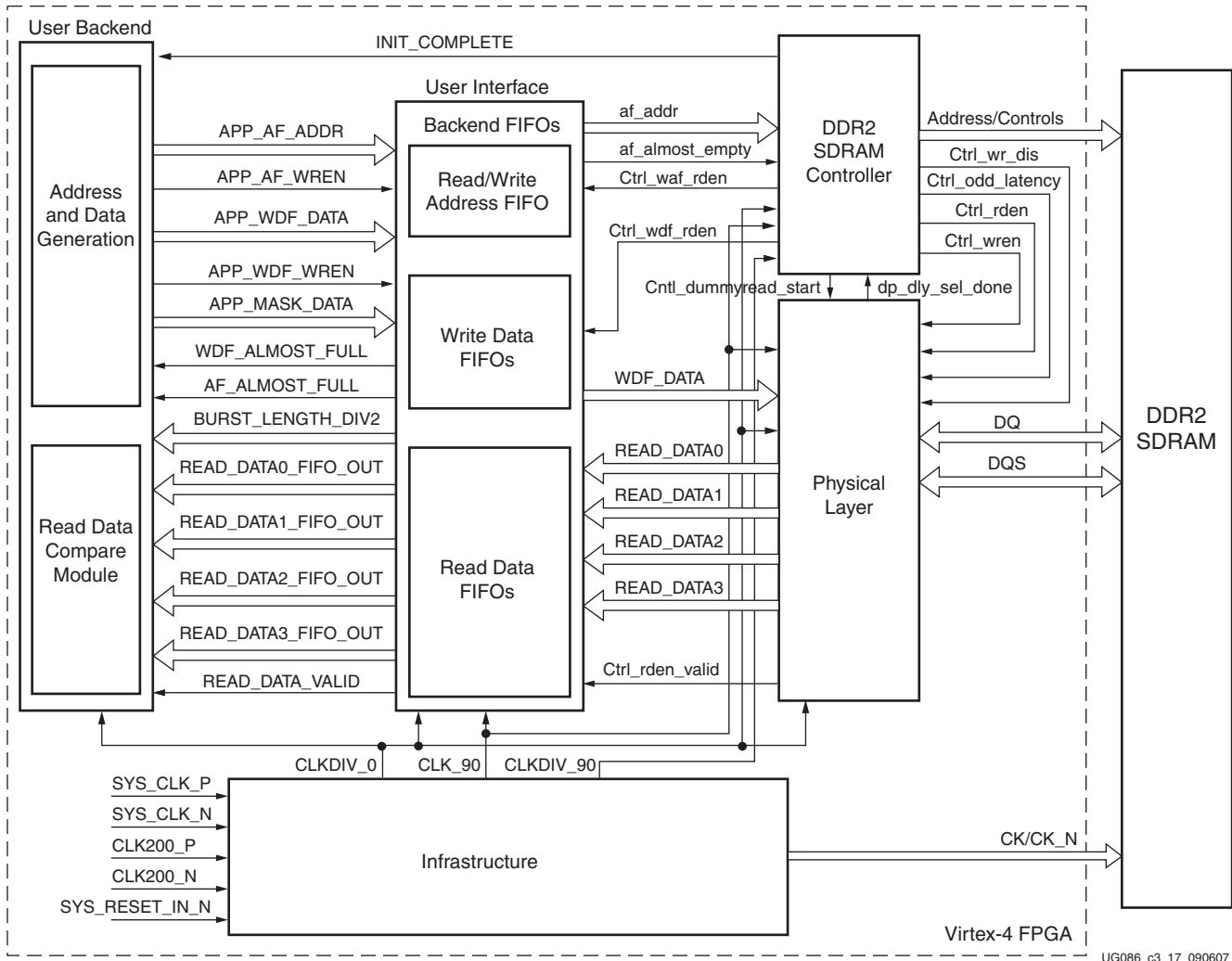


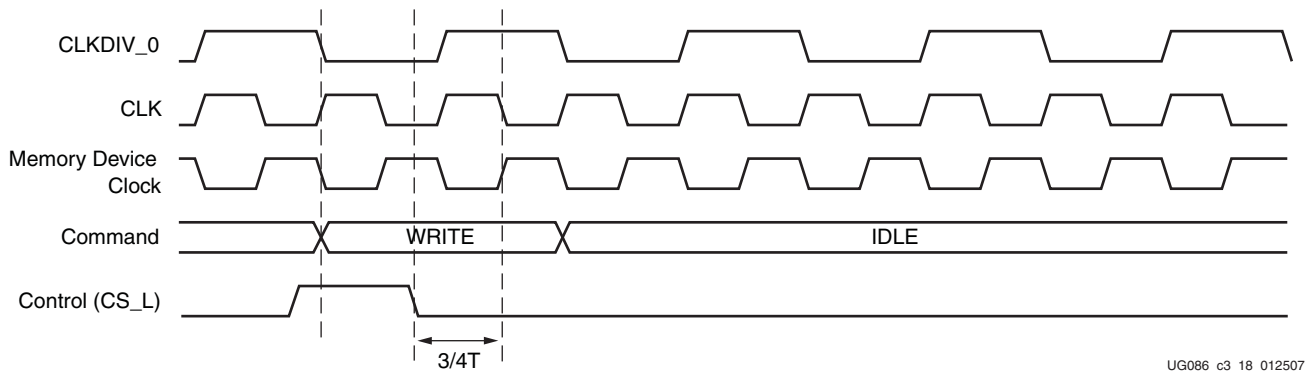
Figure 3-19: DDR2 Memory Controller Block Diagram (SerDes Clocking)

### 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. During these dummy reads, the `tap_logic` module calibrates DQ and DQS by varying the delay to center-align the data with the FPGA clock. Then the `tap_logic` module asserts the `dp_dqs_dq_calib_done` signal. After this assertion, the controller does one more write and read to the memory for read-enable calibration to determine the delay between

the read command and data. Then `dp_dly_slct_done` is asserted to start 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` and `CLK_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 3-20 shows the command and control timing diagram for unbuffered DIMMs and components in which `CS_L` is deasserted  $3/4T$  earlier when the write command is at the positive edge of the device clock to the memory. For registered DIMMs, `CS_L` is deasserted  $T/2$  earlier only.



UG086\_c3\_18\_012507

Figure 3-20: Command and Control Timing from Controller to DDR2 Memory

### Physical Layer

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`, 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_dqs_dq_calib_done` is asserted to start read-enable calibration to find the delay between the read command and data at the input of the Read Data FIFO. So the read enable generated from the controller with the read command is delayed by the same amount and is used as the write enable to the Read Data FIFO for normal reads. Once this read-enable calibration is complete, `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 data stored by the `wr_data_fifo` module is four times the interface 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 one phase-matched clock driver (PMCD) as shown in [Figure 3-21](#).

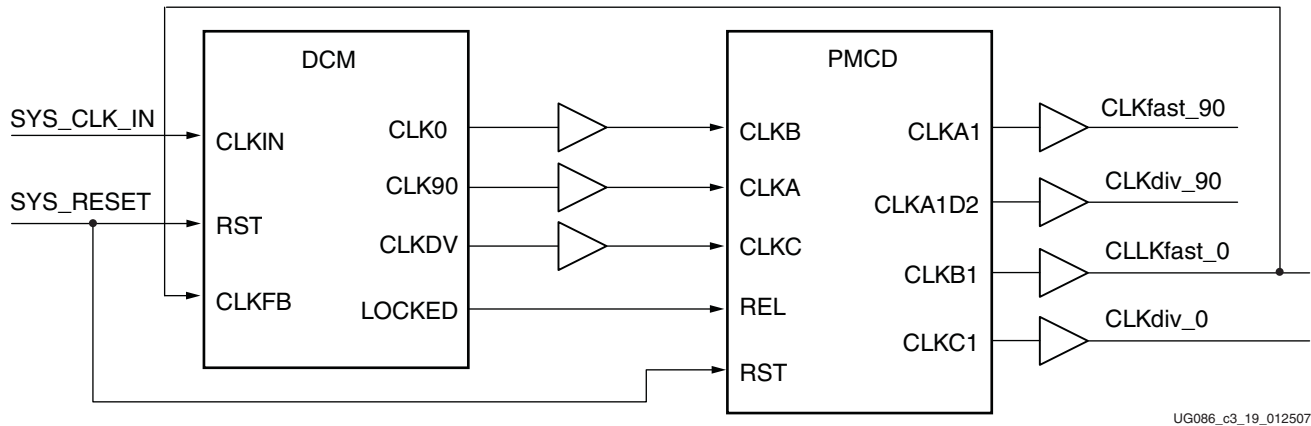


Figure 3-21: Clocking Scheme for the High-Performance Memory Interface Design

**Note:** SerDes design is not supported for FPGAs that do not have PMCDs. Unsupported FPGAs for SerDes design are:

XC4VLX15-FF668	XC4VFX12-FF668	XC4VSX25-FF668
XC4VLX15-FF676	XC4VFX12-SF363	XC4VSX25-FF676
XC4VLX15-SF363	XC4VFX20-FF672	

## DDR2 SDRAM Initialization and Calibration

DDR2 memory is initialized through a specified sequence as per both Micron and JEDEC specifications. The controller starts the memory initialization at power-up. Following the initialization, the relationship between the data and the FPGA clock is calculated using the `tap_logic`. The controller issues a dummy write command and dummy read command to the memory and compares read data with the fixed pattern. During dummy reads, the `tap_logic` module calibrates and delays the DQ and DQS to center-align with the FPGA clock. The `dqs_dq_calib_done` port in the `tap_logic` module indicates the completion of DQS to FPGA clock calibration and per bit calibration.

After the per-bit calibration is done, the controller does a read enable calibration. This calibration is used to determine the delay from read command to read data at `rd_data_fifo`. The delay between read command and read data is affected by the CAS latency and additive latency parameters, the PCB traces, and the I/O buffer delays. This in turn is used to generate a write enable to `rd_data_fifo` so that valid data is registered. The controller issues a dummy read command and compares the read data with a fixed known pattern. The `training_done` port in the `tap_logic` module indicates the completion of the read enable calibration.

The `init_complete` port indicates the completion of DQS to FPGA clock calibration, per-bit calibration, and read enable calibration. After initialization and calibration are done, the controller can start issuing user commands to the memory.

## DDR2 SDRAM System and User Interface Signals

Table 3-21 lists the system signals that are required for the design. 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 comprise the differential clock pair provided to the design. Similarly, CLK200\_P and CLK\_200N comprise the 200 MHz differential clock pair for the IDELAYCTRL module. SYS\_RESET\_IN\_N resets all the logic.

Table 3-21: DDR2 SDRAM System Signals

Signal Name	Direction	Description
SYS_CLK_P, SYS_CLK_N	Input	This differential clock pair generates 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_IN_N	Input	Active-Low reset to the design.

Table 3-22 describes the DDR2 SDRAM user interface signals.

Table 3-22: 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_DIV2[2:0]	Output	This signal determines the data burst length for each write address. 010: burst length = 4 100: burst length = 8
WDF_ALMOST_FULL	Output	This signal indicates the ALMOST_FULL status of the Write Data FIFO. When this signal is asserted, the user can write 5 more data words into the FIFO for the with testbench case and 14 more data words for the without testbench case.
APP_WDF_DATA[4n-1:0]	Input	User write data to the memory, where $n$ indicates the data width of the interface. The user data width is four times the data width of the interface. 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 interface. The mask data width is four times the mask width of the interface. 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. These signals are not present when the memory part does not have mask support (for example, certain Registered DIMMs) or when the Data Mask option is not selected in the MIG GUI.
APP_WDF_WREN	Input	Write Enable signal to the Write Data FIFO.

Table 3-22: DDR2 SDRAM Controller User Interface Signals (Continued)

Signal Name	Direction	Description
AF_ALMOST_FULL	Output	This signal indicates the ALMOST_FULL status of the Address FIFO. When this signal is asserted, the user can issue eight more commands/addresses to the 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. 001: Auto Refresh 010: Precharge All 100: Write 101: Read
APP_AF_WREN	Input	Write Enable signal to the Address FIFO.
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 interface. 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.
INIT_COMPLETE	Output	This signal indicates the completion of initialization to the memory and calibration in the design.

**Notes:**

1. All user interface signal names are prepended with a controller number for the without testbench case, because SerDes clocking supports only a single controller.

## User Interface Accesses

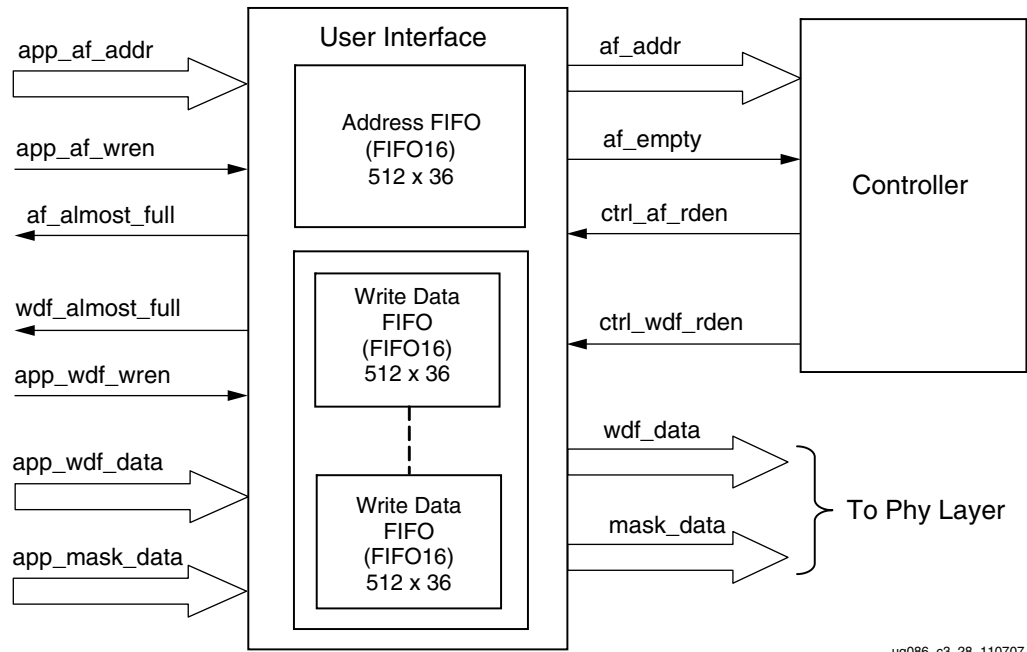
The user backend logic communicates with the memory controller through a FIFO-based user interface. This interface consists of three related buses:

- A Command/Address FIFO bus, which accepts write/read commands as well as the corresponding memory address from the user
- A Write Data FIFO bus, which accepts the corresponding write data when the user issues a write command on the Command/Address bus
- A Read bus on which the corresponding read data for an issued read command is returned

The user interface has the following timing and signaling restriction: When issuing a write command, the first write data word must be written to the Write Data FIFO no more than two clock cycles after the write command is issued. This restriction arises from the fact that the controller assumes write data is available when it receives the write command from the user.

## Write Interface

Figure 3-22 shows the user interface block diagram for write operations.



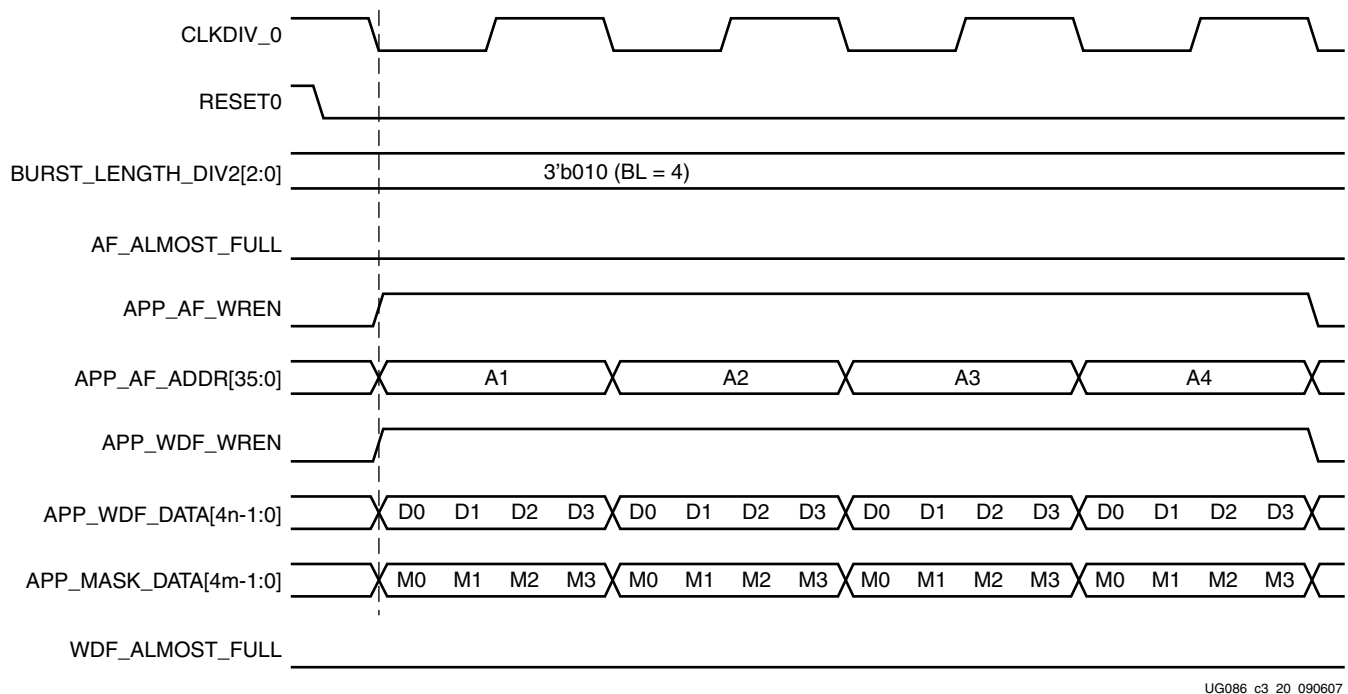
ug086\_c3\_28\_110707

Figure 3-22: User Interface Block Diagram for Write Operations

The following steps describe the architecture of the Address and Write Data FIFOs and show how to perform a write burst operation to DDR2 SDRAM from the user interface.

1. The user interface consists of an Address FIFO and a Write Data FIFO. These FIFOs are constructed using Virtex-4 FIFO16 primitives with a 512 x 36 configuration. The 36-bit architecture comprises one 32-bit port and one 4-bit port. For Write Data FIFOs, the 32-bit port is used for data bits and the 4-bit port is used for mask bits. Mask bits are available only when supported by the memory part *and* when Data Mask is enabled in the MIG GUI. Some memory parts, such as Registered DIMMs of x4 parts, do not support mask bits.
2. The Common Address FIFO is used for both write and read commands, and comprises a command part and an address part. Command bits discriminate between write and read commands.
3. User interface data width `app_wdf_data` is four times that of the memory data width. For an 8-bit memory width, the user interface is 32 bits consisting of two rising-edge data and two falling-edge data. For every 8 bits of data, there is a mask bit. For 72-bit memory data, the user interface data width `app_wdf_data` is 288 bits, and the mask data `app_mask_data` is 36 bits.
4. The minimum configuration of the Write Data FIFO is 512 x 36 for a memory data width of 8 bits.
5. Depending on the memory data width, MIG instantiates multiple FIFO16s to gain the required width. For designs using 8-bit data width, one FIFO16 is instantiated; for 72-bit data width, a total of nine FIFO16s are instantiated. The bit architecture comprises 16 bits of rising-edge data, 2 bits of rising-edge mask, 16 bits of falling-edge data, and 2 bits of falling-edge mask, which are all stored in a FIFO16. MIG routes the `app_wdf_data` and `app_mask_data` to FIFO16s accordingly.

6. The user can initiate a write to memory by writing to the Address FIFO and the Write Data FIFO when the FIFO Full flags are deasserted. Status signal `af_almost_full` is asserted when Address FIFO is full, and similarly `wdf_almost_full` is asserted when Write Data FIFO is full.
7. Both the Address FIFO and Write Data FIFO Full flags are deasserted with power-on.
8. The user should assert the Address FIFO write-enable signal `app_af_wren` along with address `app_af_addr` to store the write address and write command into the Address FIFO.
9. The user should assert the Data FIFO write-enable signal `app_wdf_wren` along with write data `app_wdf_data` and mask data `app_mask_data` to store the write data and mask data into the Write Data FIFO. The user should provide two rising-edge and two falling-edge data together for each write to the Data FIFO.
10. The controller reads the Address FIFO by issuing the `ctrl_af_rden` signal. The controller reads the Write Data FIFO by issuing the `ctrl_wdf_rden` signal after the Address FIFO is read. It decodes the command part after the Address FIFO is read.

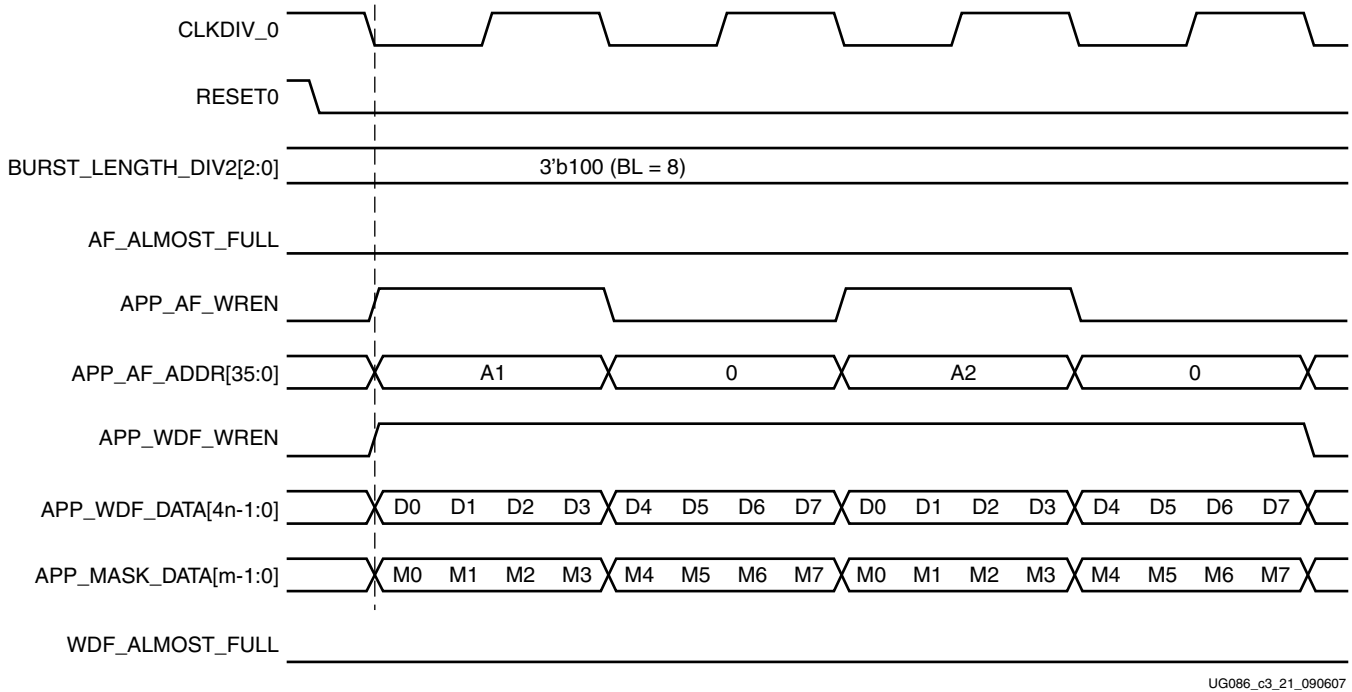


UG086\_c3\_20\_090607

Figure 3-23: DDR2 SDRAM Write Burst (BL = 4) for Four Bursts

11. The write command timing diagram in Figure 3-23 is derived from the MIG-generated test bench. As shown (burst length of 4), each write to the Address FIFO must be coupled with *one* write to the Data FIFO.

**Note:** The user can start filling the Write Data FIFO two clocks after the Address FIFO is written, because there is a two-clock latency between the command fetch and reading the Data FIFO. Using the terms shown in Figure 3-23 and Figure 3-24, therefore, the user can assert the A1 address two clocks before D0D1D2D3. Similarly, A2, A3, and A4 can be advanced by two clocks.



UG086\_c3\_21\_090607

Figure 3-24: DDR2 SDRAM Write Burst (BL = 8) for Two Bursts

12. The write command timing diagram in Figure 3-24 is derived from the MIG-generated test bench. As shown (burst length of 8), each write to the Address FIFO must be coupled with *two* writes to the Data FIFO. Because the controller first reads the address and command together, the address need not coincide with the last data. After the command is analyzed (nearly two clocks later for a worst-case timing scenario), the controller sequentially reads the data in four clocks. Thus, there are six clocks from the time the address is read to the time the last data is read.

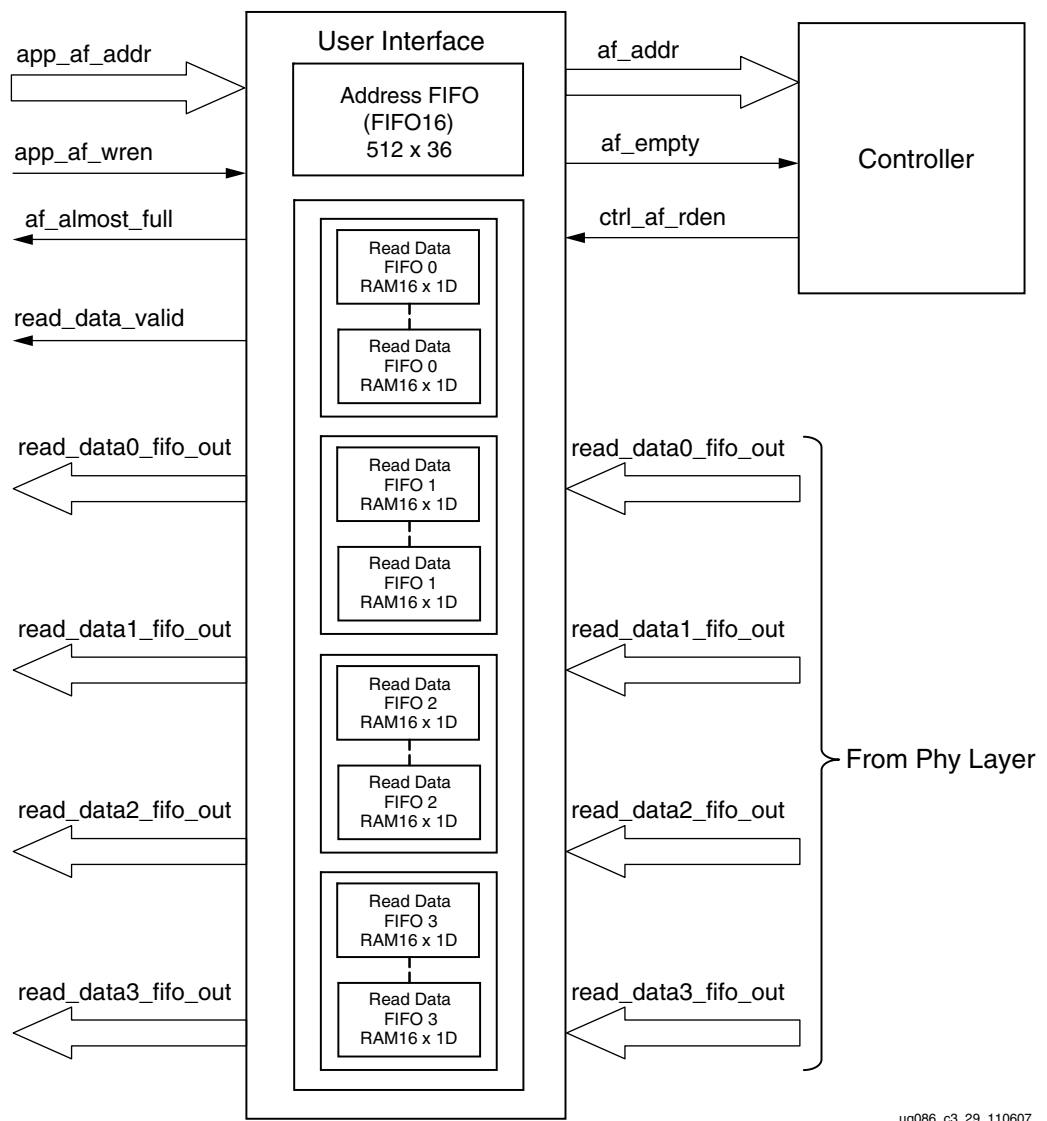
### Correlation between the Address and Data FIFOs

There is a worst case two-cycle latency from the time the address is loaded into the address FIFO on APP\_AF\_ADDR[35:0] to the time the controller decodes the address. Because of this latency, it is not necessary to provide the address on the last clock where data is entered into the data FIFO. If the address is written before the last data phase, the overall efficiency and performance increases because it eliminates or reduces the two-cycle latency. However, if the address is written before data is input into the data FIFO, a FIFO empty condition might result because the Data FIFO does not contain valid data.

Based on these considerations, Xilinx recommends entering the address into the address FIFO between the first data phase and the next-to-last data phase. For a burst of four or eight, this means the Address can be asserted two clocks before the first data phase. This implementation increases efficiency by reducing the two clock latency and guarantees that valid data is available in the Data FIFO.

## Read Interface

Figure 3-25 shows a block diagram of the read interface.



ug086\_c3\_29\_110607

Figure 3-25: User Interface Block Diagram for Read Operation

The following steps describe the architecture of the Read Data FIFOs and show how to perform a burst read operation from DDR SDRAM from the user interface.

1. The read user interface consists of an Address FIFO and a Read Data FIFO. The Address FIFO is common to both read and write operations. These FIFOs are constructed using Virtex-4 Distributed RAMs with a 16 x 1 configuration. MIG instantiates a number of RAM16Ds depending on the data width. For example, for 8-bit data width, MIG instantiates a total of 32 RAM16Ds, 16 for first and second rising-edge data and 16 for first and second falling-edge data. Similarly, for 72-bit data width, MIG instantiates a total of 288 RAM16Ds, 144 for first and second rising-edge data and 144 for first and second falling-edge data.

2. The user can initiate a read to memory by writing to the Address FIFO when the FIFO Full flag `af_almost_full` is deasserted.
3. To write the read address and read command into the Address FIFO, the user should issue the Address FIFO write-enable signal `app_af_wren` along with read address `app_af_addr`.
4. The controller reads the Address FIFO containing the address and command. After decoding the command, the controller generates the appropriate control signals to memory.
5. Prior to the actual read and write commands, the design calibrates the latency (number of clock cycles) from the time the read command is issued to the time data is received. Using this pre-calibrated delay information, the controller generates the write-enable signals to the Read Data FIFOs.

After the power-up calibration is done, dummy reads are executed to set up the delay between the read command and read data from the memory. During the time 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/CK) signals should be matched in trace length from the FPGA to the memory device. MIG ensures that a DQS and its corresponding DQ signals do not cross a bank boundary.

6. The `read_data_valid` signal is asserted when data is available in the Read Data FIFOs.

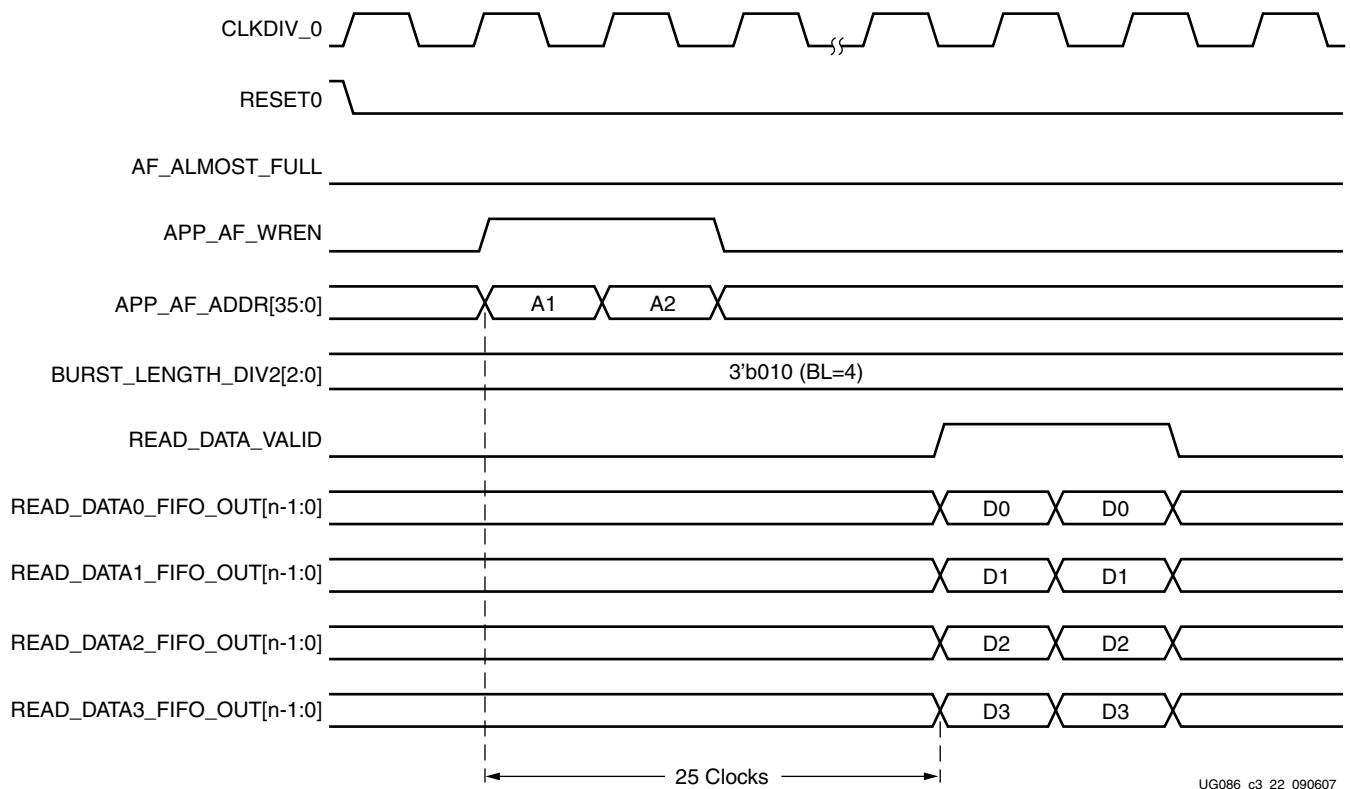
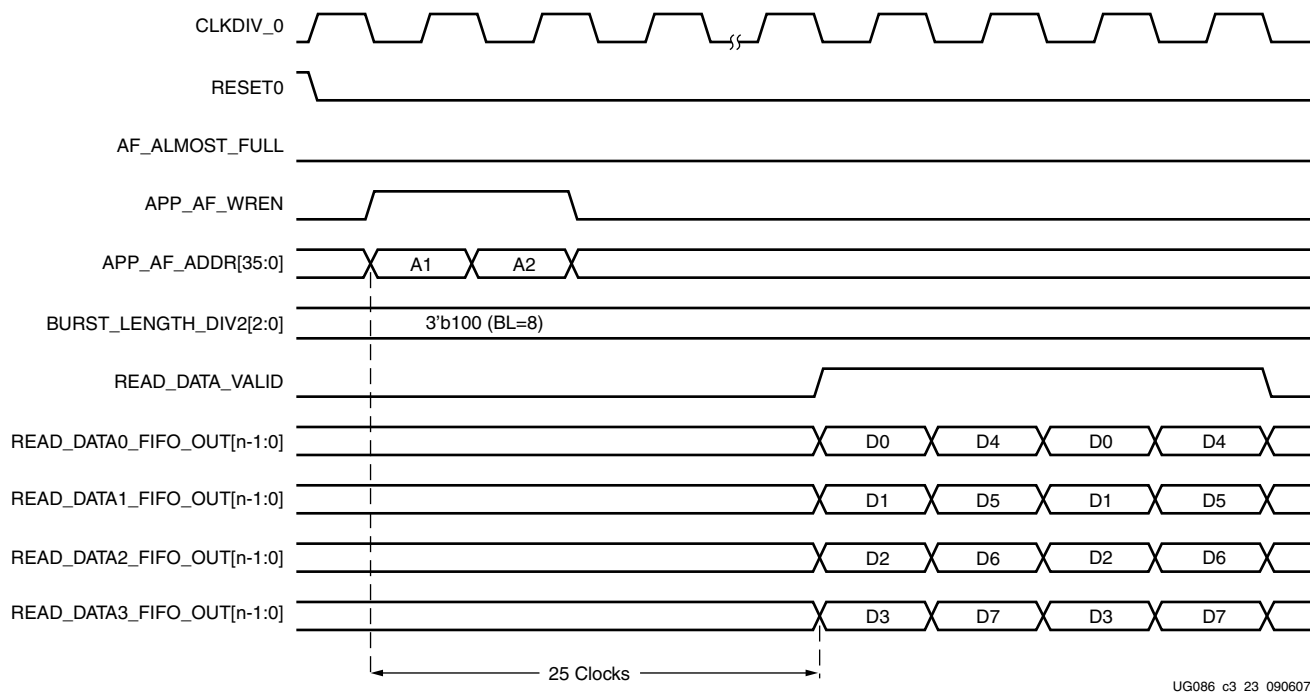


Figure 3-26: DDR2 SDRAM Read Burst (BL = 4) for Two Bursts





**Figure 3-27: DDR2 SDRAM Read Burst (BL = 8) for Two Bursts**

7. [Figure 3-26](#) shows the user interface timing diagram for a burst length of 4, and [Figure 3-27](#) shows user interface timing diagram for a burst length of 8. Both the cases shown here are for a CAS latency of 4 at 200 MHz. The read latency is calculated from the point when the read command is given by the user to the point when the data is available with the read\_data\_valid signal. The minimum latency in this case is 25 clocks, where no precharge is required, no auto-refresh request is pending, the user commands are issued after initialization is completed, and the first command issued is a Read command. Controller executes the commands only after initialization is done as indicated by the init\_done signal.
8. After the address and command are loaded into the Address FIFO, it takes 25 clock cycles minimum for the controller to assert the read\_data\_valid signal.
9. Read data is available only when the read\_data\_valid signal is asserted. The user should access the read data on every positive edge of the read\_data\_valid signal.

[Table 3-23](#) shows how the 25 clocks from the read command to the read data are broken up.

**Table 3-23: Read Command to Read Data Clock Cycles**

Parameter	Number of Clocks (CLKDIV_0)
Read Command to Empty Signal Deassertion	7 Clocks
Empty to Active Command	5.5 Clocks
Active to Read Command	3 Clocks
Memory Read Command to Read Data Valid	9.5 Clocks
<b>Total:</b>	<b>25 Clocks</b>

In general, read latency varies based on the following parameters:

- CAS latency (CL) and additive latency (AL)
- The number of commands already in the FIFO pipeline before the read command is issued
- Whether an ACTIVATE command needs to be issued to open the new bank/row
- Whether a PRECHARGE command needs to be issued to close a previously opened bank
- Specific timing parameters for the memory, such as TRAS, and TRCD in conjunction with the bus clock frequency
- Possible interruption of commands and/or forced closure of banks/rows when the periodic AUTO REFRESH command is issued
- Commands issued by the user before initialization is complete, causing latency to be indeterminate
- Board-level and chip-level (for both memory and FPGA) propagation delays

### User to Controller Interface

Table 3-24 lists the signals between the user interface and the controller.

Table 3-24: 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] Dynamic Command Request: [34:32] Reserved: [35]	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 3-28 shows the timing waveform for a burst length of 8 with two back-to-back writes followed by two back-to-back reads.

Table 3-24: List of Signals Between User Interface and Controller (Continued)

Port Name	Port Width	Port Description	Notes
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 3-28 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]
```

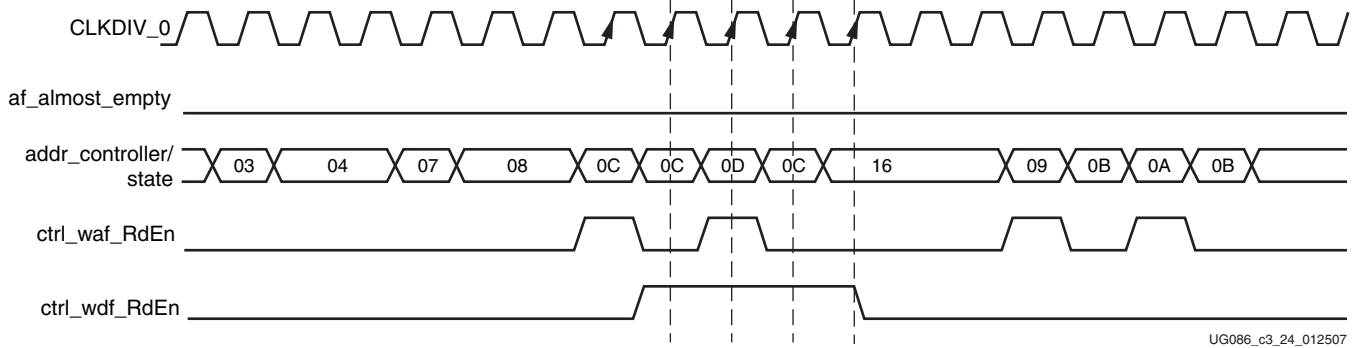
## Dynamic Command Request

Table 3-25 lists the commands supported from user interface.

Table 3-25: User Interface Commands

Command	Description
001	Auto Refresh
010	Precharge All
100	Write
101	Read

Figure 3-28 describes two consecutive writes followed by two consecutive reads with a burst length of 8. Table 3-26 lists the state signal values for Figure 3-28.



UG086\_c3\_24\_012507

Figure 3-28: Controller Read of Command and Data from User Interface FIFOs for a Burst Length of 8

Table 3-26: State Signal Values for Figure 3-28

State Signal Value (hex)	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 3-27 lists the signals between the controller and the physical layer.

Table 3-27: 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).	
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 3-29 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.

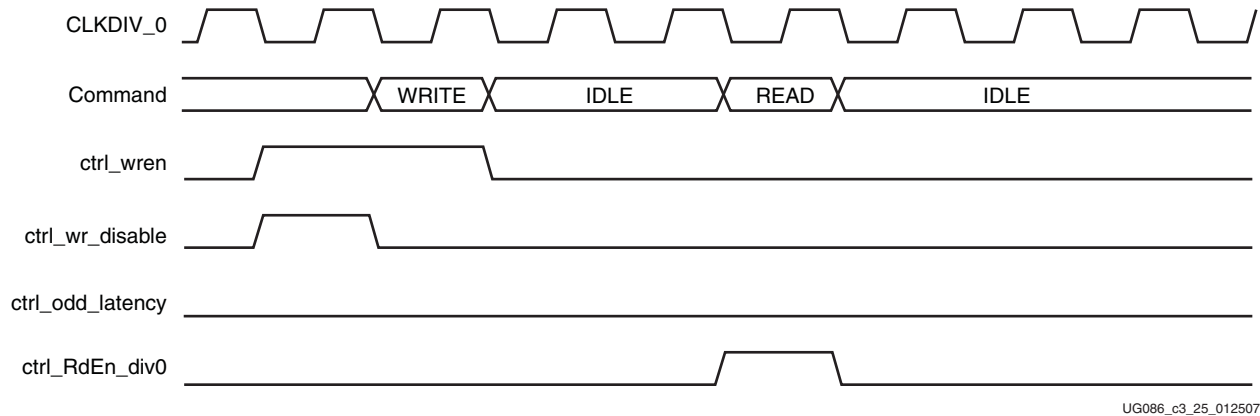


Figure 3-29: Timing Waveform for Control Signals from the Controller to the Physical Layer

MIG allows bank selection for different classes of memory signals. When a particular bank is checked for address, MIG allocates the memory address, the memory control, and the memory clocks in that bank. When a bank is checked for data, MIG allocates the data, the data mask, and the data strobes in that bank. When a bank is checked for system control, MIG allocates the system reset and status signals in that bank. When a bank is checked for system clocks, MIG allocates the system clock signals in that bank.

Table 3-28 shows the list of signals allocated in a group from bank selection check boxes.

Table 3-28: SerDes DDR2 SDRAM Signal Allocations

Bank Selected	Signals Allocated in the Group
Address	Memory address, memory control, and memory clock signals
Data	Data, data mask, and data strobes
System Control	System reset from user interface and status signals
System_Clock	System clocks from user interface

## Simulating the DDR2 SDRAM Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains the external testbench, memory model, `.do` file, and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more details about the files in the `sim` folder and to simulate the design, see `simulation_help.chm` in the `sim` folder.

## Changing the Refresh Rate

The formula is similar to the Virtex-4 DDR2 Direct Clocking case. However, since the refresh logic in the controller is running at half the memory bus rate, the formula is  $MAX\_REF\_CNT = (\text{refresh interval}) / (2 * \text{clock period})$ . For example, for a refresh rate of 3.9  $\mu\text{s}$  with a memory bus running at 267 MHz:

$$MAX\_REF\_CNT = 3.9 \mu\text{s} / (2 * \text{clock period}) = 3.9 \mu\text{s} / 7.49 \text{ ns} = 521 \text{ (decimal)} = 0 \times 209$$

If the above value exceeds  $2^{\text{MAX\_REF\_WIDTH}} - 1$ , the value of MAX\_REF\_WIDTH must be increased accordingly in `parameters_0.v` (or `.vhd`) to increase the width of the counter used to track the refresh interval.

## Supported Devices

The design generated out of MIG is independent of memory package, hence the package part of the memory component is replaced with XX, where XX indicates a don't care condition. The tables below list the components (Table 3-29) and DIMMs (Table 3-30 through Table 3-32) supported by the tool for DDR2 SerDes clocking designs.

In supported devices, an X in the component column denotes a single alphanumeric character. For example MT47H128M4XX-3 can be either MT47H128M4BP-3 or MT47H128M4B6-3. An XX for Registered DIMMs denotes a single or two alphanumeric characters. For example, MT9HTF3272XX-667 can be either MT9HTF3272Y-667 or MT9HTF3272DY-667.

**Table 3-29: Supported Components for DDR2 SDRAM**

Components	Packages (XX)	Components	Packages (XX)
MT47H64M4XX-3	BP	MT47H128M8XX-3	BT,HQ
MT47H64M4XX-37E	BP	MT47H128M8XX-37E	BT,HQ
MT47H64M4XX-5E	BP	MT47H128M8XX-5E	BT,HQ
MT47H128M4XX-3	B6,CB,GB	MT47H256M8XX-3	HG
MT47H128M4XX-37E	B6,CB,GB	MT47H256M8XX-37E	HG
MT47H128M4XX-5E	B6,CB,GB	MT47H256M8XX-5E	HG
MT47H256M4XX-3	BT,HQ	MT47H16M16XX-3	BG
MT47H256M4XX-37E	BT,HQ	MT47H16M16XX-37E	BG
MT47H256M4XX-5E	BT,HQ	MT47H16M16XX-5E	BG
MT47H512M4XX-3	HG	MT47H32M16XX-3	BN,CC,FN,GC
MT47H512M4XX-37E	HG	MT47H32M16XX-37E	BN,CC,FN,GC
MT47H512M4XX-5E	HG	MT47H32M16XX-5E	BN,CC,FN,GC
MT47H32M8XX-3	BP	MT47H64M16XX-3	BT,HR
MT47H32M8XX-37E	BP	MT47H64M16XX-37E	BT,HR
MT47H32M8XX-5E	BP	MT47H64M16XX-5E	BT,HR
MT47H64M8XX-3	B6,CB,F6,GB	MT47H128M16XX-3	HG
MT47H64M8XX-37E	B6,CB,F6,GB	MT47H128M16XX-37E	HG
MT47H64M8XX-5E	B6,CB,F6,GB	MT47H128M16XX-5E	--

**Table 3-30: Supported Registered DIMMs for DDR2 SDRAM**

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9HTF3272XX-667	--	MT18HTF25672XX-667	PDY,PY,Y
MT9HTF3272XX-53E	Y	MT18HTF25672XX-53E	PDY,PY,Y
MT9HTF3272XX-40E	Y	MT18HTF25672XX-40E	DY,PDY,Y
MT9HTF6472XX-667	PY,Y	MT18HTF6472XXX-667	--
MT9HTF6472XX-53E	Y	MT18HTF6472XXX-53E	DY,Y

**Table 3-30: Supported Registered DIMMs for DDR2 SDRAM (Continued)**

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9HTF6472XX-40E	Y	MT18HTF6472XXX-40E	DY,Y
MT9HTF12872XX-667	PY	MT18HTF12872XXX-667	DY,PDY,PY,Y
MT9HTF12872XX-53E	PY,Y	MT18HTF12872XXX-53E	DY,MY,NDY, NY,PY,Y
MT9HTF12872XX-40E	Y	MT18HTF12872XXX-40E	DY,PY,Y
MT18HTF6472G-53E	--	MT18HTF25672XXX-667	PDY,PY,Y
MT18HTF6472XX-667	--	MT18HTF25672XXX-53E	PDY,PY,Y
MT18HTF6472XX-53E	DY,Y	MT18HTF25672XXX-40E	DY,PDY,Y
MT18HTF6472XX-40E	DY,Y	MT36HTJ51272XX-667	--
MT18HTF12872XX-667	DY,PDY,PY,Y	MT36HTJ51272XX-53E	Y
MT18HTF12872XX-53E	DY,MY,NDY, NY,PY,Y	MT36HTJ51272XX-40E	Y
MT18HTF12872XX-40E	DY,PY,Y	--	--

**Table 3-31: Supported Unbuffered DIMMs for DDR2 SDRAM**

Unbuffered DIMMs	Unbuffered DIMMs
MT4HTF1664AY-667	MT8HTF12864AY-667
MT4HTF1664AY-40E	MT8HTF12864AY-40E
MT4HTF3264AY-667	MT9HTF3272AY-667
MT4HTF3264AY-40E	MT9HTF3272AY-40E
MT4HTF6464AY-667	MT9HTF6472AY-667
MT4HTF6464AY-40E	MT16HTF25664AX-40E
MT8HTF6464AY-667	MT18HTF6472AY-40E
MT8HTF6464AY-53E	MT18HTF12872AY-40E
MT8HTF6464AY-40E	MT18HTF25672AY-40E

**Table 3-32: Supported SODIMMs for DDR2 SDRAM**

SODIMMs	SODIMMs
MT4HTF1664HY-667	MT8HTF3264HY-53E
MT4HTF1664HY-53E	MT8HTF3264HY-40E
MT4HTF1664HY-40E	MT8HTF6464HY-667
MT4HTF3264HY-667	MT8HTF6464HY-53E
MT4HTF3264HY-53E	MT8HTF6464HY-40E
MT4HTF3264HY-40E	MT8HTF3264HDY-40E
MT8HTF3264HY-667	MT8HTF6464HDY-40E



## Hardware Tested Configurations

The frequencies shown in [Table 3-33](#) were achieved on the Virtex-4 FPGA ML461 Memory Interfaces Development Board under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based on a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 64-bit wide interface.

**Table 3-33: Hardware Tested Configurations**

Synthesis Tools	XST and Synplicity
HDL	Verilog and VHDL
FPGA Device	XC4VLX25-FF668-11
Burst Lengths	4, 8
CAS Latency	4, 5
Additive Latency	0, 1, 2
8-bit Design	Tested on 16-bit Component "MT47H32M16XX-3"
64-bit Design	Tested on 64-bit DIMM "MT8HTF6464AY-667"
72-bit Design	Tested on 72-bit DIMM "MT9HTF6472XX-667"
Frequency Range	140 MHz to 400 MHz for component and Registered DIMMs
	140 MHz to 290 MHz for Unbuffered DIMMs



## Implementing QDRII SRAM Controllers

This chapter describes how to implement QDRII SRAM interfaces for Virtex™-4 FPGAs generated with MIG. This design is based on XAPP703 [Ref 19].

### Feature Summary

The QDRII controller design supports the following:

- 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
- With and without a testbench
- With and without a DCM

### Design Frequency Range

Table 4-1: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-10		-11		-12	
	Min	Max	Min	Max	Min	Max
Component	120	200	120	240	120	250

### Limitations

The controller performs consecutive read and writes when the User Read Address FIFO, the User Write Address FIFO, and the User Read Data FIFOs are not full. The controller might not follow the user issued commands sequence. When the User Read Address FIFO is empty, the controller performs writes with the memory, depending on the status of the User Write Data FIFOs and User Write Address FIFO. When the either the User Write Data FIFOs or the User Write Address FIFO is empty, the controller performs reads with the memory, depending on the status of the User Read Address FIFO. The controller remains in the IDLE state when the User Read Address FIFO, the User Write Address FIFO, and the User Write Data FIFOs are empty.

## Architecture

Figure 4-1 shows a 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.

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.

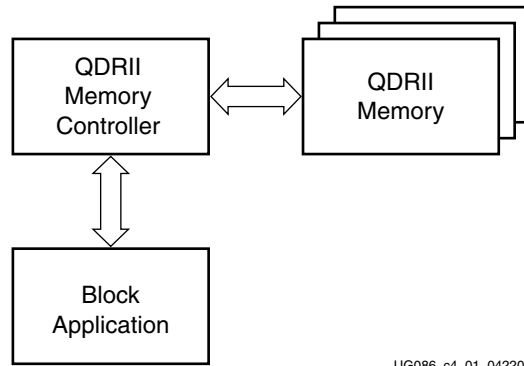


Figure 4-1: QDRII Memory Controller

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.

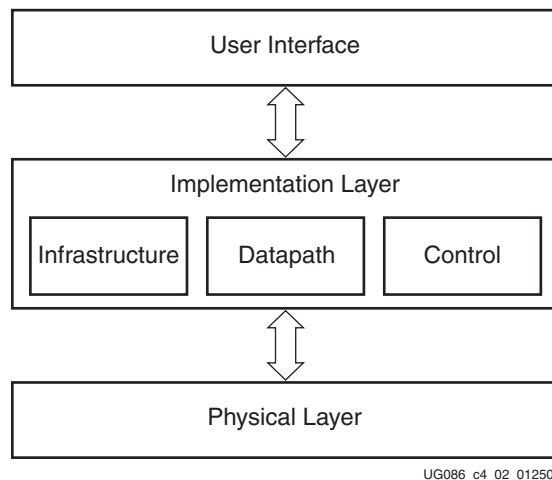


Figure 4-2: Interface Layering Model

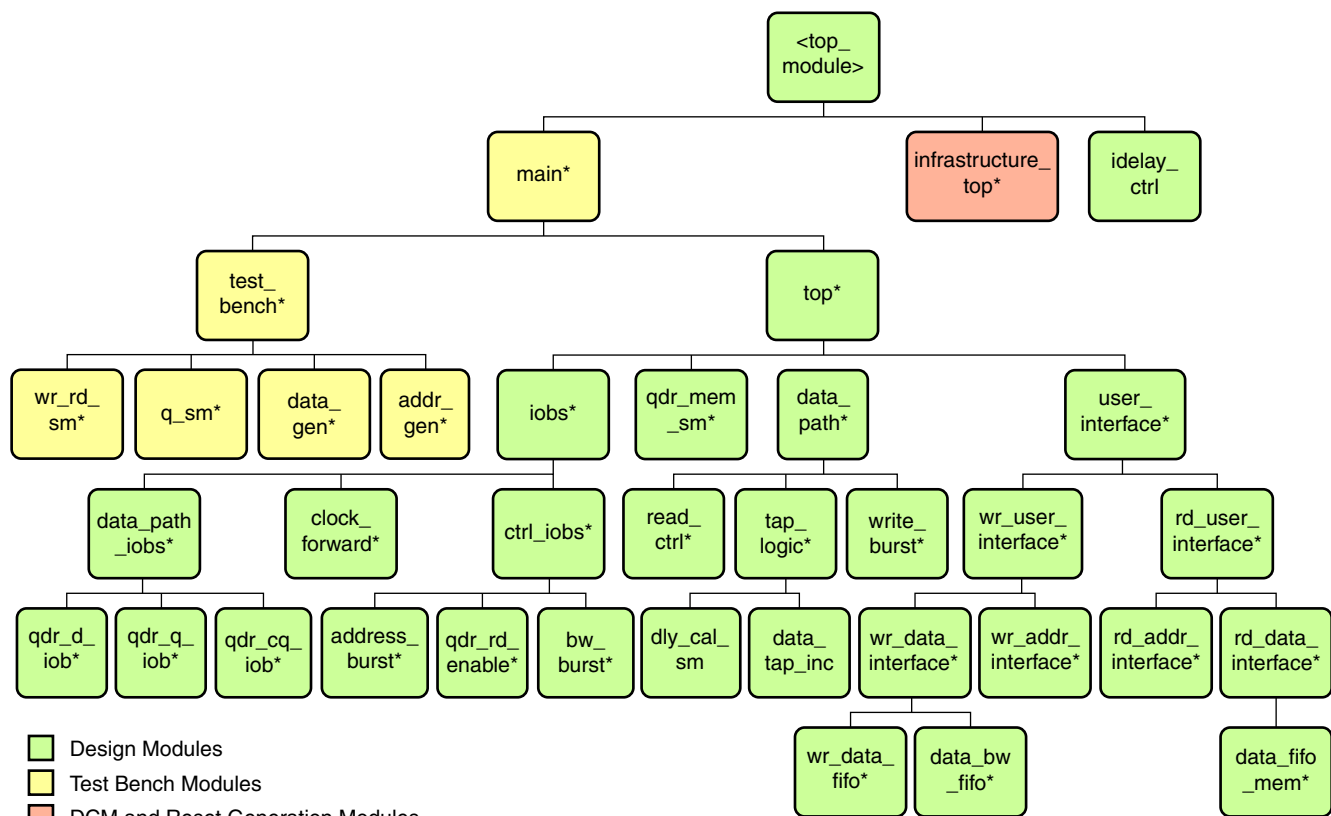
The application layer comprises the user interface, which initiates memory writes and reads by writing data and memory addresses to the User Interface FIFOs. The implementation layer comprises the infrastructure, datapath, and control logic.

- The infrastructure logic consists of the DCM and reset logic generation circuitry.
- The datapath logic consists of the calibration logic by which the data from the memory component is captured using the FPGA clock.
- The control logic determines the type of data transfer, that is, read/write with the memory component, depending on the User Interface FIFO's status signals.

The physical layer comprises the I/O elements of the FPGA. The controller communicates with the memory component using this layer. The I/O elements (such as IDDRs, ODDRs, and IDELAY elements) are associated with this layer.

## Hierarchy

Figure 4-3 shows the QDRII SRAM controller hierarchy.



- Design Modules
- Test Bench Modules
- DCM and Reset Generation Modules

Note: A block with a \* has a parameter file included.

UG086\_c4\_03\_091207

Figure 4-3: QDRII SRAM Controller Hierarchy

Figure 4-3 shows the hierarchical structure of the QDRII SRAM design generated by MIG with a testbench and a DCM. The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks and reset generation modules

There is a parameter file generated with the design that has all the user input and design parameters selected from MIG.

MIG can generate QDRII SRAM designs in four different ways:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

For a design without a testbench (user\_design) generated by MIG, the design top-level module has the user interface signals. The list of user interface signals is provided in [Table 4-4, page 181](#).

Design clocks and resets are generated in the infrastructure\_top module. When the **Use DCM** option is checked in MIG, a DCM primitive and the necessary clock buffers are instantiated in the infrastructure\_top module. The inputs to this module are the differential design clock and a 200 MHz differential clock required for the IDELAYCTRL module. A user reset is also input to this module. Using the input clocks and reset signals, the system clocks and the system resets used in the design are generated in this module.

When the **Use DCM** option is unchecked in MIG, the infrastructure\_top module does not have the DCM and the corresponding clock buffer instantiations; therefore, the system operates on the user-provided clocks. The system reset is generated in the infrastructure\_top module using the DCM\_LOCK signal and the ready signal of the IDELAYCTRL element.

Figure 4-4 shows a top-level block diagram of a QDRII SRAM design with a DCM and a testbench. Inputs to the design are referenced to a differential clock pair (REFCLK\_P and REFCLK\_N) for the controller design, a 200 MHz differential clock pair (DLY\_CLK\_200\_P and DLY\_CLK\_200\_N) for the IDELAYCTRL element, and the system reset signal, SYS\_RST\_N. All design resets are generated using the DCM\_LOCKED signal, the SYS\_RST\_N signal, and the dly\_ready signal of the IDELAYCTRL element. The COMPARE\_ERROR output signal indicates whether the design passes or fails. The DLY\_CAL\_DONE signal indicates the completion of initialization and calibration of the design. Because the DCM is instantiated in the infrastructure module, it generates the required clocks and reset signals for the design.

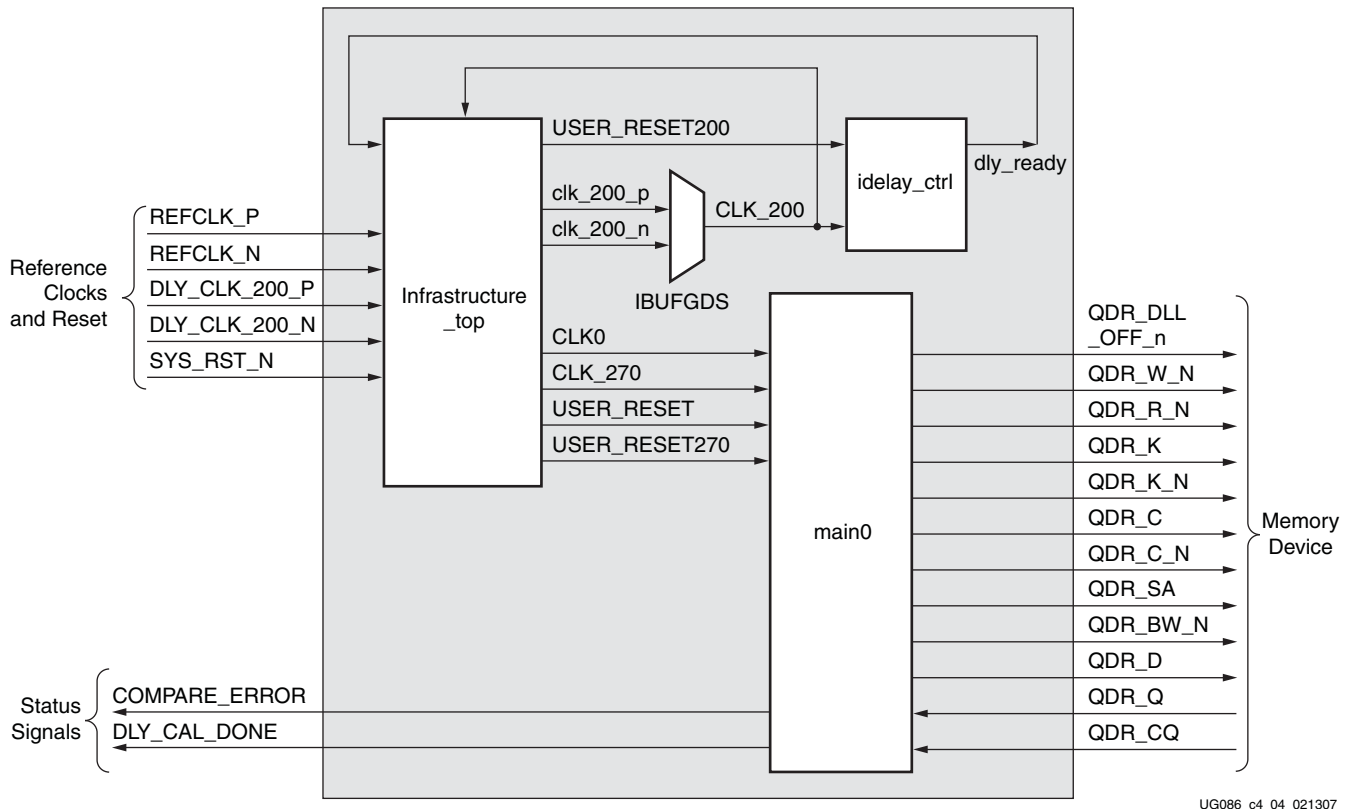
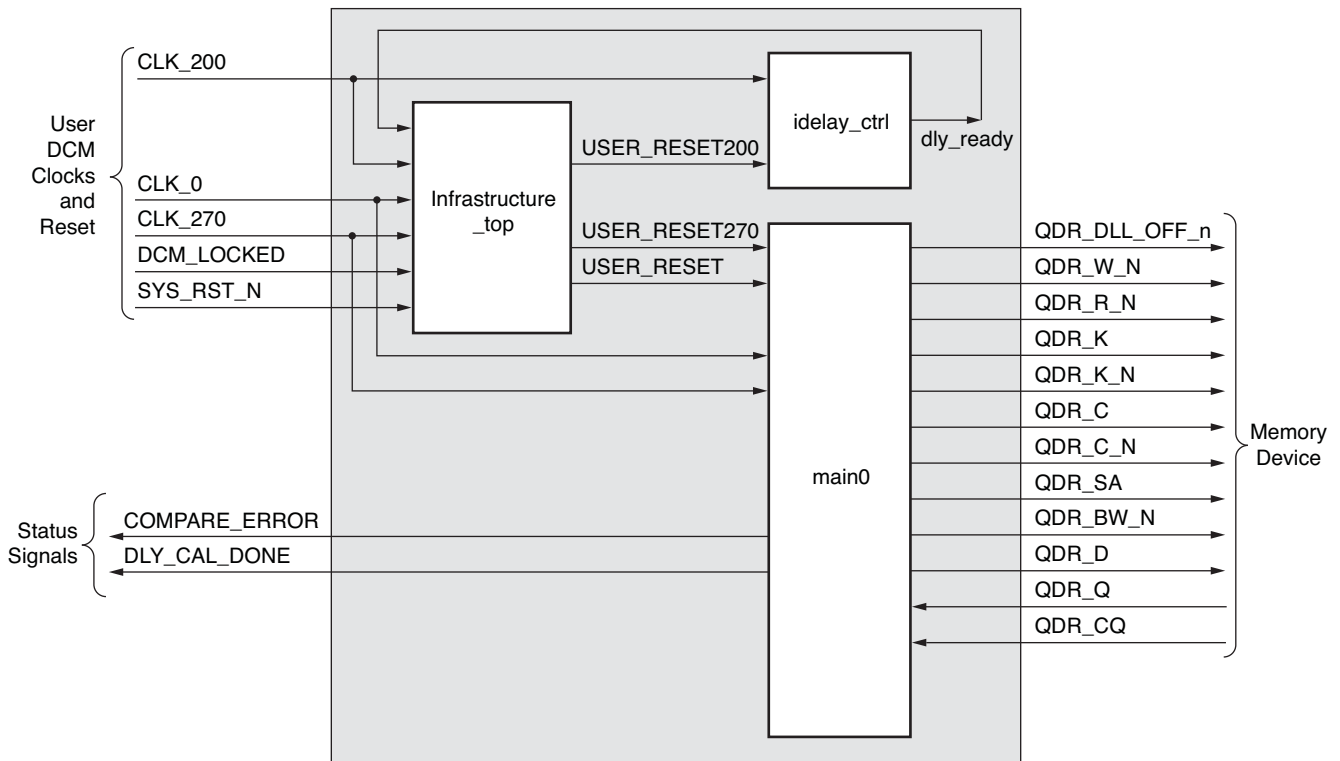


Figure 4-4: Top-Level Block Diagram of the QDRII SRAM Design with a DCM and a Testbench

Figure 4-5 shows a top-level block diagram of a QDRII SRAM design without a DCM but with a testbench. The user should provide all the clocks and the DCM\_LOCKED signal. These clocks should be single-ended. SYS\_RST\_N is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the SYS\_RST\_N signal, and the dly\_ready signal of the IDELAYCTRL element. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFs. The COMPARE\_ERROR signal, which is the output of the design, indicates whether the design passes or fails. The testbench module does writes and reads, and also compares the read data with written data. The COMPARE\_ERROR signal is set High on data mismatches. The DLY\_CAL\_DONE signal indicates the completion of initialization and calibration of the design.

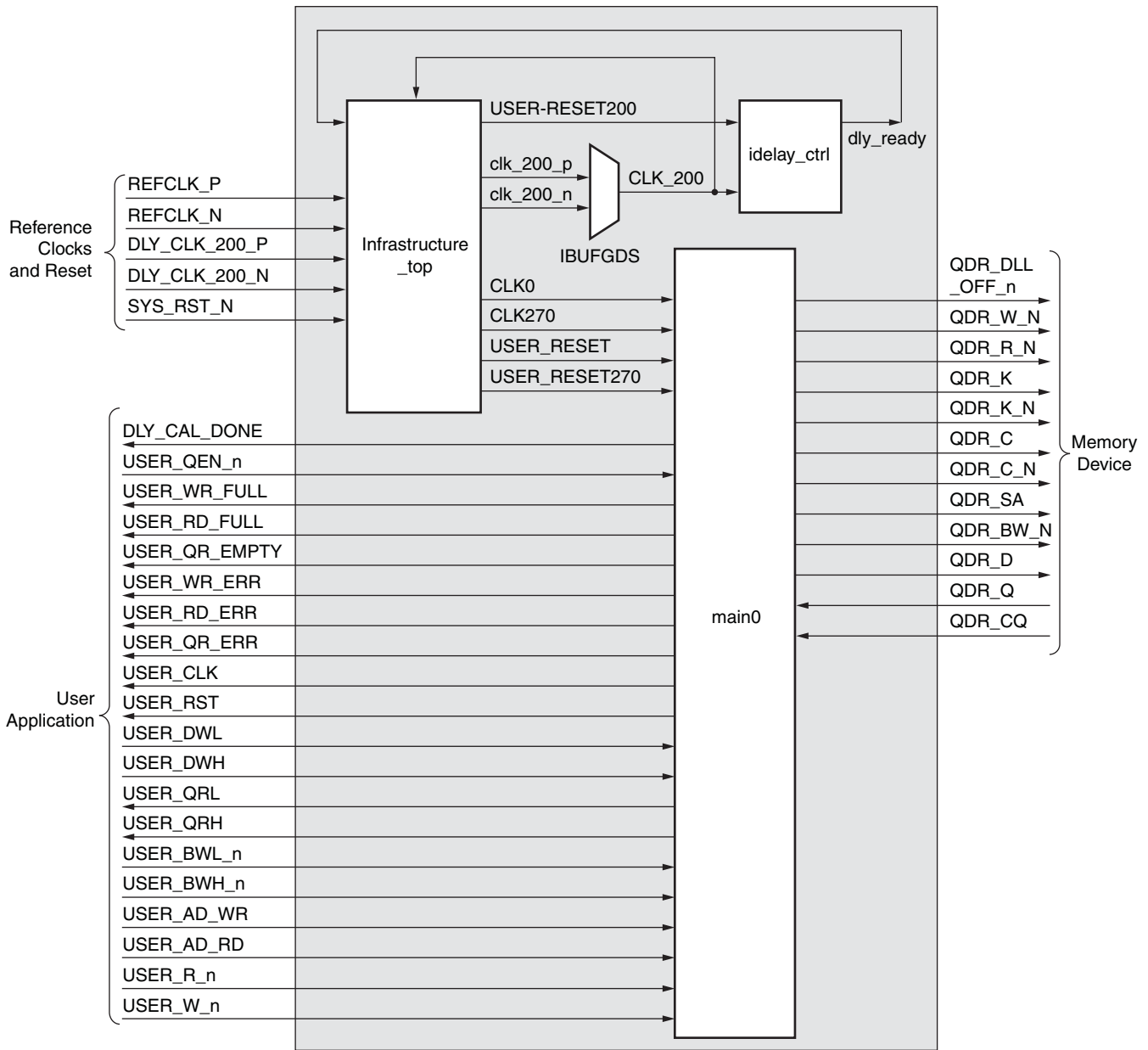


UG086\_c4\_05\_013007

Figure 4-5: Top-Level Block Diagram of the QDRII SRAM Design with a Testbench but without a DCM



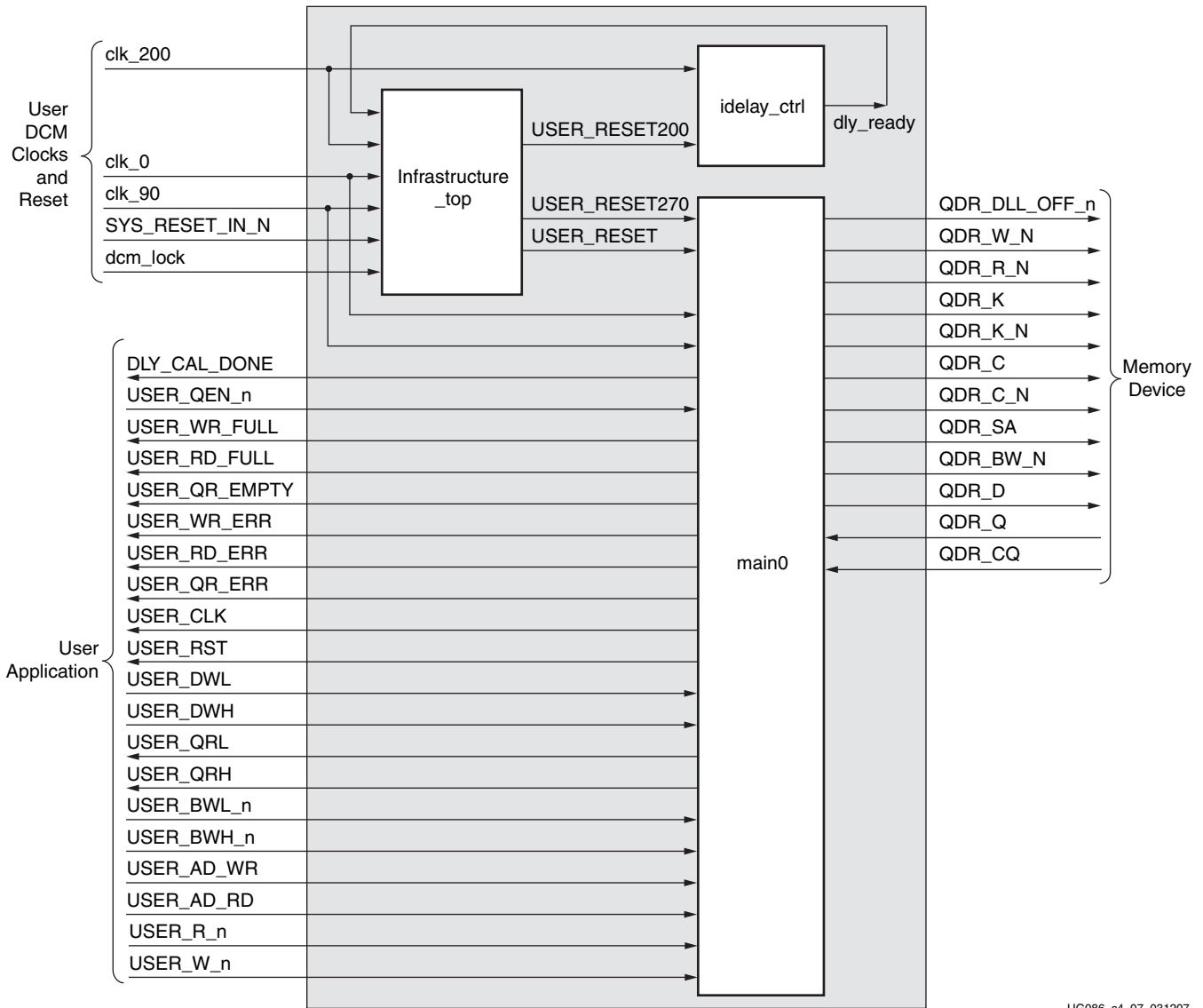
Figure 4-6 shows a top-level block diagram of a QDRII SRAM design with a DCM but without a testbench. REFCLK\_P and REFCLK\_N are differential input reference clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. DLY\_CLK\_200\_P and DLY\_CLK\_200\_N are used for the IDELAYCTRL element. SYS\_RST\_N is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the SYS\_RST\_N signal, and the dly\_ready signal of IDELAYCTRL element. The user has to drive the user application signals. The design provides the USER\_CLK and USER\_RST signals to the user to synchronize the user application signals with the design. The DLY\_CAL\_DONE signal indicates the completion of initialization and calibration of the design.



UG086\_c4\_06\_031207

Figure 4-6: Top-Level Block Diagram of the QDRII SRAM Design with a DCM but without a Testbench

Figure 4-7 shows a top-level block diagram of a QDRII SRAM design without a DCM or a testbench. The user should provide all the clocks and the DCM\_LOCKED signal. These clocks should be single-ended. SYS\_RST\_N is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the SYS\_RST\_N signal, and the dly\_ready signal of the IDELAYCTRL element. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFPGs. The user has to drive the user application signals. The design provides the USER\_CLK and USER\_RST signals to the user to synchronize the user application signals with the design. The DLY\_CAL\_DONE signal indicates the completion of initialization and calibration of the design.

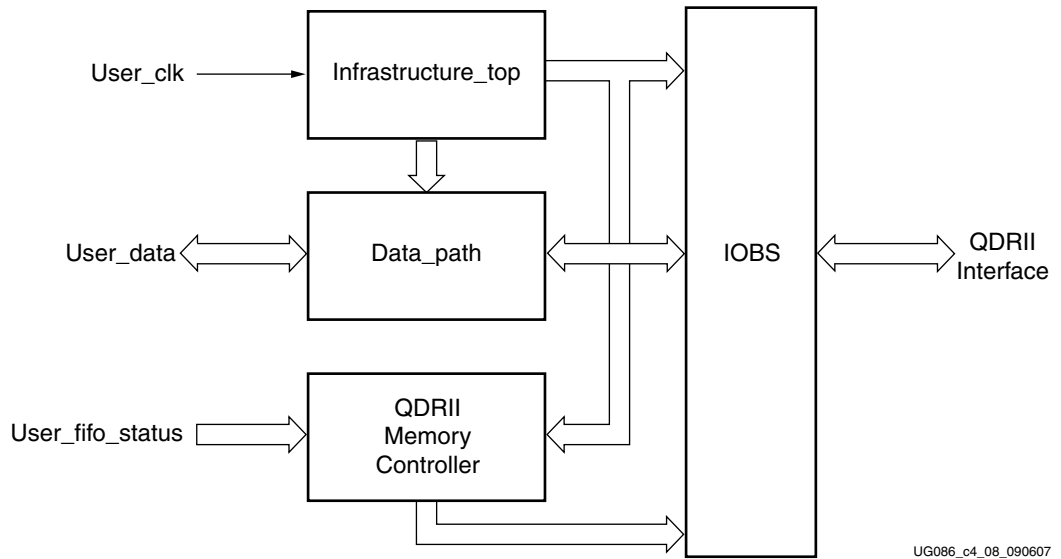


UG086\_c4\_07\_031207

Figure 4-7: Top-Level Block Diagram of the QDRII SRAM Design without a DCM or a Testbench

## QDRII Memory Controller Modules

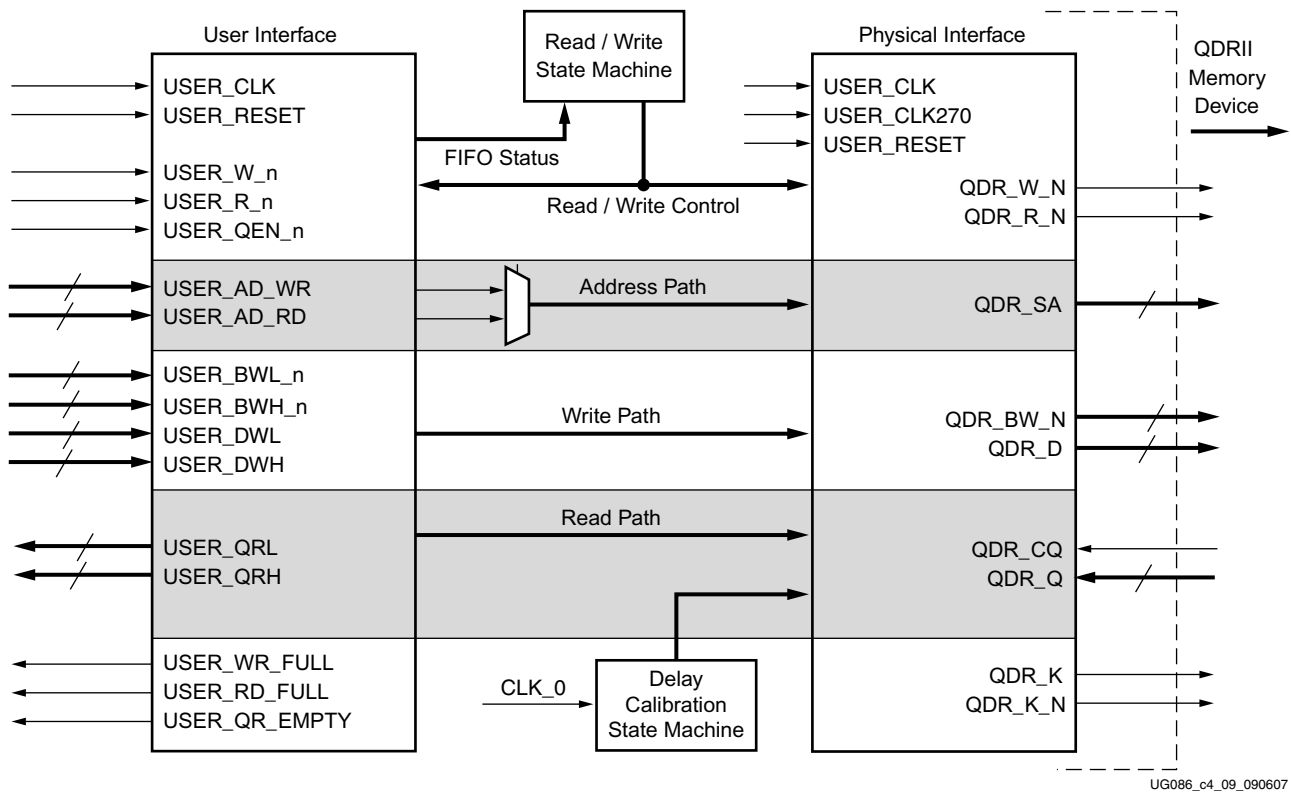
Figure 4-8 shows a detailed block diagram of the QDRII memory controller. The four blocks shown are sub-blocks of the top module. The functionalities of these blocks are explained in the subsections following the figure.



UG086\_c4\_08\_090607

Figure 4-8: QDRII Memory Controller Modules

Figure 4-9 shows the QDRII memory controller modules with a 36-bit interface.



UG086\_c4\_09\_090607

Figure 4-9: QDRII Memory Controller Modules

## Controller

The QDRII memory controller initiates alternate Write and Read commands to the memory as long as the User Write Data FIFOs, the User Write Address FIFO, and the User Read Address FIFO are not empty, and the User Read Data FIFOs are not full.

The user writes the write data and the write address into the User Write Data FIFOs and the User Write Address FIFO, respectively. When neither the User Write Data FIFOs nor the User Write Address FIFO is empty, the QDRII controller generates a write-enable signal to the memory. When the write enable is asserted, the write data and the write address are transferred to memory from the User Write Data FIFOs and the User Write Address FIFO, respectively.

The read address from where the data is to be read from memory is stored by the user in the User Read Address FIFO. The QDRII memory controller generates a read-enable signal to the memory when the User Read Address FIFO is not empty and the User Read Data FIFOs are not full. When the read enable is asserted, the read address from the Read Address FIFO is transferred to memory. The captured read data from the memory corresponding to the read address is stored in the User Read Data FIFOs. The user can access the data read from memory by reading the User Read Data FIFOs.

Figure 4-10 shows the QDRII memory controller state machine for burst lengths of four. The controller state machine is in the IDLE state when the calibration is complete. When the User Write Data FIFO and the User Write Address FIFO are not empty (that is, when there are user-written write data and write address bits in the corresponding FIFOs), the state machine goes to the WRITE state, initiating a memory write of one complete burst.

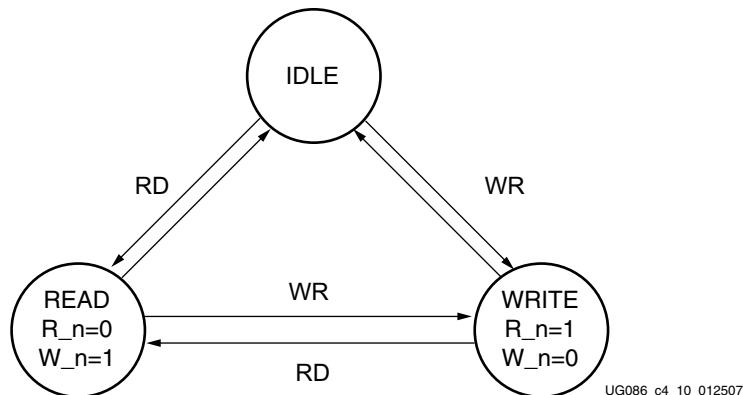


Figure 4-10: QDRII Memory Controller State Machine with Burst Lengths of 4

When the User Read Address FIFO is not empty (that is, the user has written read address bits into the User Read Address FIFO) and either Read Data FIFO is not full, 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 the not empty status of the Write Address FIFO and the Write Data FIFOs or the Read Address FIFO, and not full status of the Read Data FIFOs, respectively. Writes are given priority. In the WRITE state, a memory write is initiated, and the User Read Address Not Empty and User Read Data FIFOs full status are checked to transfer into the READ state. When the User Read Address FIFO is empty, or the User Read Data FIFOs are full, 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 4-11 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, when there are user-written write data and write address bits in the corresponding FIFOs), the state machine goes to the READ\_WRITE state, initiating a memory write of one complete burst, or when the User Read Address FIFO is not empty, that is, the user has written read address bits into the User Read Address FIFO, and the User Read Data FIFOs are not full, the state machine goes to the READ\_WRITE state, initiating a memory read of one complete burst.

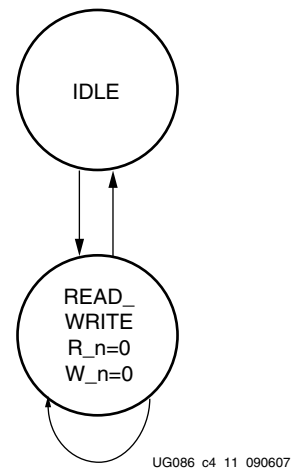


Figure 4-11: QDR II Memory Controller State Machine with Burst Lengths of 2

From the IDLE state, the QDR II memory controller goes to READ\_WRITE state if either:

- the User Write Address FIFO and the User Write Data FIFO are not empty or,
- the User Read Address FIFO is not empty and the User Read Data FIFOs are not full

In the READ\_WRITE state, the User Read Address Not Empty and User Read Data FIFOs Not Full status are checked to initiate a memory read. To initiate a memory write in the READ\_WRITE state, the User Write Data FIFOs and the User Write Address FIFO Not Empty status are checked. If both the User Write Data FIFOs and User Write Address FIFO are empty, and the User Read Address FIFO is empty, or the User Read Data FIFOs are full, the state machine goes to the IDLE state. If the User Write Data FIFO and User Write Address FIFO are not empty, or the User Read Address FIFO is not empty and the User Read Data FIFO is not full, the state machine remains in the READ\_WRITE state to issue memory writes or reads.

Refer to XAPP703 [Ref 19] for detailed design and timing analysis of the QDR II 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 XAPP703 [Ref 19] 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 for burst lengths of 4 and on both the edges of the clock cycle for burst lengths of 2.

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. The free-running strobe from the memory CQ is captured using the FPGA clock. Thus the relation between the CQ strobe and the FPGA clock is found, and the strobe CQ is center-aligned with the FPGA clock. The same logic is applied to the read data Q window, which is center-aligned with the same FPGA clock. This in turn means that the same amount of tap delays are applied to both Q and CQ through IDELAY elements to center-align the Q and CQ windows with respect to the FPGA clock. By center-aligning the Read Data window Q with respect to the FPGA clock, the data capturing logic is complete.

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 19] for details about the delay calibration.

## Infrastructure

The Infrastructure (infrastructure\_top) module comprises the reset logic generation circuitry and instantiates a DCM primitive for clock source generation. Inputs to the infrastructure\_top module are REFCLK\_P and REFCLK\_N (the differential clock pair for the entire design), DLY\_CLK\_200\_P and DLY\_CLK\_200\_N (the differential clock pair for the IDELAYCTRL elements) and SYS\_RST\_N (the user reset signal). REFCLK\_P and REFCLK\_N are used by the DCM primitive to generate the clock and the 270° phase-shifted version of the clock. This module generates multiple reset signals, each synchronous to its respective clock domain for the controller design.

## 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 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).

## QDRII SRAM Initialization and Calibration

QDRII memory is initialized through a specified sequence. The QDRII device requires 2048 clock cycles of clock input after its DLL has been enabled. After the DCM clocks are stable, the controller waits for a specified amount of time before asserting the QDR\_DLL\_OFF\_n signal to the memory. This signal can also be pulled up to a High on the memory device without being driven from the FPGA.

Any command can be issued to the memory only after the 2048 clock cycle wait time. After 2048 clock cycles, the INIT\_DONE signal is asserted indicating the completion of the initialization sequence. Following initialization, the relationship between the data and the FPGA clock is calculated using the TAP logic. The memory strobe CQ is a free-running clock from the memory component. Because the read data Q and the memory strobe CQ are edge-aligned, the strobe is passed through the IDELAY elements of the Virtex-4 device and the taps are adjusted to center-align the strobe pulse with respect to the FPGA clock. The same number of taps are applied to the data window's IDELAY element to center-align the data window with respect to the FPGA clock. XAPP701 [Ref 17] provides more information about the calibration architecture.

Calibration is done in two stages:

1. In the first stage of calibration, the read strobe CQ is center-aligned with respect to the FPGA clock. CQ is a free-running clock from QDRII memory. The read data Q is edge-aligned with the read strobe CQ. The first and second edges of the CQ strobe are detected using the FPGA clock to determine the center of the CQ window.

Once the CQ window is center-aligned with the FPGA clock, the same amount of delay (tap counts) is applied to the read data windows Q through the IDELAY element, so that the Q window is center-aligned with the FPGA clock.

Port `cq_q_cal_done` in the `data_path` module indicates the status of the first stage calibration. When `cq_q_cal_done` is asserted High, it indicates the completion of first stage calibration. After the first stage calibration is complete, the second stage calibration starts.

2. In the second stage of calibration, the write enable signal for the Read Data FIFO is determined by delaying the controller-issued read command. This delay is calibrated based on the delay between the read command and the corresponding read data at the Read Data FIFO. For this delay calibration, the controller writes a known fixed pattern of data into a memory location and reads back from the same location. This read data is compared against the known fixed pattern. The delay between the read command and the correct pattern read data comparison is the delay calibration.

The `final_dly_cal_done` port in the `data_path` module indicates the status of the second stage calibration. When `final_dly_cal_done` is asserted High, it indicates the completion of second stage calibration, which implies the completion of the whole initialization and calibration process. After the initialization and calibration is done (i.e., the `dly_cal_done` signal in `design_top` is asserted High), the controller can start issuing user commands to the memory.

In the second stage calibration, when the pattern read data does not match with the pattern write data, the controller does not issue any further pattern read commands, and the controller gets stuck in the calibration state. The design must be restarted for the calibration to start from the beginning.

## QDRII Controller System and User Interface Signals

Table 4-2 through Table 4-3 describe the QDRII controller system interface signals with and without a DCM, respectively. Table 4-4 describes the QDRII user interface signals without a testbench. Table 4-5 describes the QDRII memory interface signals. In these tables, all signal directions are with respect to the QDRII memory controller.

Table 4-2: QDRII SRAM System Interface Signals (with a DCM)

Signal Name	Direction	Description
REFCLK_P, REFCLK_N	Input	Reference 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.  When the Without a DCM option is selected, this clock pair is not present.
DLY_CLK_200_P, DLY_CLK_200_N	Input	200 MHz differential clock used in the idelay_ctrl logic.
SYS_RST_N	Input	Reset to the QDRII memory controller.
COMPARE_ERROR	Output	This signal represents the status of the comparison between the read data and the corresponding write data.
DLY_CAL_DONE	Output	This signal is asserted when the design initialization and calibration is complete.

Table 4-3: QDRII SRAM System Interface Signals (without a DCM)

Signal Name	Direction	Description
CLK_0	Input	Input clock
CLK_270	Input	Input clock with a 270° phase difference.
CLK_200	Input	200 MHz clock for the IDELAYCTRL primitives.
DCM_LOCKED	Input	This active-High signal indicates whether the user DCM is locked or not.
SYS_RST_N	Input	Reset to the QDRII memory controller.
COMPARE_ERROR	Output	This signal represents the status of the comparison between the read data and the corresponding write data.
DLY_CAL_DONE	Output	This signal is asserted when the design initialization and calibration is complete.



Table 4-4: QDRII SRAM User Interface Signals (without a Testbench)

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, all writes to the User Read Address FIFO are ignored.
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, all reads to the User Read Data FIFO are invalid.
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.
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_DWL [(data_width-1):0]	Input	Positive-edge data for memory writes. This data bus is valid when USER_W_n is asserted.
USER_DWH [(data_width-1):0]	Input	Negative-edge data for memory writes. This data bus is valid when USER_W_n is asserted.
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_BWL_n [(BW_width-1):0]	Input	Byte enables for QDRII memory positive-edge write data. These byte enables are valid when USER_W_n is asserted.
USER_BWH_n [(BW_width-1):0]	Input	Byte enables for QDRII memory negative-edge write data. These byte enables are valid when USER_W_n is asserted.
USER_AD_WR [(addr_width-1):0]	Input	QDRII memory address for write data. This bus is valid when USER_W_n is asserted.
USER_AD_RD [(addr_width-1):0]	Input	QDRII memory address for read data. This bus is valid when USER_R_n is asserted.

Table 4-4: QDRII SRAM User Interface Signals (without a Testbench) (Continued)

Signal Name	Direction	Description
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.
USER_W_n	Input	This active-Low signal is the write enable for the User Write Data and User Write Address FIFOs. 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_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.

**Notes:**

1. All user interface signal names are prepended with a controller number, for example, cntrl0\_QDR\_Q. QDRII SRAM devices currently support only one controller.

Table 4-5: QDRII SRAM Interface Signals

Signal Name	Direction	Description
QDR_D	Output	During WRITE commands, the data is sampled on both edges of K.
QDR_Q	Input	During READ commands, the data is sampled on both edges of the FPGA clk.
QDR_BW_N	Output	Byte enables for QDRII memory write data. The byte enables are valid when USER_W_n is asserted
QDR_SA	Output	Address for READ and WRITE operations.
QDR_W_N	Output	This signal represents the WRITE command.
QDR_R_N	Output	This signal represents the READ command.
QDR_CQ	Input	This read data clock transmitted by the QDRII SRAM is edge-aligned with the read data.
K, K_N	Output	Differential write data clocks.
C, C_N	Output	Input clock for output data.
QDR_DLL_OFF_n	Output	The DLL is disabled when this signal is Low.

## Write Interface

Figure 4-12 illustrates the user interface block diagram for write operations.

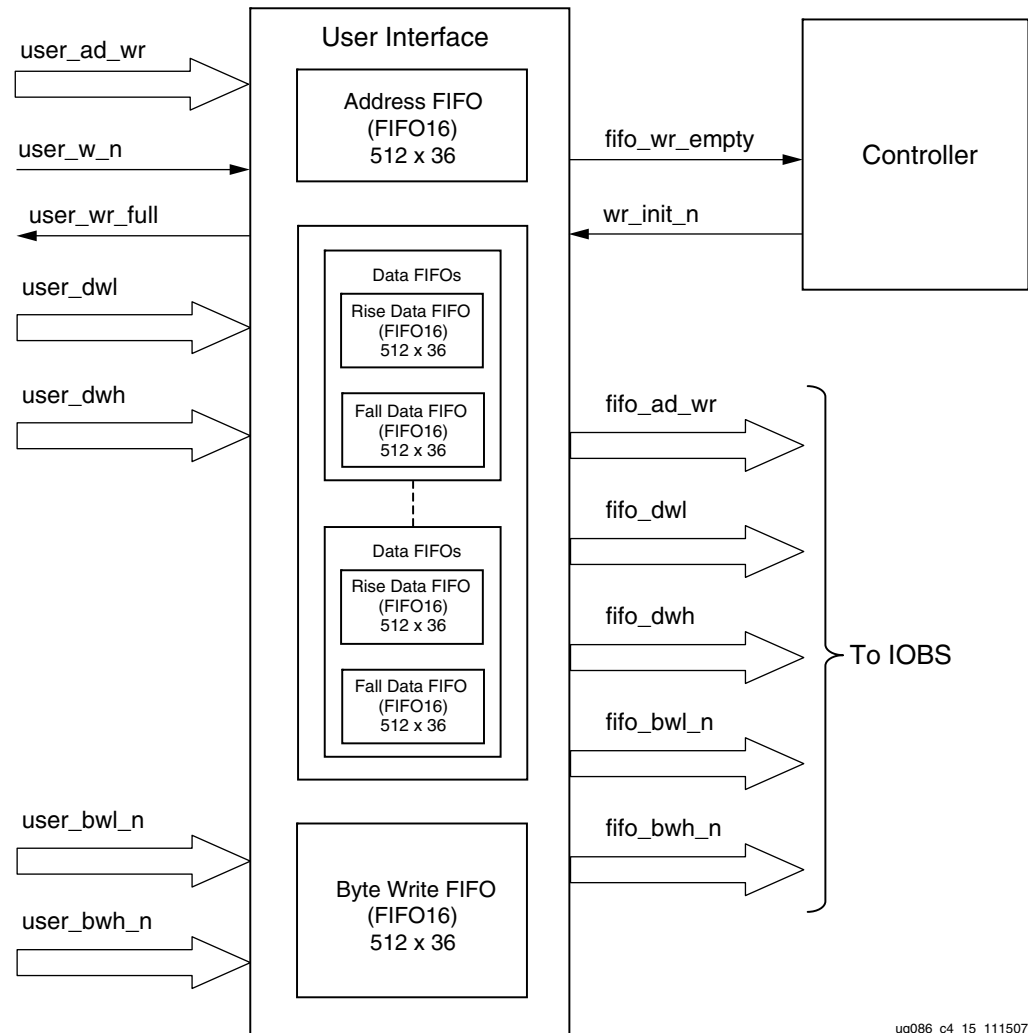
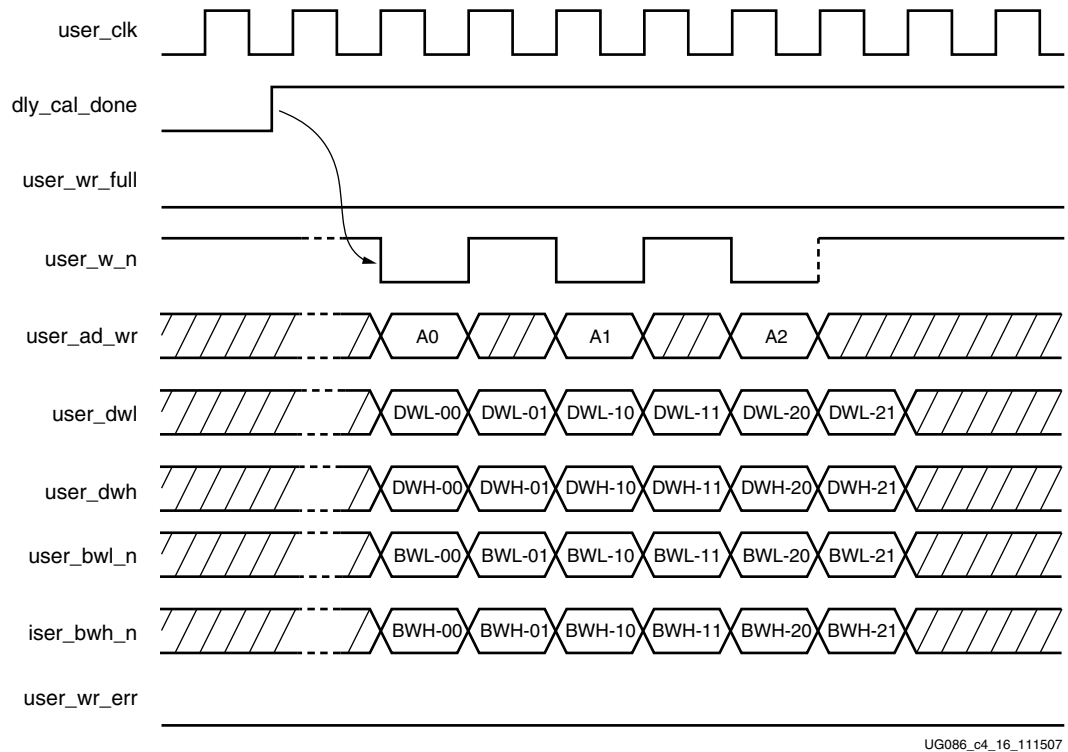


Figure 4-12: Write User Interface Block Diagram

The following steps describe the architecture of Address and Write Data FIFOs and how to perform a write burst operation to QDRII memory from user interface.

1. The user interface consists of an Address FIFO, Data FIFOs and a byte write FIFO. These FIFOs are built out of Virtex-4 FIFO16 primitives of configuration 512x36.
2. The Address FIFO stores the QDRII memory address where the data is to be written from the user interface. A single instantiation of a FIFO16 constitutes the Address FIFO.
3. Two separate sets of Data FIFOs store the rising-edge and falling-edge data to be written to QDRII memory from the user interface. For 9-bit, 18-bit, and 36-bit data widths, two FIFO16s are required for storing rising-edge and falling-edge data. For a 72-bit data width, two FIFO16s are required for storing rising-edge data and two FIFO16s for storing falling-edge data. MIG instantiates the required number of FIFOs depending on the memory data width selected. For 9-bit and 18-bit configurations, the controller pads the extra bits of the Data FIFO with 0s.

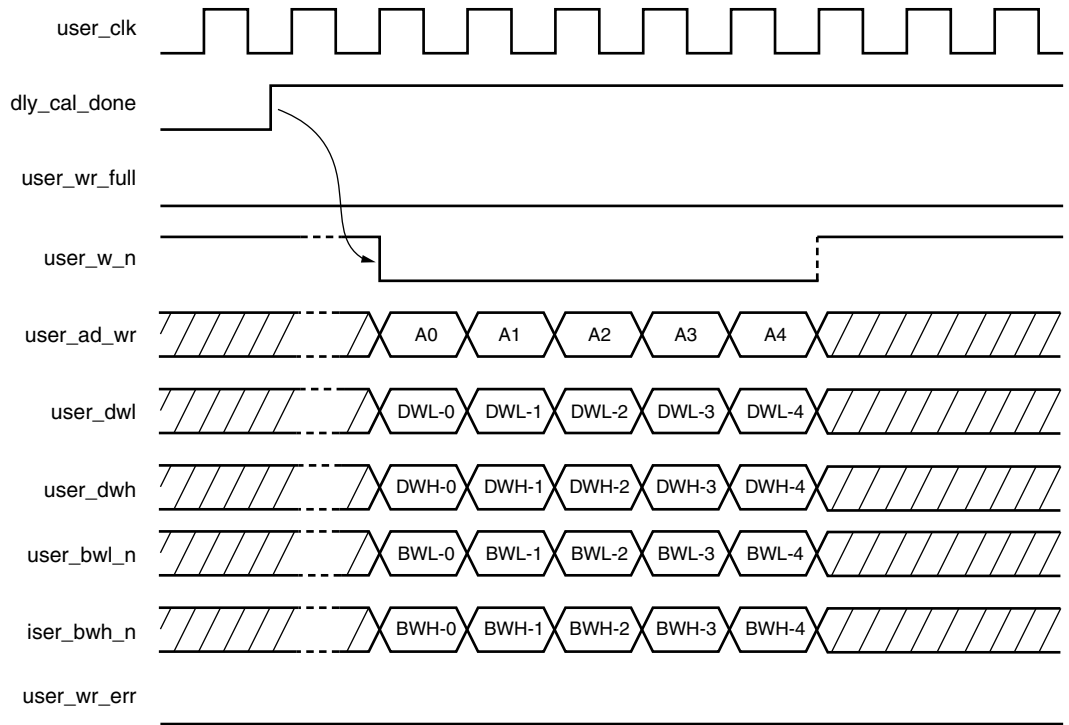
4. The Byte Write FIFO stores the Byte Write signals to QDRII memory from the user interface. Extra bits are padded with zeros.
5. The user can initiate a write command to memory by writing to the Address FIFO, Data FIFOs, and Byte Write FIFOs when FIFO Full flags are deasserted and after the calibration done signal `dly_cal_done` is asserted. Users should not access any of these FIFOs until `dly_cal_done` is asserted. The `dly_cal_done` signal assures that the clocks are stable, the reset process is completed, and the controller is ready to accept commands. Status signal `user_wr_full` is asserted when the Address FIFO, Data FIFOs, or Byte Write FIFOs are full.
6. When `user_w_n` is asserted, `user_ad_wr` is stored in the Address FIFO, `user_dwl` and `user_dwh` are stored in the Data FIFO, and `user_bwl` and `user_bwh` are stored in the Byte Write FIFOs. A common write-enable signal is used to store the data into all three FIFOs.
7. The controller reads the Address, Data, and Byte Write FIFOs when they are not empty by issuing the `wr_init_n` signal. A QDRII memory write command is generated from the `wr_init_n` signal by properly timing it.



UG086\_c4\_16\_111507

Figure 4-13: Write User Interface Timing Diagram for BL = 4

8. Figure 4-13 shows the timing diagram for a write command of BL = 4. The address must be asserted for one clock cycle as shown. For burst lengths of four, each write to the Address FIFO must have two writes to the Data FIFO consisting of two rising edge data and two falling edge data.
9. Figure 4-14 shows the timing diagram for a write command of BL = 2. For a burst length of two, each write to the Address FIFO is coupled to one write to the Data FIFO, consisting of one rising edge data and one falling edge data. For BL = 2, commands can be given in every clock.

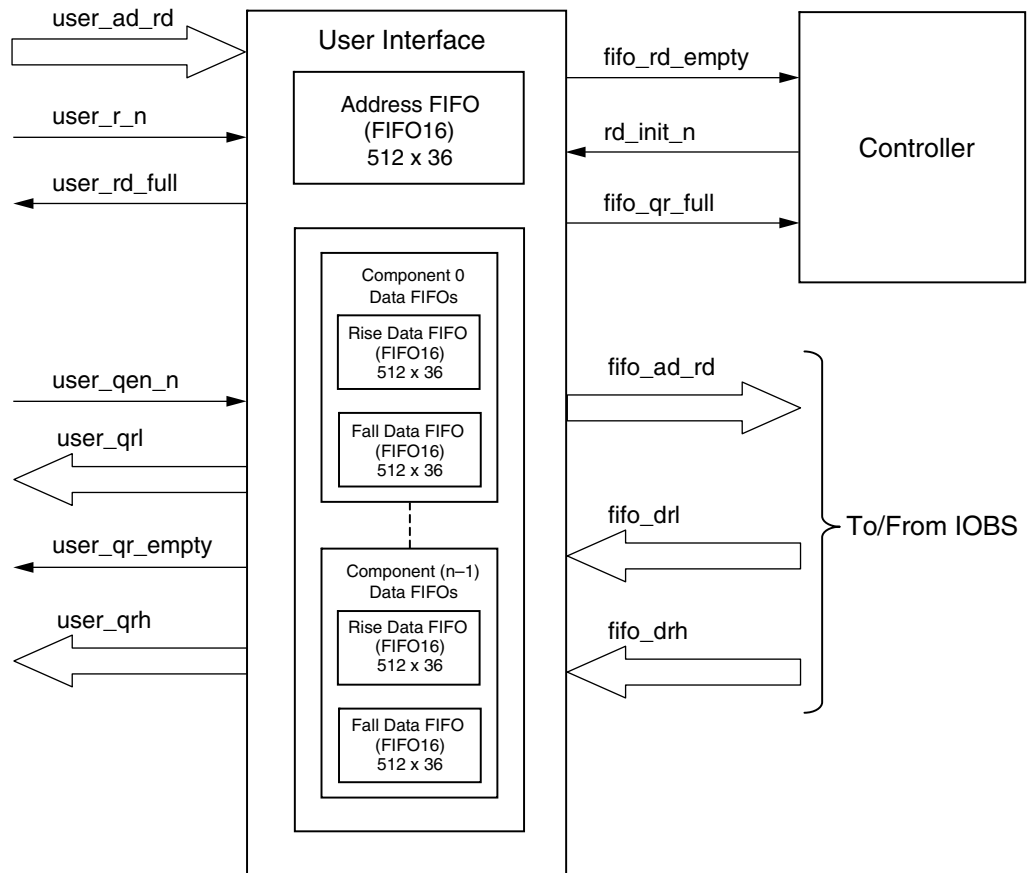


UG086\_c4\_17\_010108

Figure 4-14: Write User Interface Timing Diagram for BL = 2

## Read Interface

Figure 4-15 shows a block diagram for the read interface.



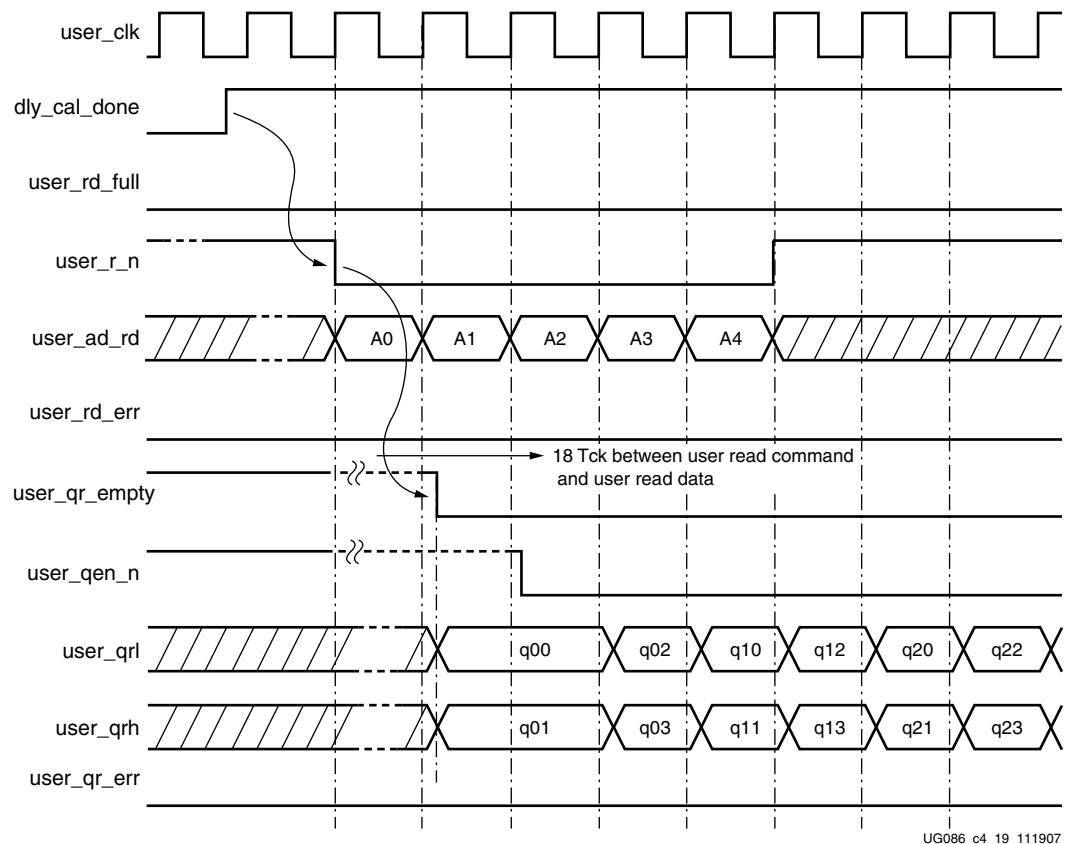
ug086\_c4\_18\_111507

Figure 4-15: Read User Interface Block Diagram

The following steps describe the architecture of the Read Data FIFOs and show how to perform a QDRII SRAM burst read operation from the user interface.

1. The read user interface consists of an Address FIFO and a Read Data FIFO. The Address FIFO and Read Data FIFO are built from Virtex-4 FIFO16s of configuration 512 x 36.
2. The size of the Address FIFO is always of 512 x 16.
3. The number of Read Data FIFOs required depends on the number of QDRII components being used. Using 9-bit components for 36-bit data width, a total of eight FIFOs are required, four for rising-edge data and four for falling-edge data. Although each FIFO can accommodate 36-bit data, the requirement of having one FIFO per component arises from CQ pattern calibration, where an internal pattern calibration is done per CQ. The controller generates the Read Data FIFO write-enable signal for each FIFO separately depending on the CQ pattern calibration.
4. To initiate a QDRII read command, the user must write the Address FIFO when the FIFO full flag user\_rd\_full is deasserted and the calibration done signal dly\_cal\_done is asserted. Writing to the Address FIFO indicates to the controller that it is a Read command. The dly\_cal\_done signal assures that the controller clocks are stable, the internal reset process is completed, and the controller is ready to accept commands.

5. The user must issue an Address FIFO write-enable signal `user_r_n` along with the read address `user_ad_rd` to write the read address to the Address FIFO.
6. The controller reads the Address FIFO when status signal `fifo_rd_empty` is deasserted and generates the appropriate control signals to QDRII memory required for a read command.
7. Prior to the actual read and write commands, the design calibrates the latency (number of clock cycles) from when the read command is issued to when the data is received. Using this precalibrated delay information, the controller generates the write-enable signals to the Read Data FIFOs. The delay calibration is done per QDRII component.
8. The Low state of `user_qr_empty` indicates read data is available. Asserting `user_qen_n` reads rising-edge data and falling-edge data simultaneously on every rising edge of the clock.
9. [Figure 4-16](#) and [Figure 4-17](#) show the user interface timing diagrams for  $BL = 4$  and  $BL = 2$ .
10. After the address is loaded into the Address FIFO, it can take 18 clock cycles (worst case) for the controller to write the Data FIFOs.



UG086\_c4\_19\_111907

Figure 4-16: Read User Interface Timing Diagram for BL = 4

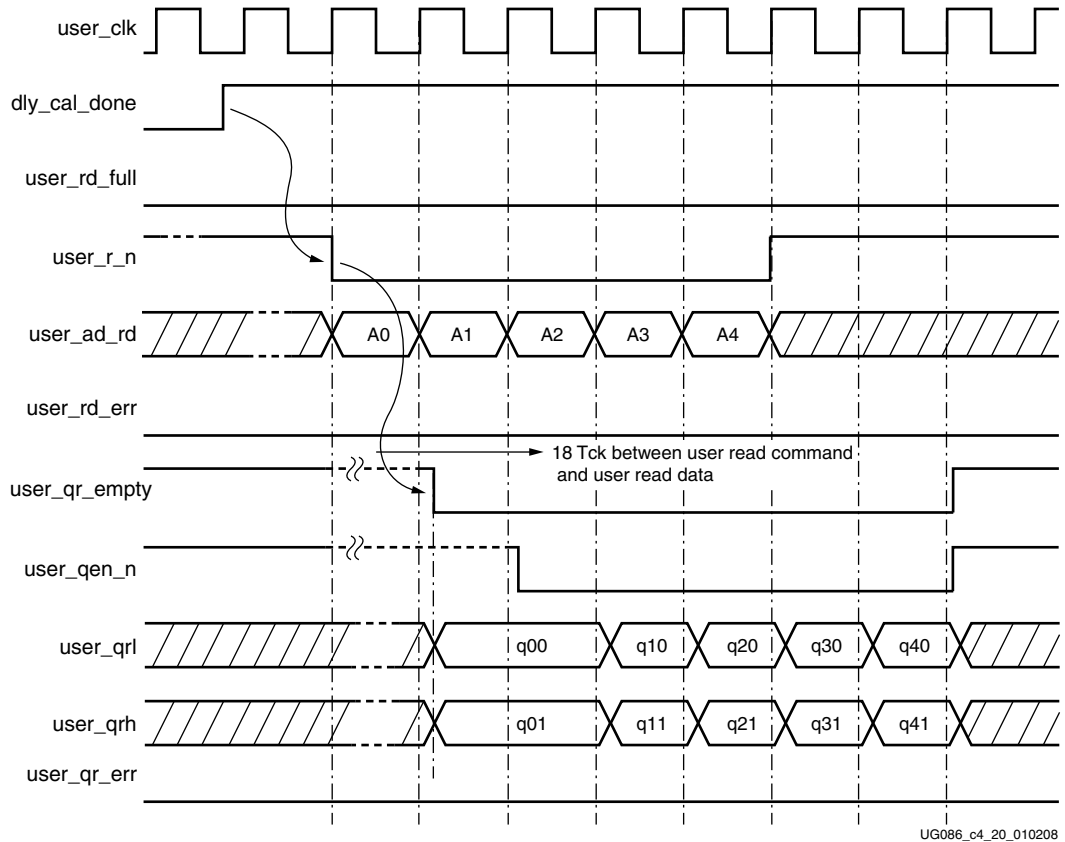


Figure 4-17: Read User Interface Timing Diagram for BL = 2

Table 4-6 shows the maximum read latency of the controller.

Table 4-6: Maximum Read Latency

Parameter	Number of Clocks	Description
User command to address FIFO empty flag	5 (2 + 3)	Two clocks for the two-stage pipeline before the FIFO input. An empty FIFO takes three clocks to deassert the empty status signal after the FIFO is written with the first data.
Command from controller state machine to QDR memory	3	One clock cycle to read the FIFO and two clocks for decoding and passing the command to QDR memory.
QDR command to FIFO input data	6	Two clocks for QDRII memory latency, two clocks for calibration delay, and two clocks for the input pipeline.
FIFO input to FIFO output	4	Four clocks to deassert the empty status signal in fall-through mode.
<b>Total Latency</b>	<b>18</b>	<b>Total latency from read command issued to Address FIFO, to data input to user interface.</b>



Table 4-7 shows the list of signals for a QDRII SRAM design allocated in a group from bank selection check boxes in MIG.

Table 4-7: QDRII Signal Allocations

Bank Selected	Signals Allocated in the Group
Address	Memory address and memory control
Data Write	Memory write data and memory byte write
Data Read	Memory read data, memory CQ, and K and C clocks
System Control	System reset from the user interface and status signals
System_Clock	System clocks from the user interface

When the Address box is checked in a bank, the address, QDR\_W\_N, QDR\_R\_N, and QDR\_DLL\_OFF\_n bits are assigned to that particular bank.

When the Data Write box is checked in a bank, the memory data write and memory byte write are assigned to that particular bank.

When the Data Read box is checked in a bank, the memory data read, memory read clocks, memory write clocks, and memory input clock for the output data are assigned to that particular bank.

When the System Control box is checked in a bank, the SYS\_RST\_N, COMPARE\_ERROR, and DLY\_CAL\_DONE bits are assigned to that particular bank.

When the System\_Clock box is checked in a bank, the REFCLK\_P, REFCLK\_N, DLY\_CLK\_200\_P, and DLY\_CLK\_200\_N bits are assigned to that particular bank.

For special cases, such as without a testbench and without a DCM, the corresponding input and output ports are not assigned to any FPGA pins in the design UCF because the user can connect these ports to the FPGA pins or can connect to some logic internal to the same FPGA.

## Supported Devices

The design generated out of MIG is independent of the memory package, hence the package part of the memory component is replaced with X, where X indicates a don't care condition. Table 4-8 shows the list of components supported by MIG.

Table 4-8: Supported Devices for QDRII SRAM

Virtex-4 FPGAs (Verilog and VHDL)		
Components	Make	Configuration
CY7C1314BV18-167BZXC	Cypress	x36
CY7C1315BV18-250BZC	Cypress	x36
CY7C1426AV18-250BZC	Cypress	x9
CY7C1526V18-250BZC	Cypress	x9
CY7C1911BV18-250BZC	Cypress	x9
CY7C1515V18-250BZC	Cypress	x36
K7R160982B-FC25	Samsung	x9

Table 4-8: Supported Devices for QDRII SRAM (Continued)

Virtex-4 FPGAs (Verilog and VHDL)		
Components	Make	Configuration
K7R161882B-FC25	Samsung	x18
K7R161884B-FC25	Samsung	x18
K7R163682B-FC25	Samsung	x36
K7R163684B-FC25	Samsung	x36
K7R320982C-FC20	Samsung	x9
K7R320982M-FC20	Samsung	x9
K7R321882C-FC20	Samsung	x18
K7R321882M-FC20	Samsung	x18
K7R321884C-FC25	Samsung	x18
K7R321884M-FC25	Samsung	x18
K7R323682C-FC20	Samsung	x36
K7R323682M-FC20	Samsung	x36
K7R323684C-FC25	Samsung	x36
K7R323684M-FC25	Samsung	x36
K7R640982M-FC25	Samsung	x9
K7R641882M-FC25	Samsung	x18
K7R641884M-FC25	Samsung	x18
K7R643682M-FC25	Samsung	x36
K7R643684M-FC30	Samsung	x36

## Simulating the QDRII SRAM Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains an external testbench, a memory model, a `.do` file, and an executable file to simulate the generated design. The Samsung memory model files are currently generated in Verilog only. For Cypress memory controller designs, a sample VHDL memory model file is provided. To learn more details about the files in the `sim` folder and to simulate the design, see `simulation_help.chm` in the `sim` folder.

## Hardware Tested Configurations

The frequencies shown in [Table 4-9](#) were achieved on the Virtex-4 FPGA ML461 Memory Interfaces Development Board under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based on a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 72-bit wide interface.

*Table 4-9: Hardware Tested Configurations*

Synthesis Tools	XST and Synplicity
HDL	Verilog and VHDL
FPGA Device	XC4VLX25-FF668-11
Memory Component	K7R163684B-FC25
Burst Length	4
Data Widths	36, 72
36-bit Frequency Range	110 to 350 MHz
72-bit Frequency Range	110 to 320 MHz



## Implementing DDRII SRAM Controllers

---

This chapter describes how to implement DDRII SRAM interfaces for Virtex™-4 FPGAs generated by MIG.

### 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
- With and without a testbench
- With and without a DCM

#### Design Frequency Range

Table 5-1: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-10		-11		-12	
	Min	Max	Min	Max	Min	Max
Component	120	200	120	240	120	250

#### Unsupported Features

The DDRII SRAM controller design does not support:

- DDR SIO memory

## Architecture

Figure 5-1 shows a 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.

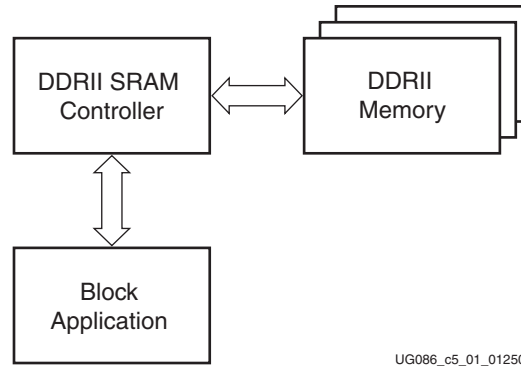


Figure 5-1: DDRII SRAM Controller Interface

Data is double-pumped to DDRII memory on both the positive and the negative edges of the clock. The HSTL\_18 Class II I/O standard is used for data, and the HSTL\_18 Class I I/O standard is used for address, control, and memory clock 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.

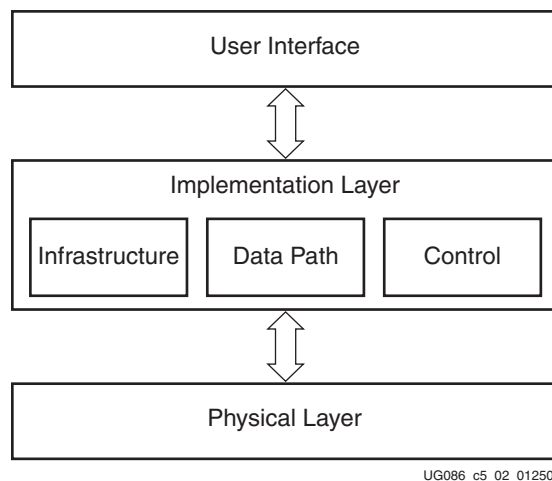


Figure 5-2: Interface Layering Model

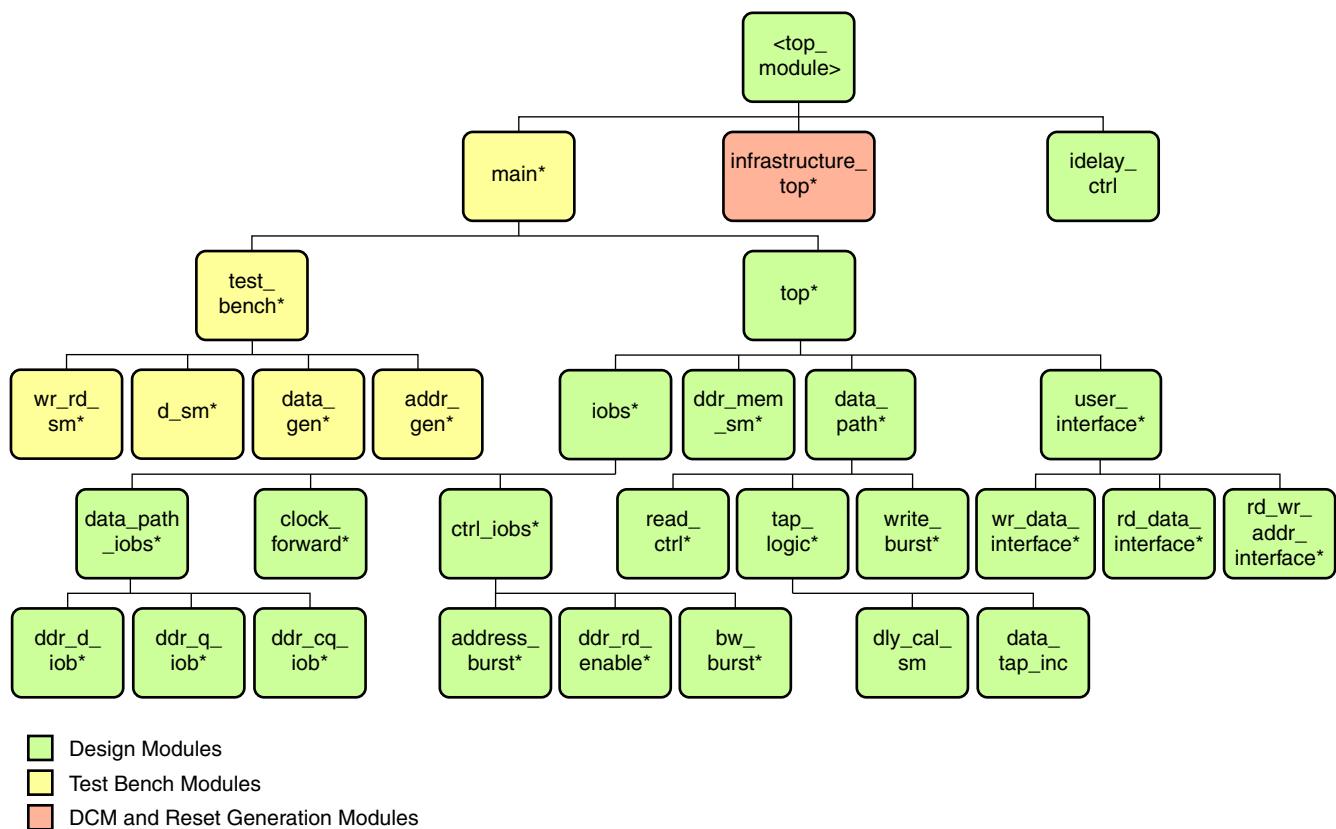
The application layer comprises the user interface, which initiates memory writes and reads by writing data and memory addresses to the User Interface FIFOs. The implementation layer comprises the infrastructure, datapath, and control logic.

- The infrastructure logic consists of the DCM and reset logic generation circuitry.
- The datapath logic consists of the calibration logic by which the data from the memory component is captured using the FPGA clock.
- The control logic determines the type of data transfer, that is, read/write with the memory component, depending on the User Interface FIFO's status signals.

The physical layer comprises the I/O elements of the FPGA. The controller communicates with the memory component using this layer. I/O elements (such as IDDRs, ODDRs, IDELAY, and OFLOPs) are associated with this layer.

## Hierarchy

Figure 5-3 shows the hierarchical structure of the DDRII SRAM design generated by MIG with a testbench and a DCM.



UG086\_c5\_03\_112907

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

The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks and reset generation modules

There is a parameter file generated with the design that has all the user input and design parameters selected from MIG.

MIG can generate DDRII SRAM designs in four different ways:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

For a design without a testbench (user\_design) generated by MIG, the design <top\_module> module has the user interface signals.

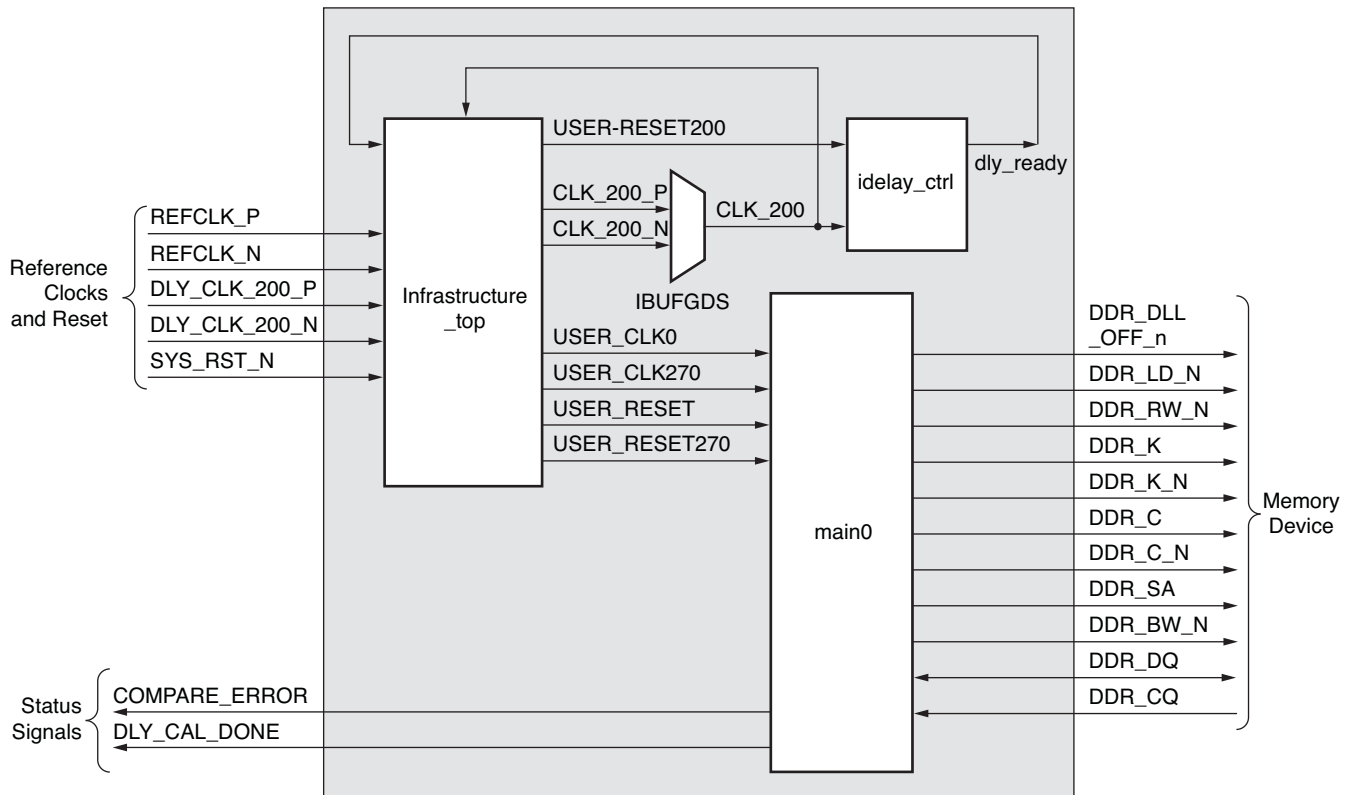
The list of user interface signals is provided in [Table 5-4](#).

Design clocks and resets are generated in the infrastructure\_top module. When **Use DCM** option is checked in MIG, a DCM primitive and the necessary clock buffers are instantiated in the infrastructure\_top module. The inputs to this module are the differential design clock and a 200 MHz differential clock required for the IDELAYCTRL module. A user reset is also input to this module. Using the input clocks and reset signals, the system clocks and system resets used in the design are generated in this module.

When the **Use DCM** option is unchecked in MIG, the infrastructure\_top module does not have the DCM and the corresponding clock buffer instantiations. Therefore, the system operates on the user-provided clocks. The system reset is generated in the infrastructure\_top module using the DCM\_LOCK signal and the ready signal of the IDELAYCTRL element.



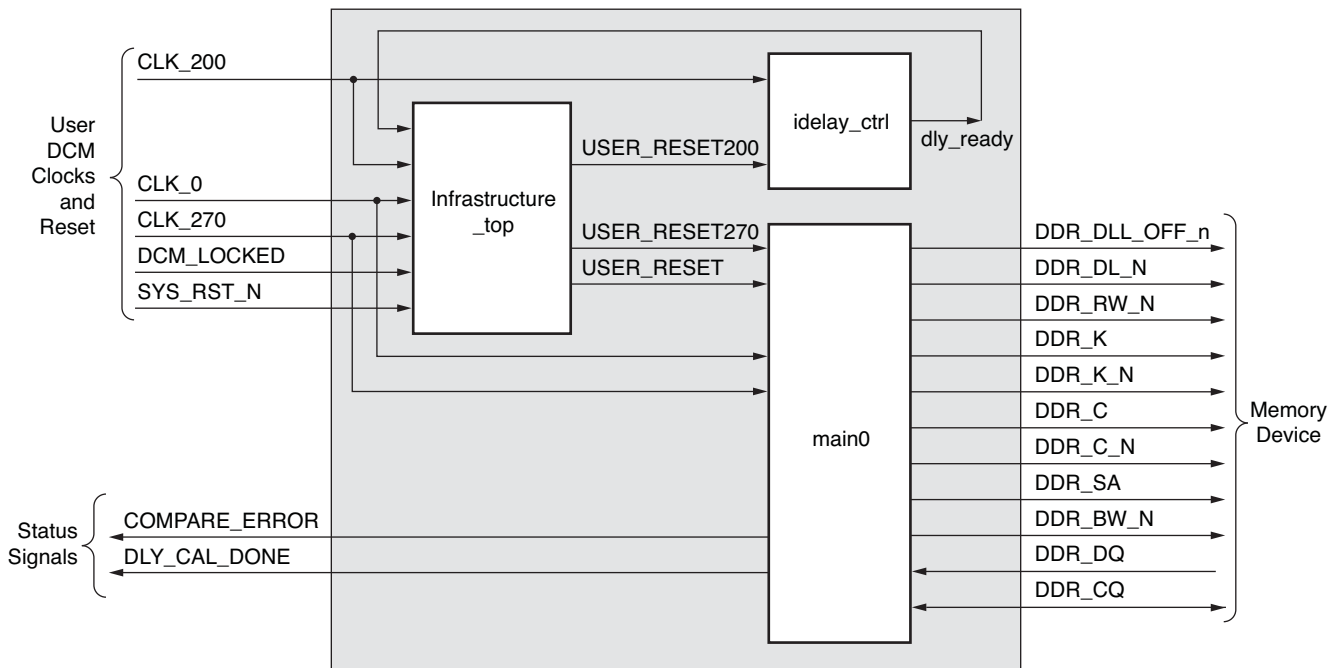
Figure 5-4 shows a top-level block diagram of a DDRII SRAM design with a DCM and a testbench. REFCLK\_P and REFCLK\_N are differential input reference clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. DLY\_CLK\_200\_P and DLY\_CLK\_200\_N are used for the IDELAYCTRL element. SYS\_RST\_N is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the SYS\_RST\_N signal, and the dly\_ready signal of the IDELAYCTRL element. The COMPARE\_ERROR output signal indicates whether the design passes or fails. The DLY\_CAL\_DONE signal indicates the completion of initialization and calibration of the design. Because the DCM is instantiated in the infrastructure module, it generates the required clocks and resets signals for the design.



UG086\_c5\_04\_013007

Figure 5-4: Top-Level Block Diagram of the DDRII SRAM Design with a DCM and a Testbench

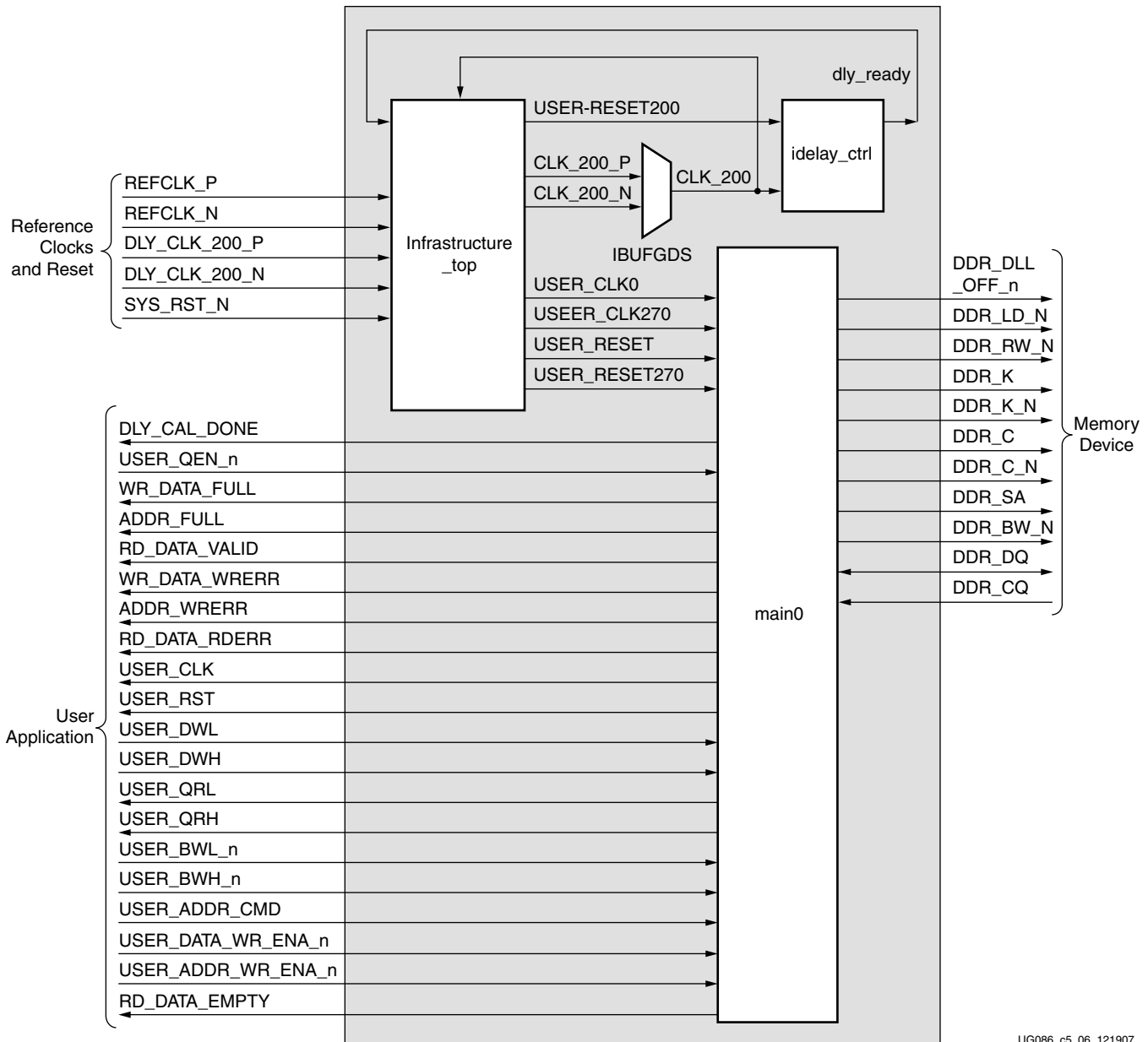
Figure 5-5 shows a top-level block diagram of a DDRII SRAM design with a testbench but without a DCM. The user should provide all the clocks and the DCM\_LOCKED signal. These clocks should be single-ended. SYS\_RST\_N is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the SYS\_RST\_N signal, and the dly\_ready signal of the IDELAYCTRL element. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFs. The COMPARE\_ERROR output signal indicates whether the design passes or fails. The testbench module does writes and reads, and also compares the read data with the written data. The COMPARE\_ERROR signal is driven High on data mismatches. The DLY\_CAL\_DONE signal indicates the completion of initialization and calibration of the design.



UG086\_c5\_05\_013007

Figure 5-5: Top-Level Block Diagram of the DDRII SRAM Design without a DCM but with a Testbench

Figure 5-6 shows a top-level block diagram of a DDRII SRAM design with a DCM but without a testbench. REFCLK\_P and REFCLK\_N are differential input reference clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. DLY\_CLK\_200\_P and DLY\_CLK\_200\_N are used for the IDELAYCTRL element. SYS\_RST\_N is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the SYS\_RST\_N signal, and the dly\_ready signal of the IDELAYCTRL element. The user has to drive the user application signals. The design provides the USER\_CLK and USER\_RST signals to the user to synchronize the user application signals with the design. The DLY\_CAL\_DONE signal indicates the completion of initialization and calibration of the design.



UG086\_c5\_06\_121907

Figure 5-6: Top-Level Block Diagram of the DDRII SRAM Design with a DCM but without a Testbench

Figure 5-7 shows a top-level block diagram of a DDRII SRAM design without a DCM or a testbench. The user should provide all the clocks and the DCM\_LOCKED signal. These clocks should be single-ended. SYS\_RST\_N is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the SYS\_RST\_N signal, and the dly\_ready signal of the IDELAYCTRL element. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFPGs. The user has to drive the user application signals. The design provides the USER\_CLK and USER\_RST signals to the user to synchronize the user application signals with the design. The DLY\_CAL\_DONE signal indicates the completion of initialization and calibration of the design.

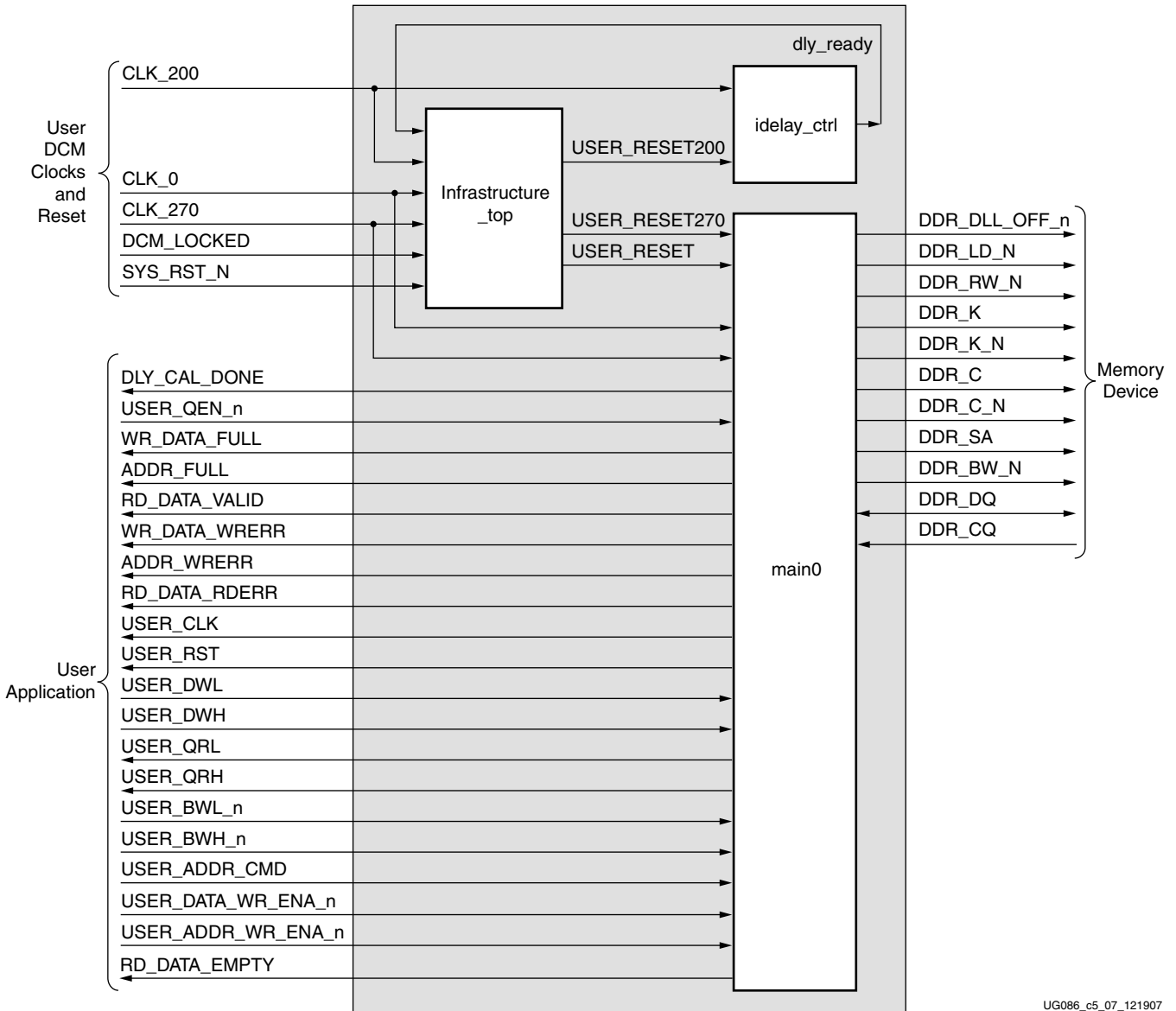


Figure 5-7: Top-Level Block Diagram of the DDRII SRAM Design without a DCM or a Testbench

## DDRII SRAM Controller Modules

Figure 5-8 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.

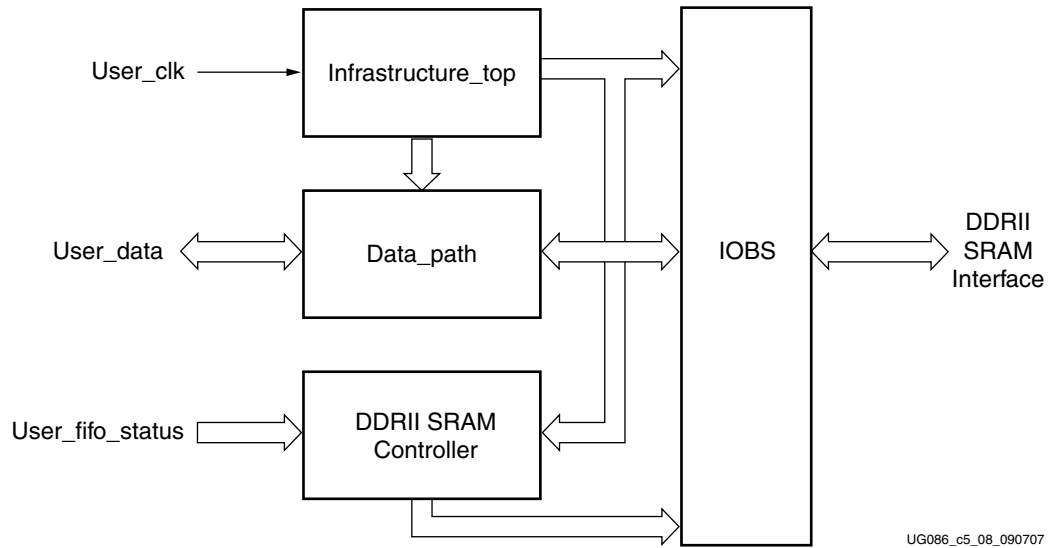


Figure 5-8: DDRII SRAM Controller Modules

Figure 5-9 shows the DDRII SRAM controller modules with a 36-bit interface.

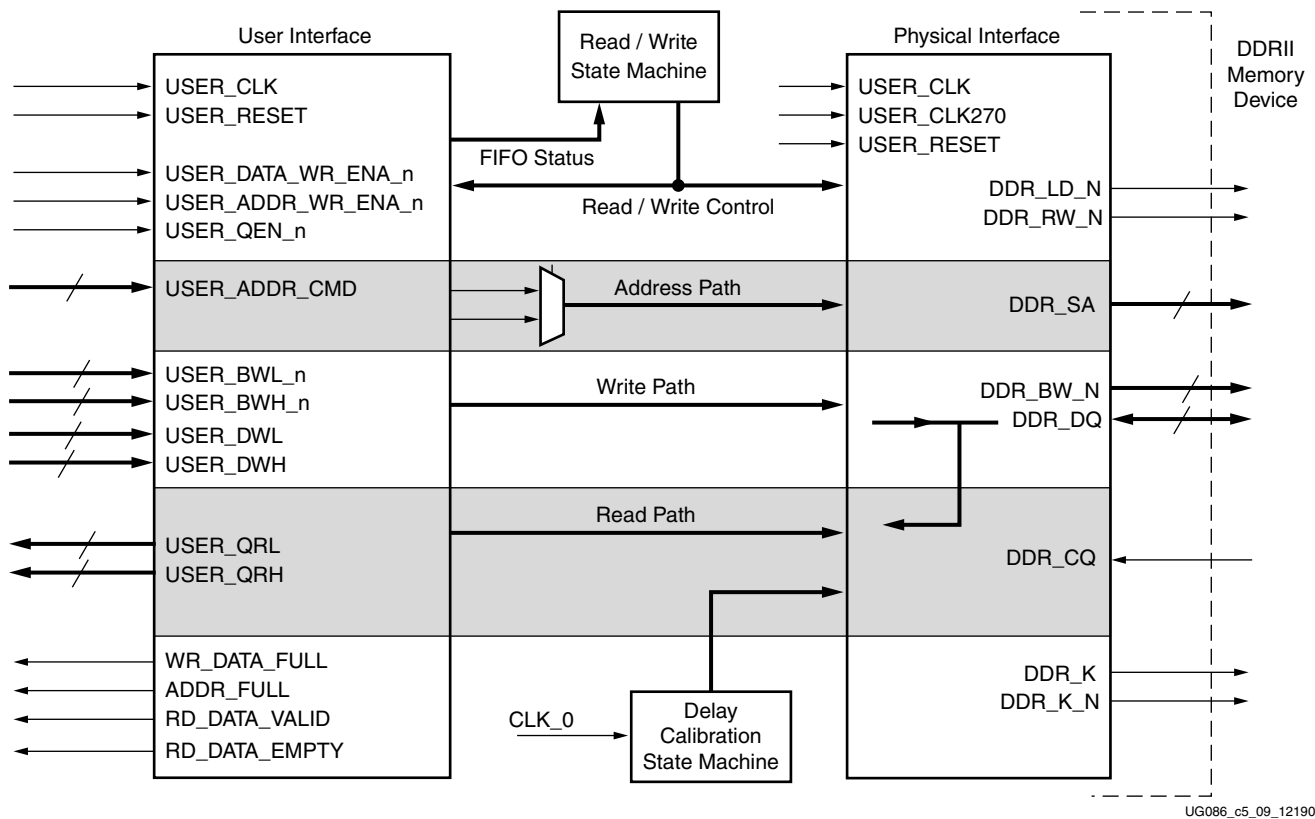


Figure 5-9: DDRII SRAM Controller Modules with Interface Signals

## Controller

The DDRII SRAM controller initializes the memory, accepts and decodes the user commands, and generates the READ and WRITE commands. It also generates control signals for other modules. After power on it starts the calibration, after the calibration is completed it process the READ or WRITE commands.

## 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 XAPP703 [Ref 19] 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, CQ, to which read data is synchronized, is a free-running strobe. The free-running strobe from the memory CQ is captured using the FPGA clock. Thus the relation between the CQ strobe and FPGA clock is found, and the strobe CQ is center-aligned with the FPGA clock by delaying the CQ strobe in the IDELAY element. The same logic is applied to the read data window. The read data window is center-aligned with the same FPGA clock. This in turn means that the same amount of tap delays are applied on both the read data window and the strobe CQ through the IDELAY elements to center-align the read data and strobe CQ windows with respect to the FPGA clock. Center-aligning the read data window with respect to the FPGA clock completes the data capturing logic.

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 19] for details about the delay calibration.

## Infrastructure

The Infrastructure (infrastructure\_top) module comprises the reset logic generation circuitry and instantiates a DCM primitive for clock source generation. Inputs to the infrastructure\_top module are the REFCLK\_P and REFCLK\_N differential clock pair for the entire design, the DLY\_CLK\_200\_P and DLY\_CLK\_200\_N differential clock pair for the IDELAYCTRL elements, and the user reset signal SYS\_RST\_N. The REFCLK\_P and REFCLK\_N differential clock pair is used by the DCM primitive to generate the clock and the 270° phase-shifted version of the clock. This module generates multiple reset signals, each synchronous to its respective clock domain.

## 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 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).

## DDRII SRAM Initialization and Calibration

DDRII SRAM is initialized through a specified sequence. Following the initialization, the relationship between the read data and the FPGA clock is calculated using the TAP logic. After the DCM clocks are stable, the controller waits for a specified amount of time before asserting the DDR\_DLL\_OFF\_n signal to the memory. This signal can also be pulled up to a High on the memory device without being driven from the FPGA.

The memory strobe CQ is a free-running clock from the memory component. Because the read data and the memory strobe CQ are edge-aligned, the strobe is passed through the IDELAY elements of the Virtex-4 device and the taps are adjusted to center-align the strobe pulse with respect to the FPGA clock. The same number of taps are applied to the data window's IDELAY element to center-align the data window with respect to the FPGA clock. XAPP701 [Ref 17] provides more information about the calibration architecture.

Calibration is done in two stages:

1. In the first stage of calibration, the read strobe CQ is center-aligned with respect to the FPGA clock. CQ is a free-running clock from DDRII SRAM. The read data window is edge-aligned with the read strobe CQ. The first and second edges of the CQ strobe are detected using the FPGA clock to determine the center of the CQ window.

Once the CQ window is center-aligned with the FPGA clock, the same amount of delay (tap counts) is applied to the read data window through the IDELAY element, so that the read data window is center-aligned with the FPGA clock.

Port `cq_q_cal_done` in the `data_path` module indicates the status of the first stage calibration. When `cq_q_cal_done` is asserted High, it indicates the completion of first stage calibration. After the first stage calibration is complete, the second stage calibration starts.

2. In the second stage of calibration, the write enable signal for the Read Data FIFO is determined by delaying the controller-issued read command. This delay is calibrated based on the delay between the read command and the corresponding read data at the Read Data FIFO. For this delay calibration, the controller writes a known fixed pattern of data into a memory location and reads back from the same location. This read data is compared against the known fixed pattern. The delay between the read command and the correct pattern read data comparison is the delay calibration.

The `final_dly_cal_done` port in the `data_path` module indicates the status of the second stage calibration. When `final_dly_cal_done` is asserted High, it indicates the completion of second stage calibration, which implies the completion of the whole initialization and calibration process. After the initialization and calibration is done (i.e., the `dly_cal_done` signal in `design_top` is asserted High), the controller can start issuing user commands to the memory.

In the second stage calibration, when the pattern read data does not match with the pattern write data, the controller does not issue any further pattern read commands and the controller gets stuck in the calibration state. The design must be restarted for the calibration to start from the beginning.

## User Interface

The user interface consists of seven FIFOs. The User Write interface has four FIFOs: one FIFO is used for the memory address, two FIFOs contain positive-edge and negative-edge data for memory, and the remaining FIFO is used for Byte Writes. 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 [Table 5-2](#) for how the user can access these FIFOs.

## DDRII SRAM Controller Interface Signals

[Table 5-2](#) through [Table 5-3](#) describe the DDRII controller system interface signals. [Table 5-4](#) describes the DDRII SRAM user interface signals. [Table 5-5](#) describes the DDRII memory interface signals. In these tables, all signal directions are with respect to the DDRII memory controller.

Table 5-2: DDRII SRAM System Interface Signals (with a DCM)

Signal Name	Direction	Description
REFCLK_P, REFCLK_N	Input	Reference 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.
DLY_CLK_200_P, DLY_CLK_200_N	Input	200 MHz differential clock used in the IDELAY_CTRL logic
SYS_RST_N	Input	Reset to the DDRII memory controller
COMPARE_ERROR	Output	This signal indicates the status of the comparison between the read data with the corresponding write data
DLY_CAL_DONE	Output	This signal is asserted when the design initialization and calibration is complete

Table 5-3: DDRII SRAM System Interface Signals (without a DCM)

Signal Name	Direction	Description
CLK_0	Input	Input clock
CLK_270	Input	Input clock with 270° phase difference
CLK_200	Input	200 MHz clock for IDELAYCTRL primitives
DCM_LOCKED	Input	This active-High signal indicates whether the user DCM is locked or not
SYS_RST_N	Input	Reset to the DDRII memory controller



Table 5-3: DDRII SRAM System Interface Signals (without a DCM) (Continued)

Signal Name	Direction	Description
COMPARE_ERROR	Output	This signal indicates the status of the comparison between the read data with the corresponding write data
DLY_CAL_DONE	Output	This signal is asserted when the design initialization and calibration is complete.

Table 5-4: DDRII SRAM User Interface Signals (without a Testbench)

Signal Name	Direction	Description
WR_DATA_FULL	Output	This signal indicates the User Write FIFOs status. It is asserted when the User Write Data FIFOs are full. When this signal is asserted, any writes to the User Write Data FIFO are invalid, possibly leading to controller malfunction.
ADDR_FULL	Output	This signal indicates the User Read Write Address FIFO status. It is asserted when the User Read Write Address FIFO is full. When this signal is asserted, any writes to the User Read Write Address FIFO are ignored.
RD_DATA_VALID	Output	This signal indicate to the user that data available at read data FIFOs.
WR_DATA_WRERR	Output	This signal is asserted when an error occurs while writing to the User Write Data FIFOs.
ADDR_WRERR	Output	This signal is asserted when an error occurs while writing to the User Read Write Address FIFO.
RD_DATA_RDERR	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
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_DWL [(data_width-1):0]	Input	Positive-edge data for memory writes. The data bus is valid when the WRITE command (DDR_LD_N=0 && DDR_RW_N=0) is asserted.
USER_DWH [(data_width-1):0]	Input	Negative-edge data for memory writes. The data bus is valid when the WRITE command (DDR_LD_N=0 && DDR_RW_N=0) is asserted.
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_BWL_n [(BW_width-1):0]	Input	Byte enables for DDRII memory positive-edge write data. The byte enables are valid when the WRITE command (DDR_LD_N=0 && DDR_RW_N=0) is asserted.

Table 5-4: DDRII SRAM User Interface Signals (without a Testbench) (Continued)

Signal Name	Direction	Description
USER_BWH_n[(BW_width-1):0]	Input	Byte enables for DDRII memory negative-edge write data. The byte enables are valid when the WRITE command (DDR_LD_N=0 && DDR_RW_N=0) is asserted.
USR_ADDR_CMD[addr_width:0]	Input	DDRII memory address for read or write operation. This address is valid when USER_DATA_WR_ENA_n is asserted. An extra bit is driven by the user to represent the command.
USER_QEN_n	Input	This active-Low signal is the read enable for the User Read Data FIFOs. The DDRII 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.
USER_DATA_WR_ENA_n	Input	This active-Low signal is the write enable for the User Write Data FIFOs. 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.
USER_ADDR_WR_ENA_n	Input	This active-Low signal is the write enable for the User Read Write Address FIFO. The user asserts this signal to write write/read address and command in to user read write address FIFO.

**Notes:**

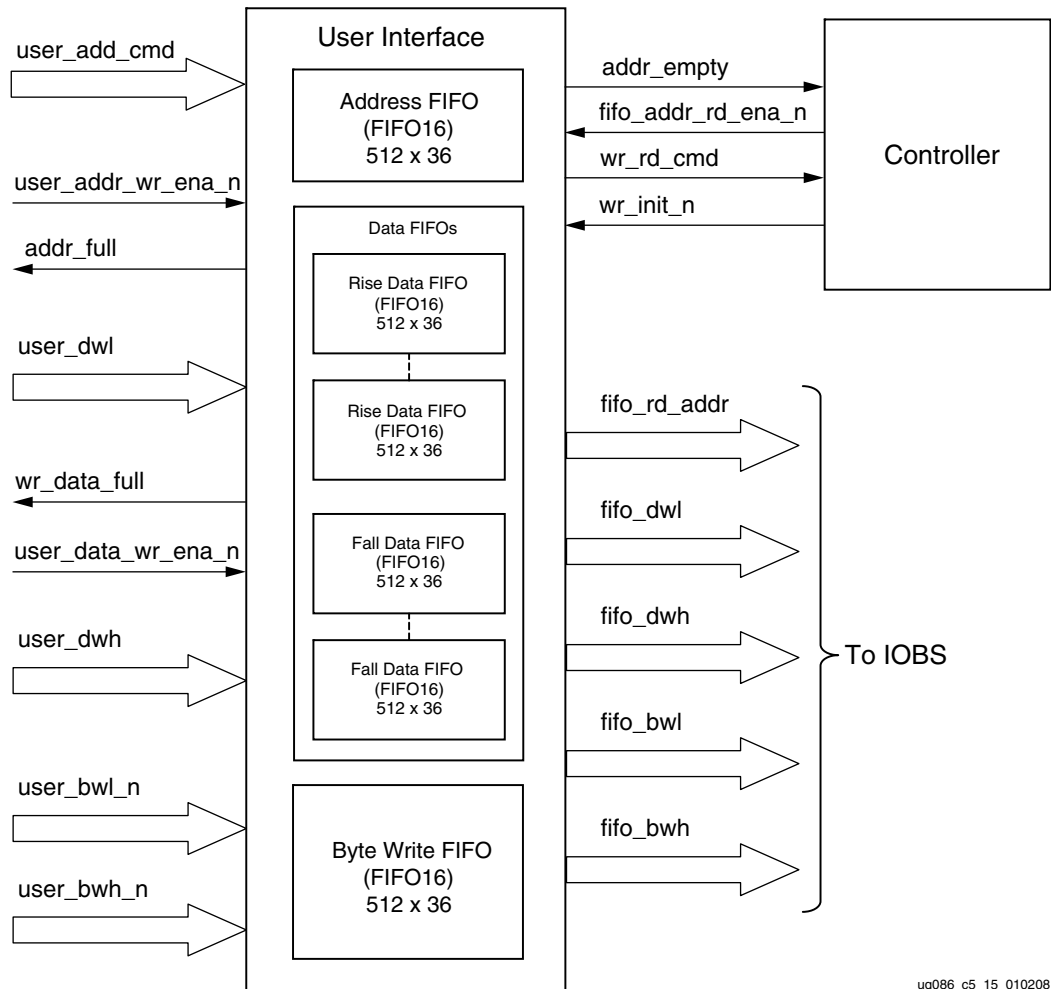
1. All user interface signal names are prepended with a controller number, for example, cntrl0\_DDR\_DQ. DDRII SRAM devices currently support only one controller.

Table 5-5: DDRII SRAM Interface Signals

Signal Name	Direction	Description
DDR_DQ	Input/ Output	Bidirectional data bus. During READ commands, the data is sampled on both edges of the FPGA clk. During WRITE commands, the data is sampled on both edges of the K clk.
DDR_BW_N	Output	Byte enables for DDRII memory write data. The byte enables are valid when the WRITE command (DDR_LD_N=0 && DDR_RW_N=0) is asserted.
DDR_SA	Output	Address for READ and WRITE operations
DDR_LD_N	Output	Synchronous load pin. The bus cycle sequence is to be defined when this signal is Low.
DDR_RW_N	Output	Read/Write control pin. Read is active when High.
DDR_CQ	Input	This read data clock, transmitted by DDRII SRAM, is edge-aligned with read data
K, K_N	Output	Differential write data clocks
C, C_N	Output	Input clock for output data
DDR_DLL_OFF_n	Output	The DLL is disabled when this signal is Low

## Write Interface

Figure 5-10 shows the user interface block diagram for write operations.



ug086\_c5\_15\_010208

Figure 5-10: Write User Interface Block Diagram

The following steps describe the architecture of the Address and Write Data FIFOs and show how to perform a write burst operation to DDRII memory from the user interface.

1. The user interface consists of an Address FIFO, Data FIFOs, and a Byte Write FIFO. These FIFOs are constructed using Virtex-4 FIFO16 primitives with a 512 x 36 configuration.
2. The common Address FIFO is used for both write and read commands, and comprises a command part and an address part. The command bit (bit 0 of the Address FIFO) discriminates between write and read commands; the address starts at bit 1. The command bit should be set to 0 for writes and to 1 for reads.
3. Two separate sets of Data FIFOs are used for storing the rising-edge and falling-edge data to be written to DDRII memory from the user interface. For 9-bit, 18-bit, and 36-bit data widths, two FIFO16s are required for storing rising-edge and falling-edge data. For 72-bit data width, two FIFO16s are required for rising-edge data and two for falling-edge data. MIG instantiates the required number of FIFOs to gain the required

- data width. For 9-bit and 18-bit configurations, the controller pads the extra bits of the Data FIFO with 0s.
4. The Byte Write FIFO is used to store the Byte Write signals to DDRII memory from the user interface. The controller internally pads all zeros for the unused bits.
  5. The user can initiate a write to memory by writing to the Address FIFO, Data FIFOs, and Byte Write FIFO when the FIFO full flags are deasserted and after `dly_cal_done` is asserted. The user should not access any of these FIFOs until `dly_cal_done` is asserted. The `dly_cal_done` signal assures that the clocks are stable, the reset process is completed, and the controller is ready to accept commands. Status signals `addr_full` and `wr_data_full` are asserted when the Address FIFO and Data FIFOs or Byte Write FIFO are full.
  6. When `user_addr_wr_ena_n` is asserted, the user address is stored in the Address FIFO. Similarly, when `user_data_wr_ena_n` is asserted, `user_dw1`, `user_dwh`, `user_bwl`, and `user_bwh` are stored into corresponding FIFOs. A common write-enable signal is used to enable both the Data FIFO and the Byte Write FIFO.
  7. The controller reads the address and decodes the command bit. The write command `wr_init_n` is issued if the command bit is 0 when the Address FIFO is not empty. This command acts as a read-enable to the Data and Byte Write FIFOs. The DDRII memory write command is generated from the `wr_init_n` signal by properly timing it.
  8. [Figure 5-11](#) shows the timing diagram for a write command of BL = 4. The address should be asserted for one clock cycle as shown. For burst lengths of four, each write to the Address FIFO should have two writes to the Data FIFO consisting of two rising-edge data and two falling-edge data.

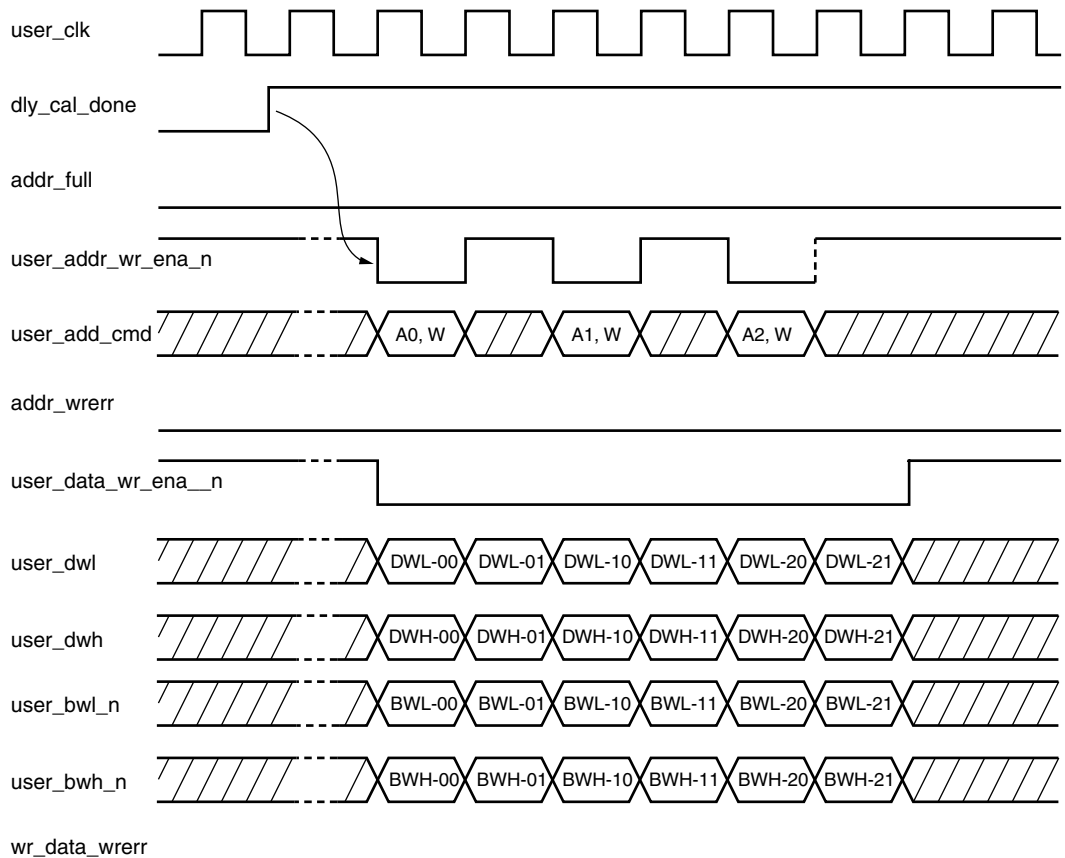
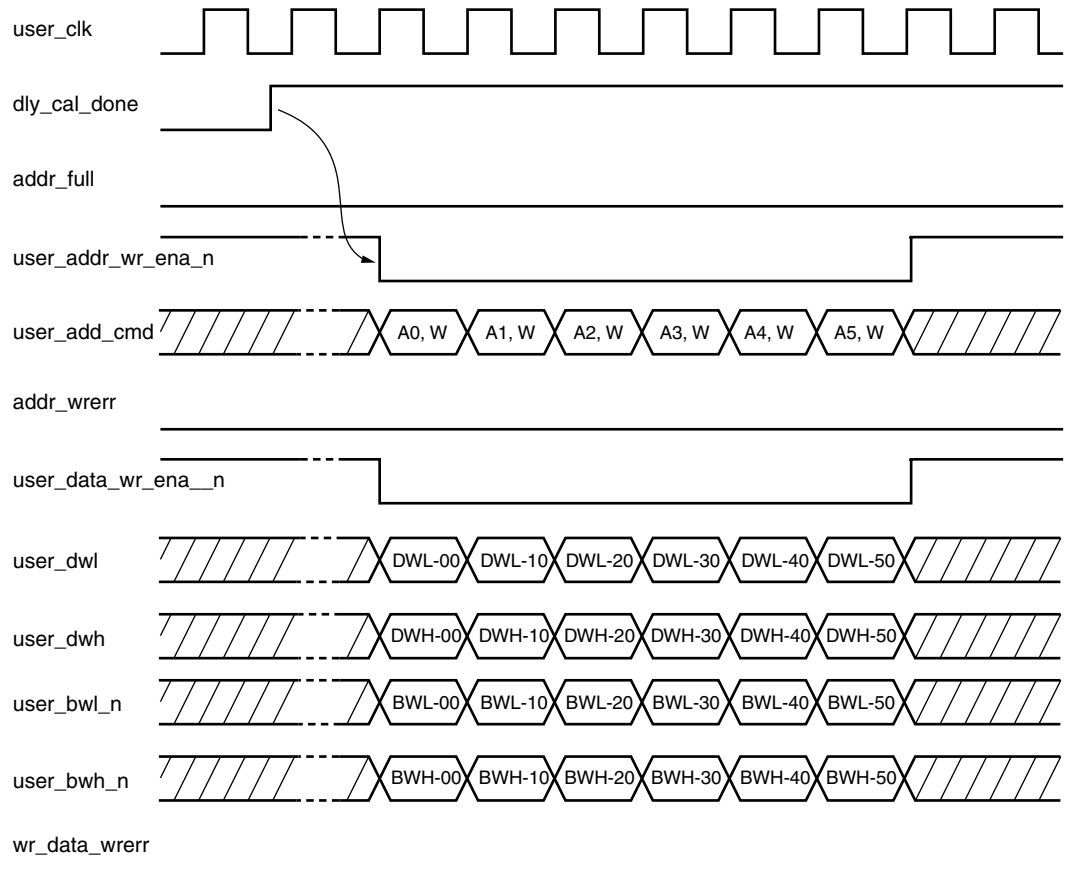


Figure 5-11: Write User Interface Timing diagram for BL = 4

9. [Figure 5-12](#) shows the timing diagram for a write command of BL = 2. For burst length of two, each write to Address FIFO has one write to Data FIFO, consisting of one rising-edge data and one falling-edge data. For burst length of two, commands can be given in every clock.

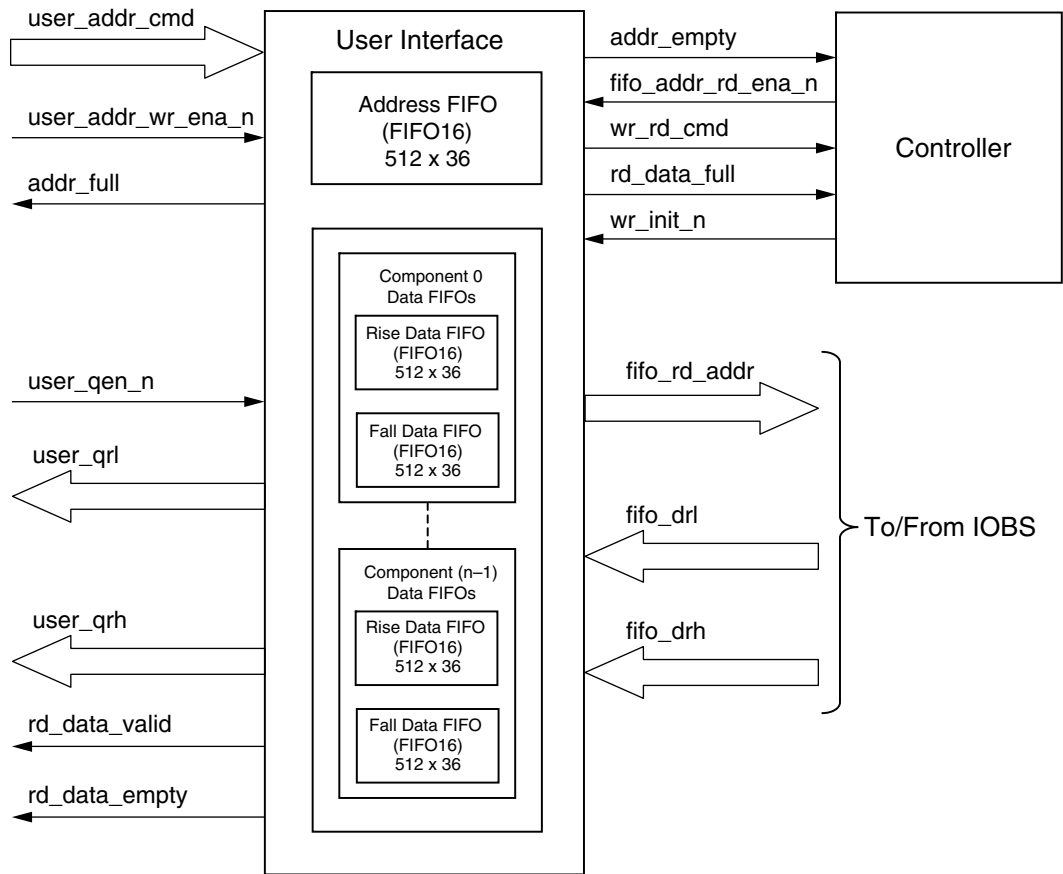


UG086\_c5\_17\_112907

Figure 5-12: Write User Interface Timing diagram for BL = 2

## Read Interface

Figure 5-13 shows the user interface block diagram for read operations.



ug086\_c5\_18\_010108

Figure 5-13: Read User Interface Block Diagram

The following steps describe the architecture of Read Data FIFOs and show how to perform a burst read operation from DDRII SRAM from the user interface.

1. The read user interface consists of a common Address FIFO and a Read Data FIFO. The Address FIFO and Read Data FIFO are constructed using FIFO16s with a 512 x 16 configuration.
2. The number of Read Data FIFOs required depends on the number of DDRII components used. Using 9-bit components for 36-bit data width, a total of eight FIFOs are required, four FIFOs for rising-edge data and four FIFOs for falling-edge data. Though each FIFO can accommodate 36-bit data, the requirement of having one FIFO per component arises from the CQ pattern calibration. Internal pattern calibration is done per CQ. Controller generates the Read Data FIFO write-enable signal for each FIFO separately, depending on the CQ pattern calibration.
3. To initiate a DDRII read command, the user should write the Address FIFO with the command bit set to logic 1 when the FIFO `addr_full` flag is deasserted and the `dly_cal_done` signal is asserted. The `dly_cal_done` signal assures the controller clocks are stable, the internal reset process is completed, and the controller is ready to accept commands.

4. The user should issue the Address FIFO write enable signal `user_addr_wr_ena_n` along with `user_addr_cmd` to write the address to the Address FIFO.
5. When status signal `addr_empty` is deasserted, the controller reads the Address FIFO. If the command bit is 1 when the Read Data FIFO is not full, the appropriate control signal required for a read command is sent to the DDRII memory.
6. Prior to the actual read and write commands, the design calibrates the latency from the time the read command is issued to the time data is received in terms of the number of clock cycles. Using the precalibrated delay information between the read commands to read data, the controller generates the write-enable signals to the Read Data FIFOs. The delay calibration is done per DDRII component.
7. The Low state of `rd_data_empty` indicates read data is available. Asserting `user_qen_n` reads rising-edge data and falling-edge data simultaneously on every rising edge of the clock.
8. [Figure 5-14](#) and [Figure 5-15](#) shows the user interface timing diagrams for  $BL = 2$  and  $BL = 4$ .

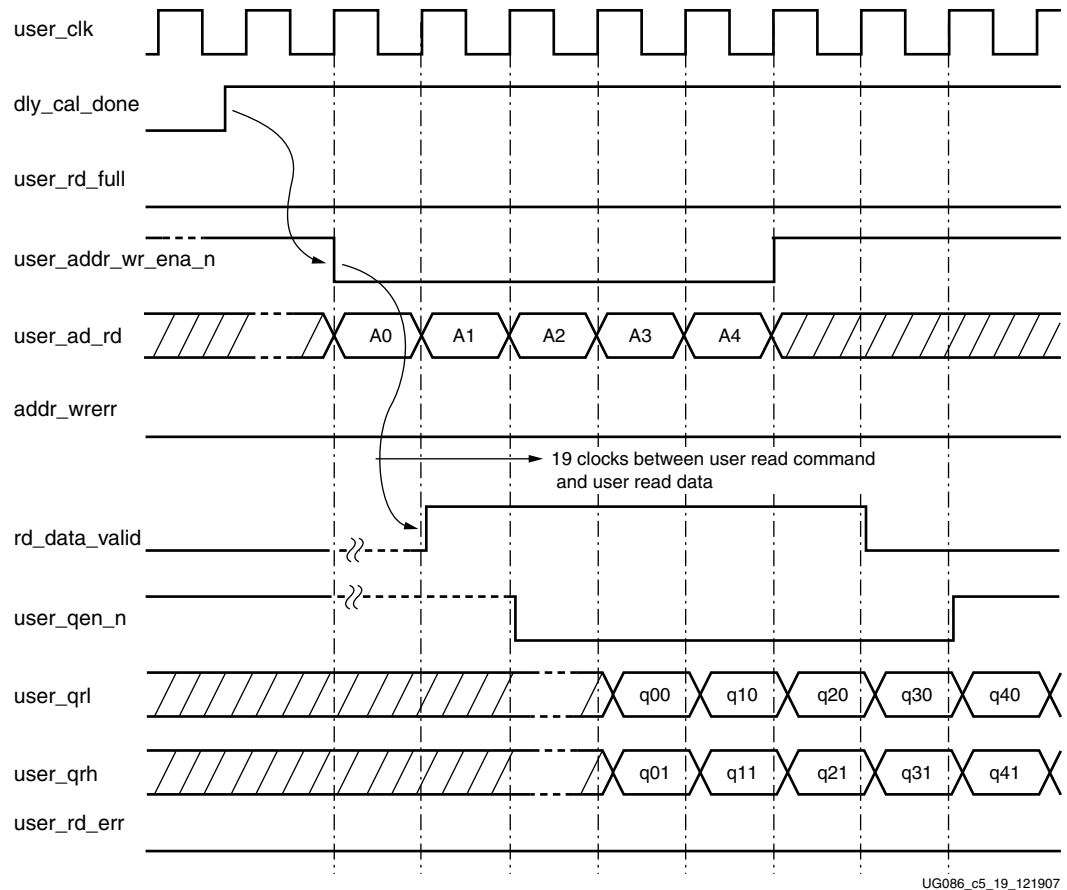
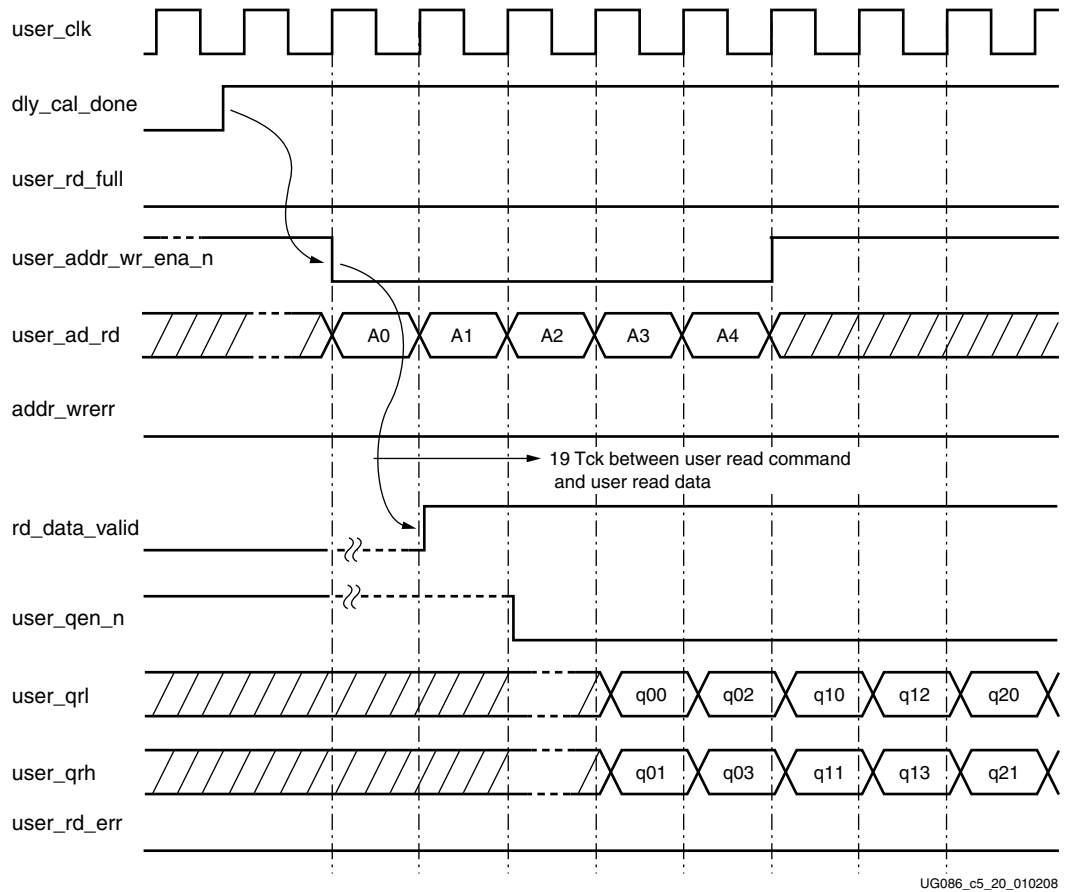


Figure 5-14: Read User Interface Timing Diagram for  $BL = 2$



UG086\_c5\_20\_010208

Figure 5-15: Read User Interface Timing Diagram for BL = 4

Table 5-6 shows the maximum read latency of the controller. Maximum latency occurs when the read command is given to an empty FIFO.

Table 5-6: Maximum Read Latency

Parameter	Number of Clocks	Description
User command to address FIFO empty flag	6 (2 + 4)	Two clocks for the two-stage pipeline before the FIFO input. An empty FIFO takes four clocks to deassert the empty status signal after the FIFO is written with the first data in FWFT.
Command from controller state machine to DDR memory	3	Decoding and passing the command to DDR memory.
DDR command to FIFO input data	4	Two clocks for DDRII memory latency, two clocks for calibration delay.
FIFO input to FIFO output	6	Pipelines the write enable six clock cycles (two-stage pipeline at the FIFO and one reg for calibration, and four clocks for deassertion of read data fifo empty).
<b>Total Latency</b>	<b>19</b>	<b>Total latency from read command issued to Address FIFO, to data input to user interface.</b>



Table 5-7 shows the list of signals for a DDRII SRAM design allocated in a group from bank selection check boxes in MIG.

Table 5-7: DDRII Signal Allocations

Bank Selected	Signals Allocated in the Group
Address	Memory address and memory control
Data	Memory data and memory byte read/write
System Control	System reset from the user interface and status signals
System_Clock	System clocks from the user interface

When the Address box is checked in a bank, the address, DDR\_LD\_N, DDR\_RW\_N, DDR\_DLL\_OFF\_n bits are assigned to that particular bank.

When the Data box is checked in a particular bank, the memory data, the memory byte write, the memory read clocks, the memory write clocks, and the memory input clock for the output data are assigned to that particular bank.

When the System Control box is checked in a bank, the SYS\_RST\_N, COMPARE\_ERROR, and DLY\_CAL\_DONE bits are assigned to that particular bank.

When the System\_Clock box is checked in a bank, the REFCLK\_P, REFCLK\_N, DLY\_CLK\_200\_P, and DLY\_CLK\_200\_N bits are assigned to that particular bank.

For special cases, such as without a testbench and without a DCM, the corresponding input and output ports are not assigned to any pins of the FPGA in the design UCF because the user can connect these ports to the FPGA pins or can connect to some logic internal to the same FPGA.

## Supported Devices

The design generated out of MIG is independent of the memory package, hence the package part of the memory component is replaced with X, where X indicates a don't care condition. Table 5-8 shows the list of components supported by MIG.

Table 5-8: Supported Devices for DDRII SRAM

Virtex-4 FPGAs (Verilog and VHDL)		
Components	Make	Configuration
CY7C1319BV18-250BZC	Cypress	x18
CY7C1318BV18-250BZC	Cypress	x18
CY7C1320BV18-200BZC	Cypress	x36
CY7C1320BV18-250BZC	Cypress	x36
CY7C1321AV18-250BZC	Cypress	x36
CY7C1321BV18-250BZC	Cypress	x36
CY7C1419AV18-250BZC	Cypress	x18
CY7C1420AV18-250BZC	Cypress	x36
CY7C1421AV18-250BZC	Cypress	x36

Table 5-8: Supported Devices for DDRII SRAM (Continued)

Virtex-4 FPGAs (Verilog and VHDL)		
Components	Make	Configuration
CY7C1427AV18-250BZC	Cypress	x9
CY7C1428AV18-250BZC	Cypress	x9
CY7C1518V18-250BZC	Cypress	x18
CY7C1520V18-250BZC	Cypress	x36
CY7C1916BV18-250BZC	Cypress	x9
CY7C1917BV18-250BZC	Cypress	x9
K7I161882B-FC25	Samsung	x18
K7I161884B-FC25	Samsung	x18
K7I163682B-FC25	Samsung	x36
K7I163684B-FC25	Samsung	x36
K7I321884C-FC25	Samsung	x18
K7I321884M-FC25	Samsung	x18
K7I323684C-FC25	Samsung	x36
K7I323684M-FC25	Samsung	x36
K7I641882M-FC25	Samsung	x18

## Simulating the DDRII SRAM Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains an external testbench, a memory model, a `.do` file, and an executable file to simulate the generated design. The Samsung memory model files are currently generated in Verilog only. For Cypress memory controller designs, a sample VHDL memory model file is provided. To learn more details about the files in the `sim` folder and to simulate the design, see `simulation_help.chm` in the `sim` folder.

## Hardware Tested Configurations

This design is not hardware verified.

# 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 generated with MIG. This design is based on XAPP710 [Ref 21].

## Feature Summary

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

### Supported Features

The RLDRAM II controller design supports the following:

- A maximum frequency of 250 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
- With and without a testbench
- With or without a DCM

## Design Frequency Range

Table 6-1: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-10		-11		-12	
	Min	Max	Min	Max	Min	Max
Component (SIO/CIO)	175	200	175	230	175	250

## Unsupported Features

The RLDRAM II controller design does not support:

- Commands in successive clocks with a burst length of 2. The controller processes these commands with one extra clock latency. For example, a READ or WRITE sequence of commands, BL = 2, Configuration = Any, CIO/SIO.

## Supported RLDRAM II Devices

The RLDRAM II controller design supports the RLDRAM II devices from Micron indicated in Table 6-2. MIG generates the designs for the list of components mentioned in Table 6-2 in both VHDL and Verilog. The design generated out of MIG is independent of memory package, hence the package part of the memory component is replaced with XX, where XX indicates any package.

Table 6-2: 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
MT49H32M9BM	Micron	CIO	x9	(-5), (-25), (-33)	9, 18, 36, 72
MT49H16M18FM	Micron	CIO	x18	(-5), (-25), (-33)	18, 36, 72
MT49H16M18BM	Micron	CIO	x18	(-5), (-25), (-33)	18, 36, 72
MT49H8M36FM	Micron	CIO	x36	(-5), (-25), (-33)	36, 72
MT49H8M36BM	Micron	CIO	x36	(-5), (-25), (-33)	36, 72
MT49H32M9CFM	Micron	SIO	x9	(-5), (-25), (-33)	9, 18, 36, 72
MT49H32M9CBM	Micron	SIO	x9	(-5), (-25), (-33)	9, 18, 36, 72
MT49H16M18CFM	Micron	SIO	x18	(-5), (-25), (-33)	18, 36, 72
MT49H16M18CBM	Micron	SIO	x18	(-5), (-25), (-33)	18, 36, 72

# Architecture

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

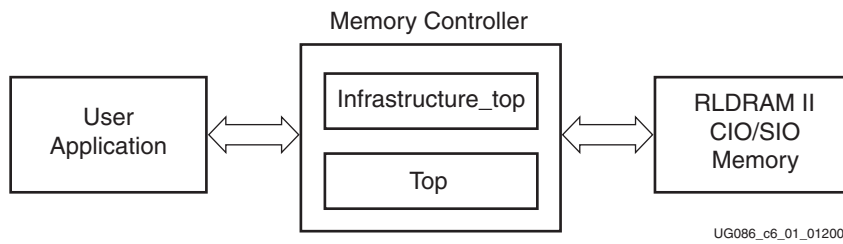
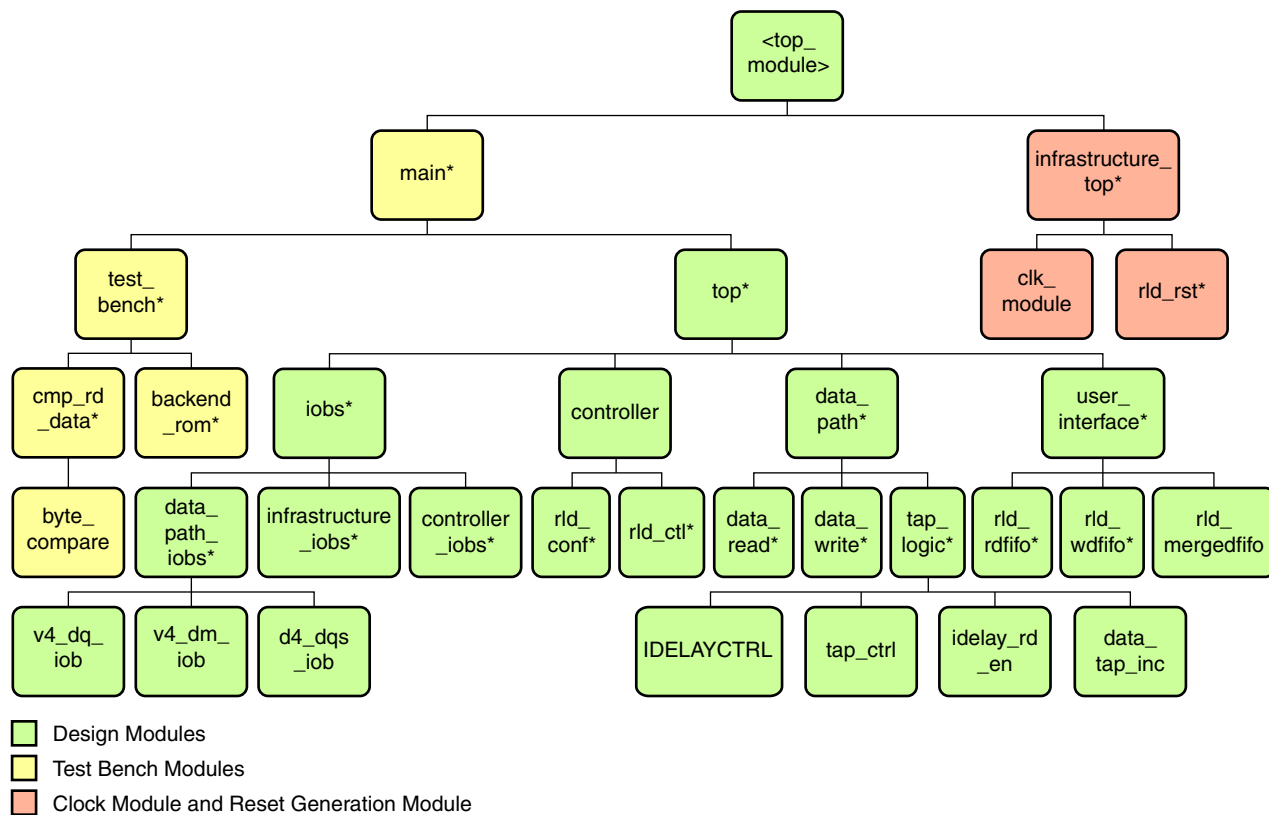


Figure 6-1: RLDRAM II Memory Controller Block Diagram

Figure 6-2 shows the hierarchical structure of the RLDRAM II design generated by MIG with a testbench and a DCM.



Note: A block with a \* has a parameter file included.

UG086\_c6\_02\_091307

Figure 6-2: RLDRAM II Memory Controller Hierarchy

The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks and reset generation modules

There is a parameter file generated with the design that has all the user input and design parameters selected from MIG.

MIG can generate four different RLDRAM II designs:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

When the testbench is not generated by MIG, the <top\_module> module has the user interface signals for designs without a testbench. The list of user interface signals is provided in [Table 6-10](#).

Design clocks and resets are generated in the `infrastructure_top` module, which comprises `clk_module` and `rld_rst` modules. The DCM clock is instantiated in the `clk_module` module for designs with a DCM. The differential design clock is an input to this module, which generates the system clocks. A user reset is input to the `rld_rst` module, which generates the system resets. A 200 MHz differential clock for the `IDELAYCTRL` module is derived from 200 MHz differential clocks. This clock is present in the top-level module.

The `clk_module` is not instantiated in the `infrastructure_top` module if the “DCM” option is not checked in MIG. So, the system operates on the user-provided clocks. The system reset is generated in the `rld_rst` module using the `DCM_LOCK` signal and the ready signal of the `idelay` control element.

Figure 6-3 shows a block diagram representation of an RLD2 II design with a DCM and a testbench. The design inputs are the system clocks and the user reset. `sysReset_n` is the system reset signal. All design resets are generated using the `DCM_LOCKED` signal, the `sysReset_n` signal, and the `idelay_ctrl_rdy` signal of the `IDELAYCTRL` element. The `PASS_FAIL` output signal indicates whether the design passes or fails. The `init_done` signal indicates the completion of initialization and calibration of the design. Required clocks and reset signals for the design are generated from the `clk_module` and the `rld_rst` modules, respectively. `clk_module` instantiates the DCM primitive. The `Infrastructure_top` module instantiates the `clk_module` and the `rld_rst` modules.

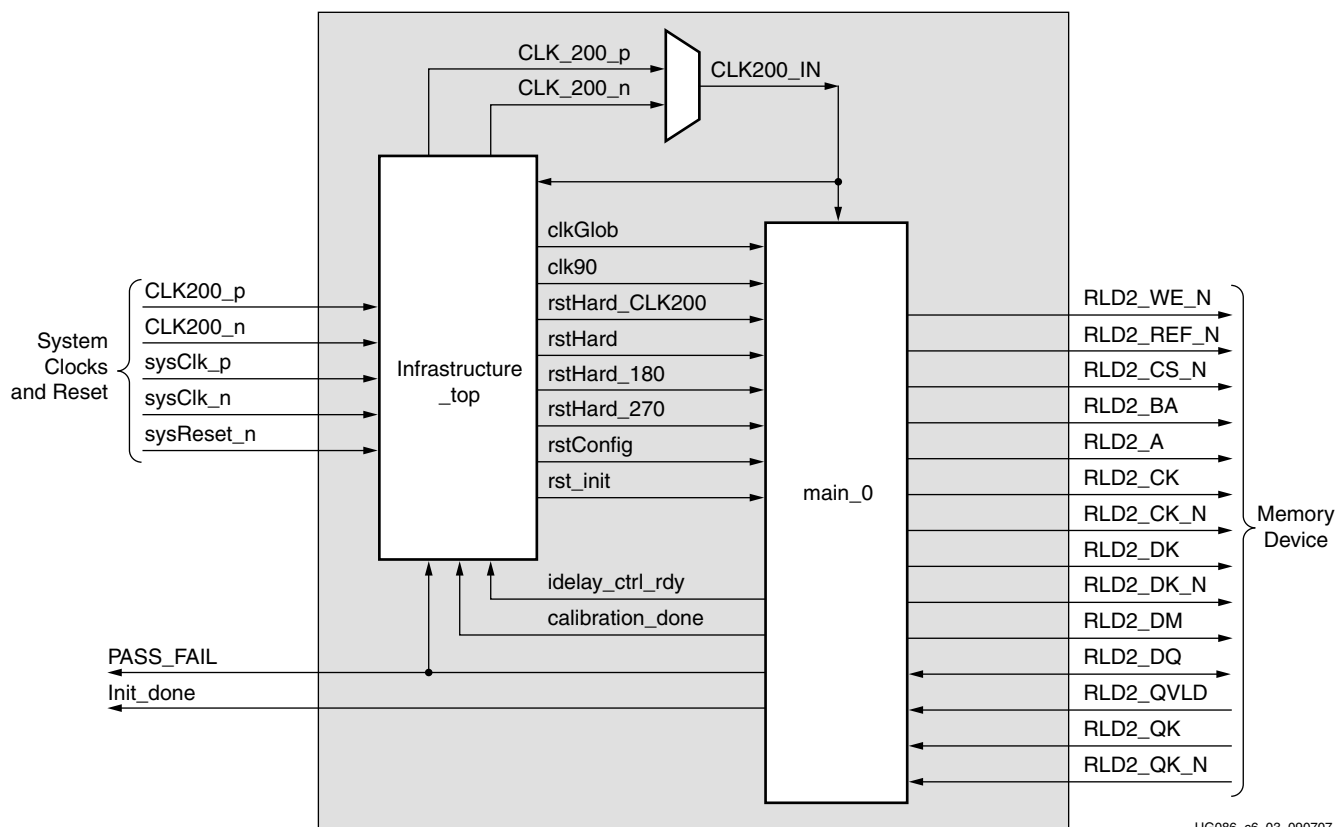
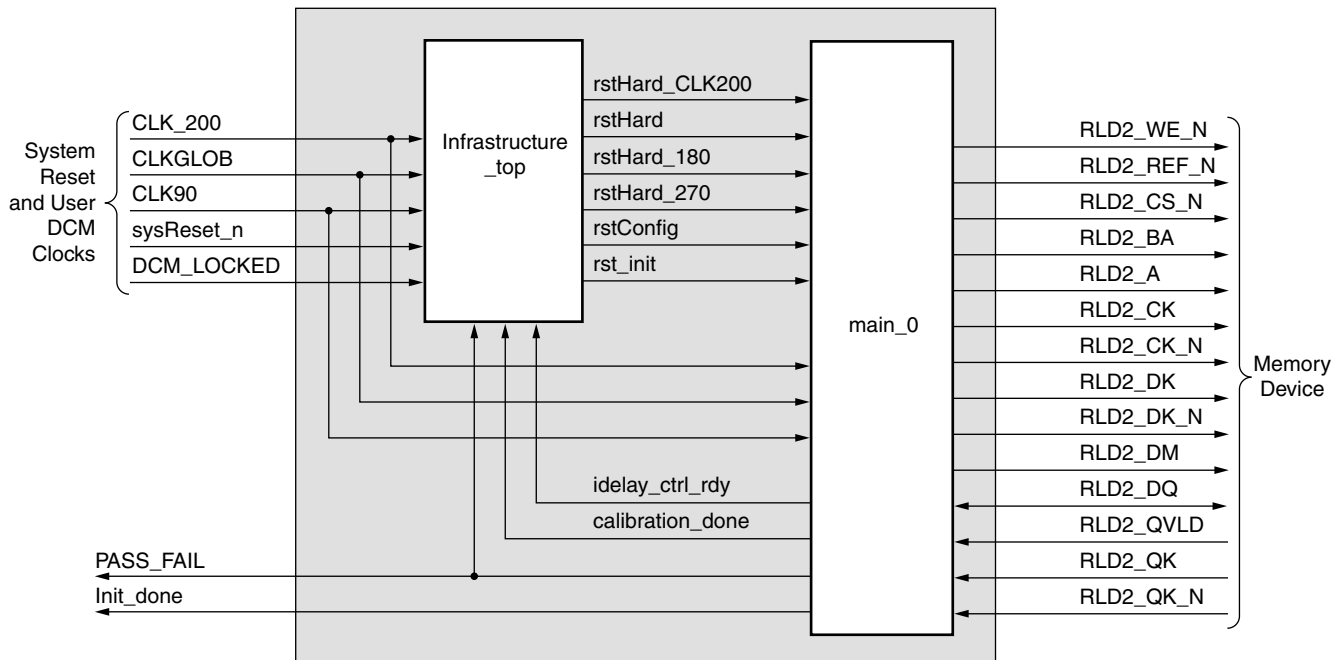


Figure 6-3: Top-Level Block Diagram of the RLD2 II Design with a DCM and a Testbench

Figure 6-4 shows a block diagram representation of the top-level RLD2 II module without a DCM but with a testbench. Design inputs are the user clocks and the user reset. sysReset\_n is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the sysReset\_n signal, and the idelay\_ctrl\_rdy signal of the IDELAYCTRL element. The design uses the user input clocks. These clocks should be single-ended. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFPGs. The PASS\_FAIL output signal indicates whether the design passes or fails. The init\_done signal indicates the completion of initialization and calibration of the design.

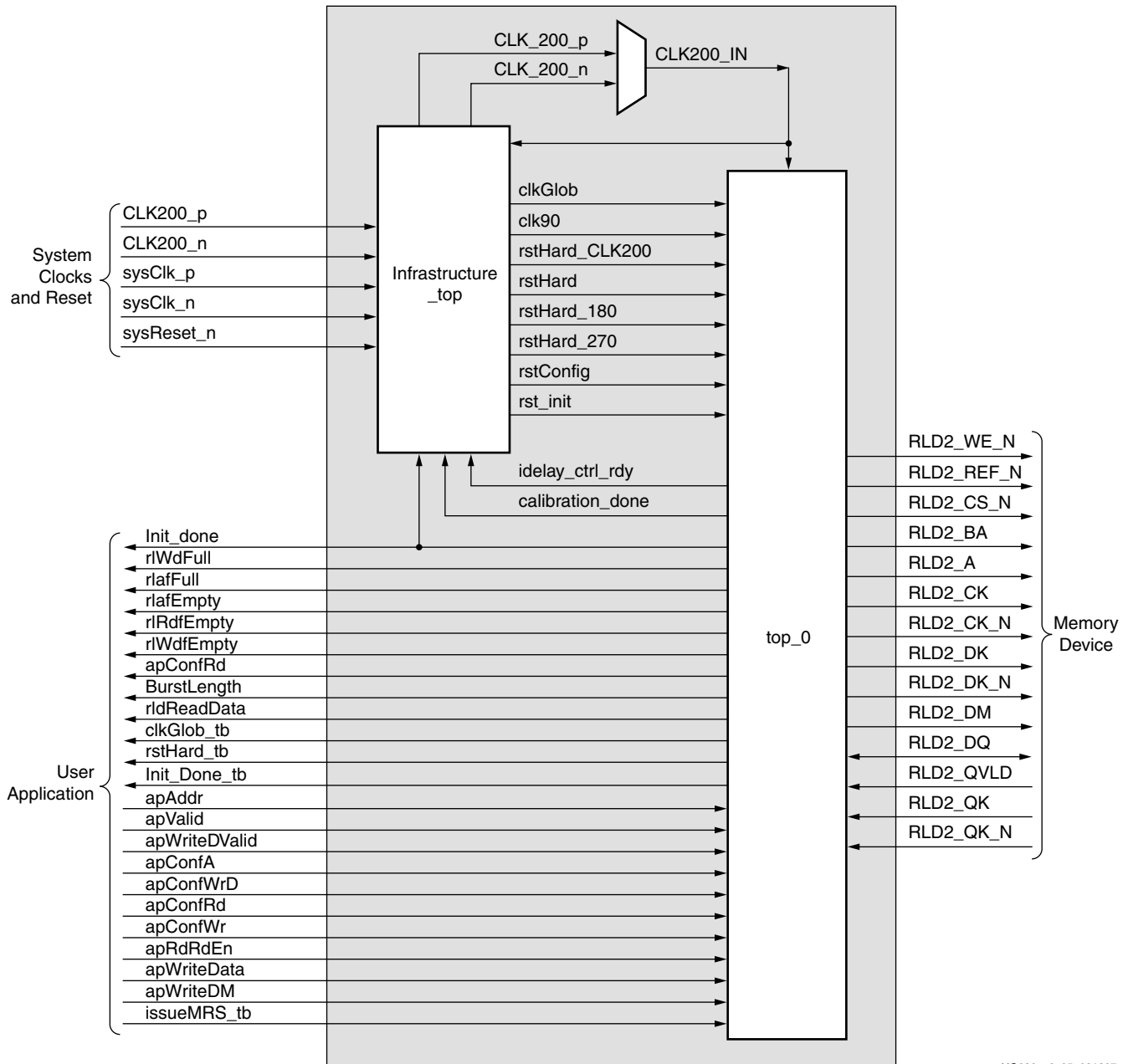


UG086\_c6\_04\_090707

Figure 6-4: Top-Level Block Diagram of the RLD2 II Design without a DCM but with a Testbench



Figure 6-5 shows a block diagram representation of the top-level RLD2RAM II module with a DCM but without a testbench. Design inputs are the system clocks and reset. sysReset\_n is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the sysReset\_n signal, and the idelay\_ctrl\_rdy signal of the IDELAYCTRL element. User must drive the user application signals. The design provides the clkGlob\_tb and rstHard\_tb signals to the user to synchronize the user application signals with the design. The required clocks and reset signals for the design are generated from the clk\_module and the rld\_rst modules, respectively. clk\_module instantiates the DCM primitive. The infrastructure\_top module instantiates the clk\_module and rld\_rst modules. The Init\_done signal indicates the completion of initialization and calibration of the design.



UG086\_c6\_05\_031207

Figure 6-5: Top-Level Block Diagram of the RLD2RAM II Design with a DCM but without a Testbench

Figure 6-6 shows a block diagram representation of the top-level RLD2RAM II module without a DCM or a testbench. Design inputs are the user clocks and the user reset. sysReset\_n is the system reset signal. All design resets are generated using the DCM\_LOCKED signal, the sysReset\_n signal, and the idelay\_ctrl\_rdy signal of the IDELAYCTRL. The design uses the user input clocks, which should be single-ended. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFPGs. User must drive the user application signals. The design provides the clkGlob\_tb and rstHard\_tb signals to the user to synchronize the user application signals with the design. The Init\_done signal indicates the completion of initialization and calibration of the design.

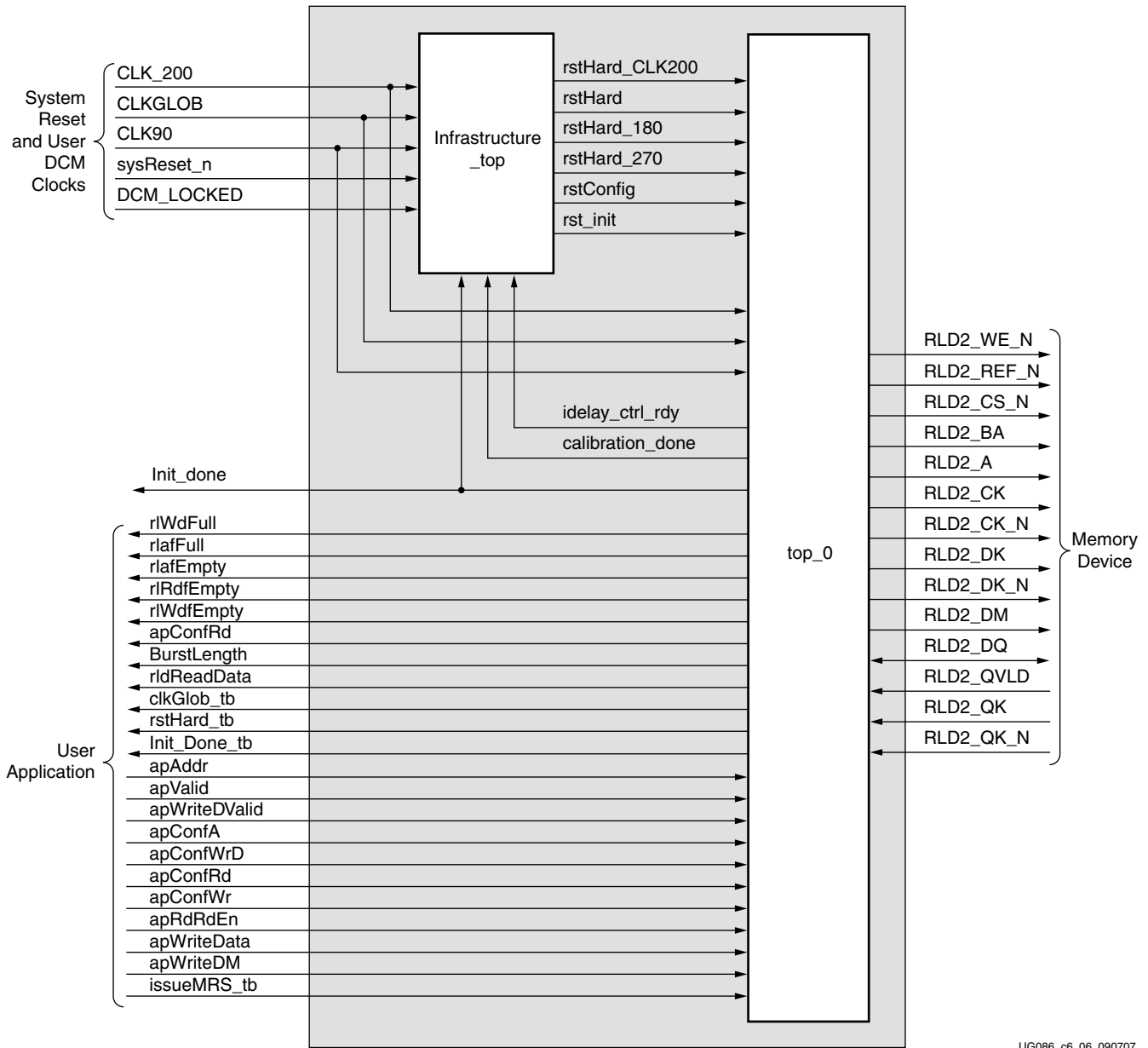


Figure 6-6: Top-Level Block Diagram of the RLD2RAM II Design without a DCM or a Testbench

UG086\_c6\_06\_090707

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 testbench to generate all the RLDRAM commands. The built-in testbench 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 testbench. MIG generates designs with and without a testbench. 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, control, and data signals, and 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 6-3](#) provides the address mapping between the controller and the RLDRAM II devices for the multiplexed address mode.

**Table 6-3: 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 17] for details on this technique.

Calibration is done in two stages:

1. In the first stage of calibration, QK is center-aligned with respect to the FPGA clock. QK is a free-running clock from RLDRAM II. The DQ data is edge-aligned with the QK read strobe, and the QVLD read data valid signal is edge-aligned with the QK read strobe. The first and second edges of the QK strobe are detected using the FPGA clock to determine the center of the QK window.

Once the QK window is center-aligned with the FPGA clock, the same amount of delay (tap counts) is applied to the DQ through the IDELAY element, so that the DQ window is center-aligned with the FPGA clock. Signal `qk_tap_sel_done` in the `tap_logic` module indicates the status of the first stage calibration. When `qk_tap_sel_done` is asserted High, it indicates the completion of first stage calibration. After the first stage calibration is complete, the second stage calibration starts.

2. In the second stage of calibration, the write-enable signal for the read data FIFO is determined in order to store the read data from memory into the Read Data FIFO. QVLD from RLDRAM II is delayed such that it exactly aligns with the delayed DQ window. This delayed QVLD signal is used as the write-enable signal for the Read Data FIFO.

The `sel_done` port in the `data_path` module indicates the status of the second stage calibration. When `sel_done` is asserted High, it indicates the completion of second stage calibration, which implies the completion of the whole initialization and calibration process. After the initialization and calibration is done (i.e., the `init_done` signal in `design_top` is asserted High), the controller can start issuing user commands to the memory.

When calibration is complete, the `calibration_done` signal is asserted High.

## 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.

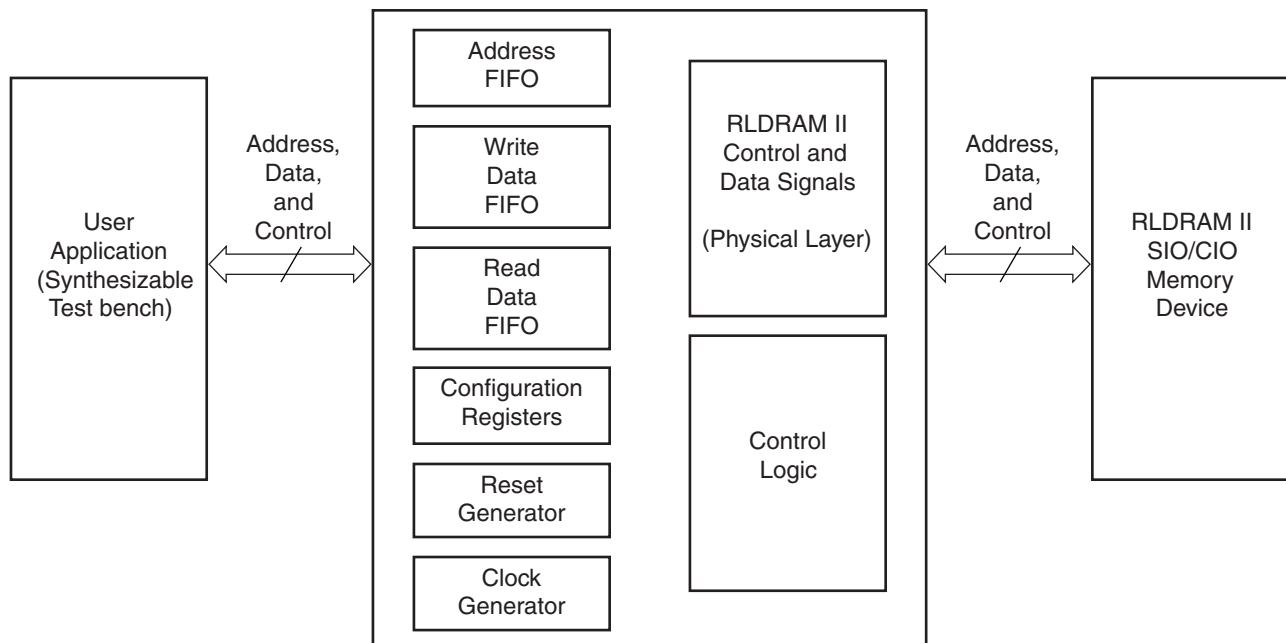
After all power supply and reference voltages are stable and the master clock (RLD\_CK and RLD\_CK\_N) is stable, the RLDRAM II device requires a 200  $\mu$ s (minimum) delay prior to applying an executable command. After the 200  $\mu$ s (minimum) delay has passed, three MODE REGISTER SET (MRS) commands 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 6-7 shows a detailed block diagram of the RLDRAM II memory controller. The major blocks of the controller are described following the figure.



UG086\_c6\_09\_012507

Figure 6-7: 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 6-4 defines the configuration of the 26 bits.

Table 6-4: Address FIFO Bit Configuration

Bit Configuration	Description
25	User Refresh
24	Read/ $\overline{\text{Write}}$
[23:3]	Memory Address bits A[20:0]
[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 synchronous 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 6-5 defines the FIFO configuration for 36-bit data width using x36 memory components.

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

Table 6-6: 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 6-7 shows the internal configuration register read and write details from the user interface. A 4-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 6-7: 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 with 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 17] for more details.

## RLDRAM II Interface Signals

Table 6-8 and Table 6-9 define the RLDRAM II system interface signals with and without a DCM, respectively.

Table 6-8: RLDRAM II System Interface Signals (with a DCM)

Signal Name	Direction	Description
sysClk_p, sysClk_n	Input	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. When the Without DCM option is selected, this clock pair is not present.
CLK200_p, CLK200_n	Input	Differential clock used in the idelay_ctrl logic.
sysReset_n	Input	Active-Low reset to the RLDRAM II controller.
PASS_FAIL[2:0]	Output	This signal bus indicates the status the comparison between the read data compared with the corresponding write data. 001: INITIALIZATION STATE 010: PASS 100: FAIL
Init_done	Output	This signal is asserted when the design initialization and calibration is complete.

Table 6-9: RLDRAM II System Interface Signals (without a DCM)

Signal Name	Direction	Description
CLKGLOB	Input	Input clock
CLK90	Input	Input clock with a 90° phase difference
CLK_200	Input	200 MHz clock for Idelayctrl primitives
DCM_LOCKED	Input	This active-High signal indicates whether the user DCM is locked or not
sysReset_n	Input	Active-Low reset to the RLDRAM II controller
PASS_FAIL[2:0]	Output	This signal bus indicates the status the comparison between the read data compared with the corresponding write data. 001: INITIALIZATION STATE 010: PASS 100: FAIL
Init_done	Output	This signal is asserted when the design initialization and calibration is complete



Table 6-10 describes the RLDRAM II user interface signals.

Table 6-10: RLDRAM II User Interface Signals (without a Testbench)

Signal Name	Direction	Description
rlWdfFull	Output	Almost full status signal for the Write Data FIFO. When this signal is asserted, the user can write three more data words into the FIFO.
rlafFull	Output	Almost full status signal for the Address FIFO. When this signal is asserted, the user can write two more data words into the FIFO.
rlafEmpty	Output	Empty status signal for the Address FIFO
rlRdfEmpty	Output	Empty status signal for the Read Data FIFO
rlWdfEmpty	Output	Empty status signal for the Write Data FIFO
apAddr[25:0]	Input	Address FIFO data input. This bus consists of the user-defined bank address, the address, the WRITE/READ command, and the user-defined REFRESH command.
apValid	Input	Address FIFO write-enable signal
apWriteDValid	Input	Write Data FIFO write-enable signal
apConfA[3:0]	Input	Address bus for the Configuration registers
apConfWrD[7:0]	Input	Write data for the Configuration registers
apConfRd	Input	Read enable for the Configuration registers
apConfRdD[7:0]	Output	Read data for the Configuration registers
apConfWr	Input	Write data valid for the Configuration registers
apRdfRdEn	Input	Read enable for the Read Data FIFO
BurstLength[1:0]	Output	Indicates the number of bursts that can be written to or read from the memory: 00: Burst length = 2 01: Burst length = 4 10: Burst length = 8
rldReadData[(2*n)-1:0]	Output	Read data from the memory, where $n$ is the data width of the design. This read data is stored in the Read Data FIFOs and can be read from the FIFOs depending upon the status of the FIFOs.
apWriteData[(2*n)-1:0]	Input	Write data to be written into the memory, where $n$ is the data width of the design. This data is stored in the Write Data FIFO and is written into the memory depending upon the controller status (write command).
apWriteDM[m-1:0]	Input	Data mask of the write data, where $m$ is the number of data mask bits associated with the write data width.
clkGlob_tb	Output	clkGlob clock input. All the corresponding signals must be synchronized with clkGlob_tb.
rstHard_tb	Output	Active-Low system reset for the user interface, synchronous with clkGlob_tb.
Init_Done_tb	Output	When asserted, this signal indicates that memory initialization is complete.
issueMRS_tb	Input	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.)

**Notes:**

1. All user interface signal names are prepended with a controller number, for example, cntrl0\_apWriteData. RLDRAM II devices currently support only one controller.

Table 6-11 describes the RLD2RAM II memory interface signals.

Table 6-11: RLD2RAM II Memory Interface Signals

Signal Name	Direction	Description
RLD2_DQ (CIO)	Input/ Output	Data input/outputs. During READ commands, the data is captured using the FPGA clock. During WRITE commands, the data is sampled on both edges of DK.
RLD2_D (SIO)	Output	Write data
RLD2_Q (SIO)	Input	Read data
RLD2_A	Output	Row and column addresses for READ and WRITE operations. During a MODE REGISTER SET command, the address inputs define the register settings.
RLD2_BA	Output	These bank addresses select the internal bank to which to apply commands.
RLD2_WE_N	Output	Write-enable command
RLD2_REF_N	Output	REFRESH command
RLD2_CS_N	Output	Chip-select command
RLD2_DM	Output	Data mask signals for the write data
RLD2_QVLD	Input	Data valid signals transmitted by the RLD2RAM II devices. They indicate valid read data.
RLD2_QK, RLD2_QK_N	Input	Differential read data clocks. These clocks are transmitted by the RLD2RAM II devices and are edge-aligned with the read data.
RLD2_DK, RLD2_DK_N	Output	Differential write data clocks.
RLD2_CK, RLD2_CK_N	Output	Master differential clocks for addresses and commands.

## User Command Interface

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

### User Interface Accesses

The user backend logic communicates with the memory controller through a FIFO-based user interface. This interface consists of three related buses:

- A Command/Address FIFO bus accepts write/read commands as well as the corresponding memory address from the user
- A Write Data FIFO bus accepts the corresponding write data when the user issues a write command on the command/address bus
- A Read bus on which the corresponding read data for an issued read command is returned

The user interface has the following timing and signaling restrictions:

- Commands and write data cannot be written by the user until calibration is complete (as indicated by `init_done`). In addition, the `apvalid` and `app_wdf_wren` interface signals need to be held Low until calibration is complete.

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

## Write Interface

Figure 6-8 shows the user interface block diagram for write operations.

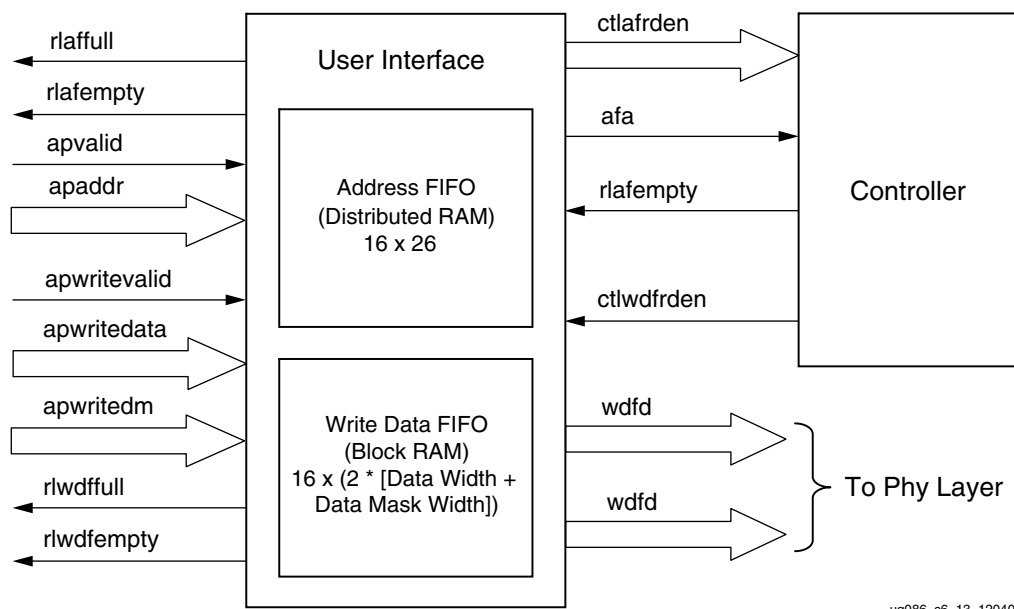


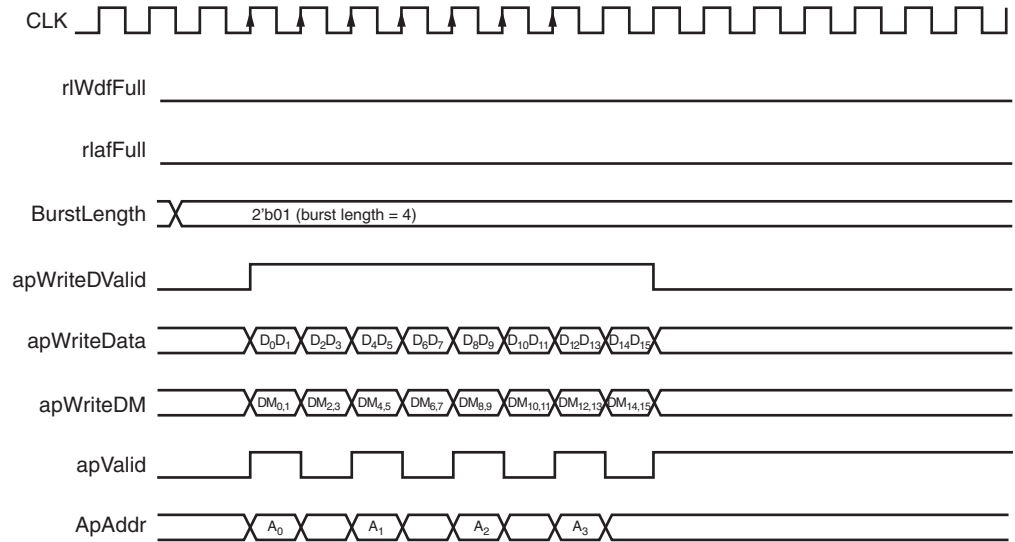
Figure 6-8: User Interface Block Diagram for Write Operations

The following steps describe the architecture of the Address and Write Data FIFOs and show how to perform a write burst operation to RLDRAM II from the user interface.

- The user interface consists of an Address FIFO and a Write Data FIFO. These FIFOs are constructed using the CORE Generator™ FIFO generator module. Address FIFO is a distributed RAM with 16 x 26 configuration. Data FIFO is a block RAM, with a depth of 16 locations and width equal to two times the Data width and Data Mask width together.
- The Common Address FIFO is used for both write and read commands, and comprises a command part and an address part. Command bits discriminate between write and read commands.
- User interface data width apwritedata is twice that of the memory data width. For every memory component there is a mask bit. For 9-bit memory width, the user interface is 20 bits consisting of rising-edge data, falling-edge data, rising-edge mask bit, and falling-edge mask bit.
- For a 9-bit memory component with 72-bit data, the user interface data width apwritedata is 144 bits, and the mask data apwritedm is 8 bits.
- The user can initiate a write to memory by writing to the Address FIFO and the Write Data FIFO when the FIFO Full flags are deasserted and after the init\_done signal is

asserted. Status signal rlafull is asserted when Address FIFO is full, and similarly rlwdfull is asserted when Write Data FIFO is full.

6. Both the Address FIFO and Write Data FIFO Full flags are deasserted with power-on.
7. The user should assert the Address FIFO write-enable signal apvalid along with address apaddr to store the write address and write command into the Address FIFO.
8. The user should assert the Data FIFO write-enable signal apwritedata along with write data apwritedata and mask data apwritedm to store the write data and mask data into the Write Data FIFO. The user should provide both rise and fall data together for each write to the Data FIFO.
9. The controller reads the Address FIFO by issuing the ctlafrden signal. The controller reads the Write Data FIFO by issuing the ctldwfrden signal after the Address FIFO is read. It decodes the command part after the Address FIFO is read.



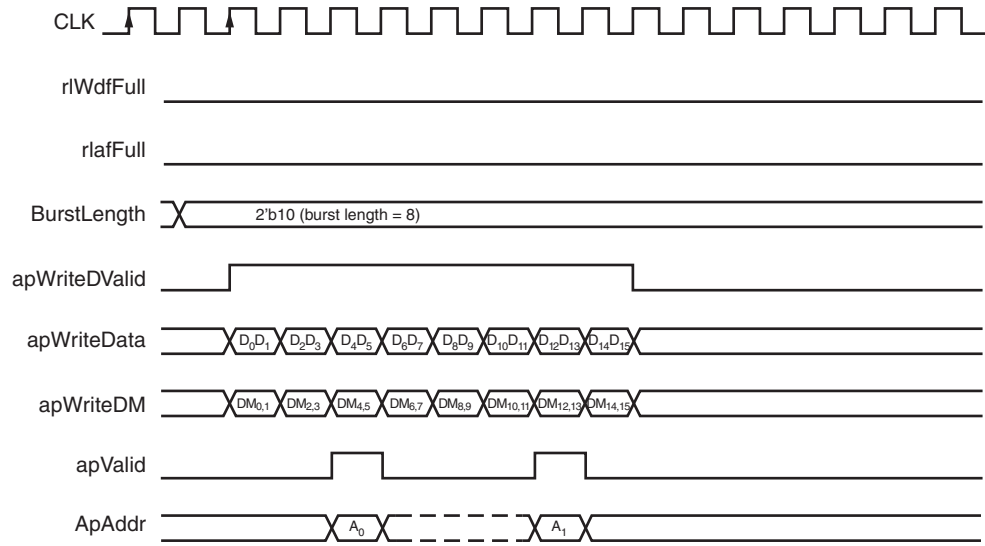
UG086\_c6\_10\_012807

Figure 6-9: RLDRAM II Write Burst Timing Diagram (BL = 4), Four Bursts

10. The write command timing diagram in Figure 6-9 is derived from the MIG-generated test bench. As shown (burst length of 4), each write to the Address FIFO must be coupled with two writes to the Data FIFO. Similarly, for a burst length of 8, every write to the Address FIFO must be coupled with four writes to the Data FIFO. Failure to follow this rule can cause unpredictable behavior.

**Note:** The user can start filling the Write Data FIFO two clocks after the Address FIFO is written, because there is a two-clock latency between the command fetch and reading the Data FIFO. Using the terms shown in Figure 6-9, therefore, the user can assert the A0 address two clocks before D0D1.

11. The write command timing diagram in Figure 6-10, page 233 is derived from the MIG-generated test bench. As shown (burst length of 8), each write to the Address FIFO must be coupled with four writes to the Data FIFO. Because the controller first reads the address and command together, the address need not coincide with the last data. After the command is analyzed (nearly two clocks later for a worst-case timing scenario), the controller sequentially reads the data in four clocks. Thus, there are six clocks from the time the address is read to the time the last data is read.

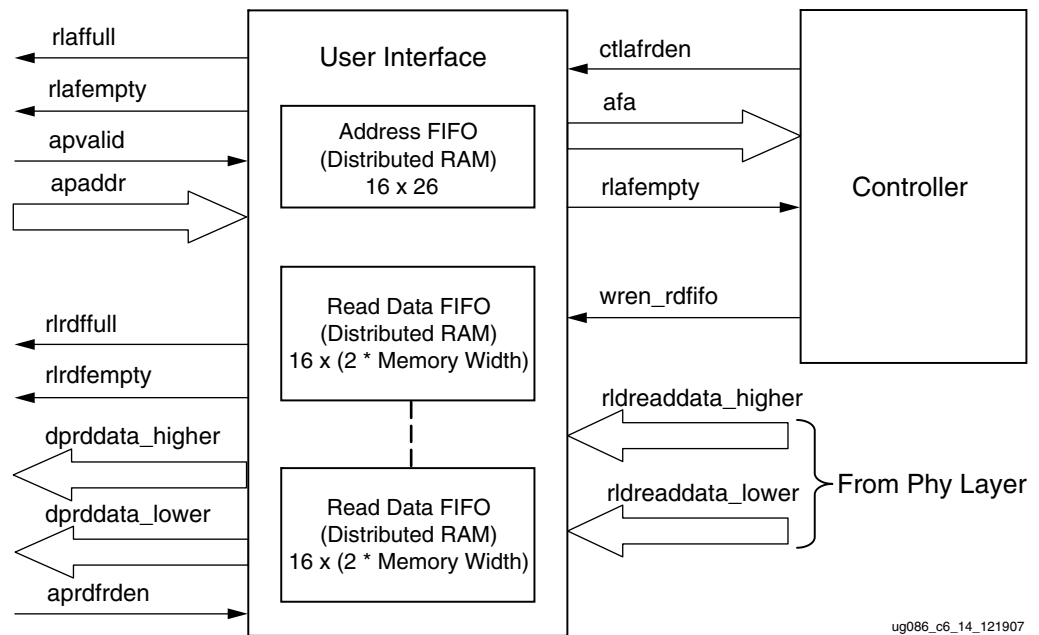


UG086\_c6\_11\_012807

Figure 6-10: RLDRAM II Write Burst Timing Diagram (BL = 8), Two Bursts

## Read Interface

Figure 6-11 shows a block diagram of the read interface.



ug086\_c6\_14\_121907

Figure 6-11: User Interface Block Diagram for Read Operations

The following steps describe the architecture of the Read Data FIFOs and show how to perform a burst read operation from RLDRAM II from the user interface.

1. The read user interface consists of an Address FIFO and a Read Data FIFO. The Address FIFO is common to both read and write operations. The Read Data FIFOs are constructed using the CORE Generator FIFO generator module. The Read Data FIFO is a Distributed RAM with depth of 16 locations and width equal to two times the

memory device width, consisting of rising-edge data and falling-edge data. For example, for a 9-bit memory component, the Read Data FIFO configuration is 16 x 18. MIG instantiates a number of Read Data FIFO modules depending on the QK signal width of the design. For example, for 9-bit memory component and 72-bit data width designs, MIG instantiates a total of nine Read Data FIFO modules.

2. The user can initiate a read to memory by writing to the Address FIFO when the FIFO Full flag rlafull is deasserted and after init\_done is asserted.
3. To write the read address and read command into the Address FIFO, the user should issue the Address FIFO write-enable signal apvalid along with read address apaddr.
4. The controller reads the Address FIFO containing the address and command. After decoding the command, the controller generates the appropriate control signals to memory.
5. Prior to the actual read and write commands, the design calibrates the latency (number of clock cycles) from the time the read command is issued to the time data is received. Using this pre-calibrated delay information, the controller generates the write-enable signals to the Read Data FIFOs.
6. The rlrdfempty signal is deasserted when data is available in the Read Data FIFOs.
7. The user can read the read data from the Read Data FIFOs by asserting aprdfrden to High.

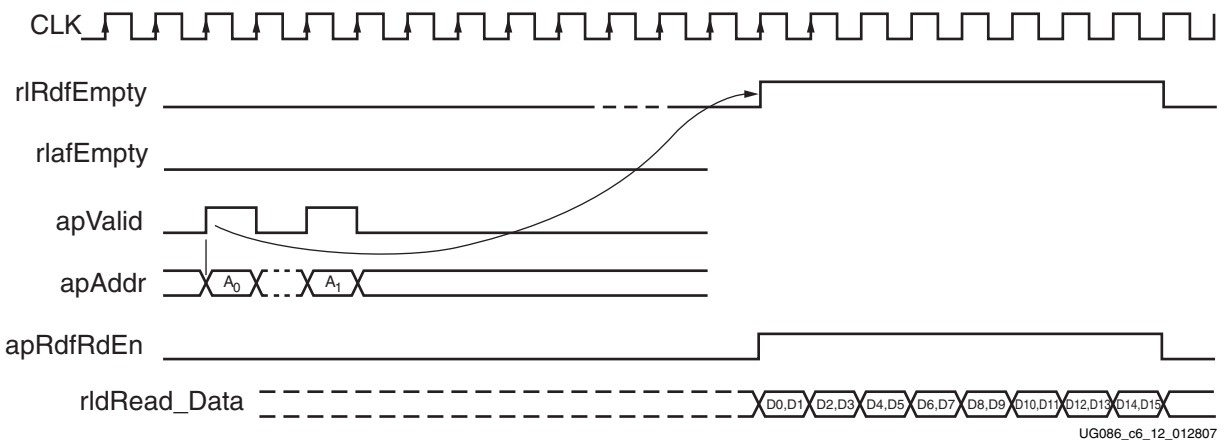


Figure 6-12: RLDRAM II Read Burst Timing Diagram (BL = 8), Two Bursts

8. Figure 6-12 shows the user interface timing diagram for a burst length of 8. The read latency is calculated from the point when the read command is given by the user to the point when the rlrdfempty signal is deasserted. The minimum latency in this case is 21 clocks. Where no auto-refresh request is pending, the user commands are issued after initialization is completed, and the first command issued is a Read command. The controller executes the commands only after initialization is done, as indicated by the init\_done signal.
9. After the address and command are loaded into the Address FIFO, it takes 21 clock cycles minimum for the controller to deassert the rlrdfempty signal.
10. Read data is available only when the rlrdfempty signal is deasserted. The user can access the read data by asserting the aprdfrden signal, a read enable signal to the Read Data FIFOs, to High.

**Note:** The RLDRAM controller does not check the status of the Read Data FIFO, and can issue read commands even when the Read Data FIFO is full. The user must make this determination and ensure that read commands are not issued by the controller when the Read Data FIFO is full.

Table 6-12: Read Command Latency

Parameter	Number of Clocks	Description
User command to deassertion of the Address FIFO empty flag	1	When the read command is given to an empty FIFO, it takes one clock time to deassert the empty flag
Controller command reading and decoding time	3	The FIFO outputs the data one clock after the read command. Two clocks for decoding the command.
Command from the controller to the controller IOB's output	3	Three-stage pipeline
RLDRAM II command to read data latency (max)	8	RLDRAM II worst-case latency
Read data from the IOB to dq_iob	2	Two-stage pipeline from IOB to dq_iob
dq_iob output to Read Data FIFO input	2	Two-stage pipeline
Read Data FIFO input to Read Data FIFO output	2	One clock for deassertion of empty signal, and one clock for outputting the data
<b>Total Latency</b>	<b>21</b>	<b>Total of all latencies</b>

In general, read latency varies based on the following parameters:

- Configuration
- The number of commands already in the FIFO pipeline before the read command is issued
- Whether commands are interrupted when the periodic AUTO REFRESH command is issued
- Whether the user issues the commands before initialization is complete (if so, the latency cannot be determined)
- Board-level and chip-level propagation delays for both memory and FPGA

## 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.

MIG shows the check boxes listed in [Table 6-13](#) when a bank is selected for an RLDRAM II design.

Table 6-13: RLDRAM II Signal Allocation

Bank Selected by Check Box	Signals Allocated in the Group
Address	Memory address and memory control
Data (CIO)	Memory data, memory data mask, and memory clocks
Data_Write (SIO)	Memory write data, memory data mask, and memory write clocks
Data_Read (SIO)	Memory read data, memory QVLD, and memory read clocks
System Control	System reset from the user interface and status signals
System_Clock	System clocks from the user interface

When the Address box is checked in a particular bank, the bank address, the address, the WE\_N, the REF\_N, and the CS\_N bits are assigned to that particular bank.

When the Data box is checked in a particular bank for a CIO design, the memory data, the memory data mask, the memory data valid (QVLD), the memory read clock, the memory write clock, the memory address, and the command clock bits are assigned to that particular bank.

When the Data\_Write box is checked in a particular bank for an SIO design, the memory data write, the memory data mask, and the memory write clock bits are assigned to that particular bank.

When the Data\_Read box is checked in a particular bank for an SIO design, the memory data read, the memory data valid (QVLD), the memory read clock, the memory address, and the command clock bits are assigned to that particular bank.

When the System Control box is checked in a particular bank, the sysReset\_n, the PASS\_FAIL, and the Init\_done bits are assigned to that particular bank.

When the System\_Clock box is checked in a particular bank, the sysClk\_p, sysClk\_n, CLK200\_p, and CLK200\_n bits are assigned to that particular bank.

For special cases, such as without a testbench and without a DCM, the corresponding input and output ports are not assigned to any FPGA pins in the design UCF because the user can connect these ports to the FPGA pins or can connect to some logic internal to the same FPGA.

## Simulating the RLDRAM II Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains the external testbench, memory model, `.do` file, and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more details about the files in the `sim` folder and to simulate the design, see `simulation_help.chm` in the `sim` folder.



## Hardware Tested Configurations

The frequencies shown in [Table 6-14](#) were achieved on the Virtex-4 FPGA ML461 Memory Interfaces Development Board under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based on a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 72-bit wide interface.

**Table 6-14: Hardware Tested Configurations**

FPGA Device	XC4VLX25-FF668-11
Memory Component	MT49H16M18XX-25
Data Bus Options	CIO
Data Width	36
Configuration	1, 2, 3
Burst Length	2, 4, 8
Addressing Mode	Multiplexing and Non-Multiplexing Addressing mode
Frequency	120 MHz to 330 MHz
Flow Vendors	Synplicity and XST
Design Entry	VHDL and Verilog





## *Section III: Spartan-3/3E/3A/3AN/3A DSP FPGA to Memory Interfaces*

*Chapter 7, "Implementing DDR SDRAM Controllers"*

*Chapter 8, "Implementing DDR2 SDRAM Controllers"*



## Implementing DDR SDRAM Controllers

---

This chapter describes how to implement DDR SDRAM interfaces for Spartan™-3, Spartan-3E, Spartan-3A, Spartan-3AN, and Spartan-3A DSP FPGAs. The designs are based on XAPP768c [Ref 23].

### Feature Summary

The DDR SDRAM controller design supports the following:

- Burst lengths of two, four, and eight
- CAS latencies of 2, 2.5, and 3
- Sequential and interleaved burst types
- Auto refresh
- Spartan-3 FPGA maximum frequency:
  - ◆ 133 MHz with a -4 speed grade device
  - ◆ 166 MHz with a -5 speed grade device
- Spartan-3E FPGA maximum frequency:
  - ◆ 133 MHz with a -4 speed grade device
  - ◆ 166 MHz with a -5 speed grade device
- Spartan-3A, Spartan-3AN, and Spartan-3A DSP FPGA maximum frequency:
  - ◆ 133 MHz with a -4 speed grade device
  - ◆ 166 MHz with a -5 speed grade device
- Components, unbuffered DIMMs, registered DIMMs, and SODIMMs
- With and without a testbench
- With or without a DCM
- All Spartan-3, Spartan-3E, Spartan-3A, Spartan-3AN, and Spartan-3A DSP FPGAs
- Verilog and VHDL
- XST and Synplicity synthesis tools

## Design Frequency Ranges

Table 7-1: Design Frequency Range in MHz

FPGA Family	Memory	FPGA Speed Grade			
		-4		-5	
		Min	Max	Min	Max
Spartan-3	Component	77	133	77	166 <sup>(1)</sup>
	DIMM	77	133	77	133
Spartan-3A/3AN/3A DSP	Component	77	133	77	166
	DIMM	77	133	77	166
Spartan-3E	Component	77	133	77	166
	DIMM	(Not supported)			

**Notes:**

1. Spartan-3 devices support 133 MHz for data widths greater than 32 bits.

## 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.

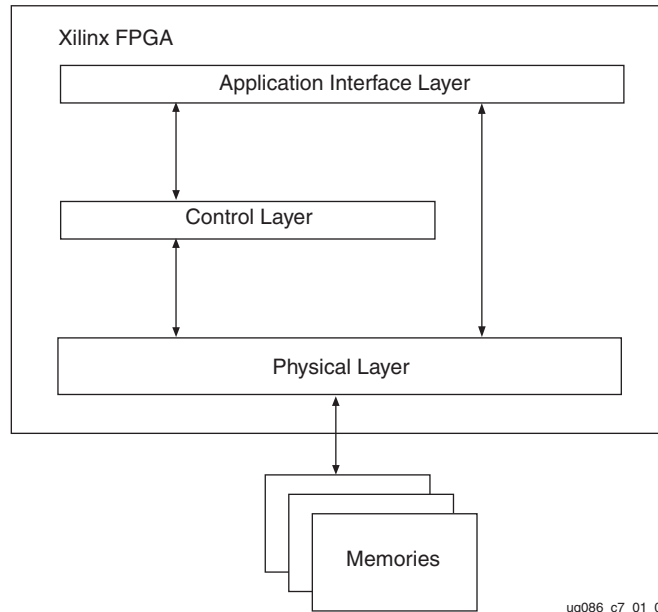
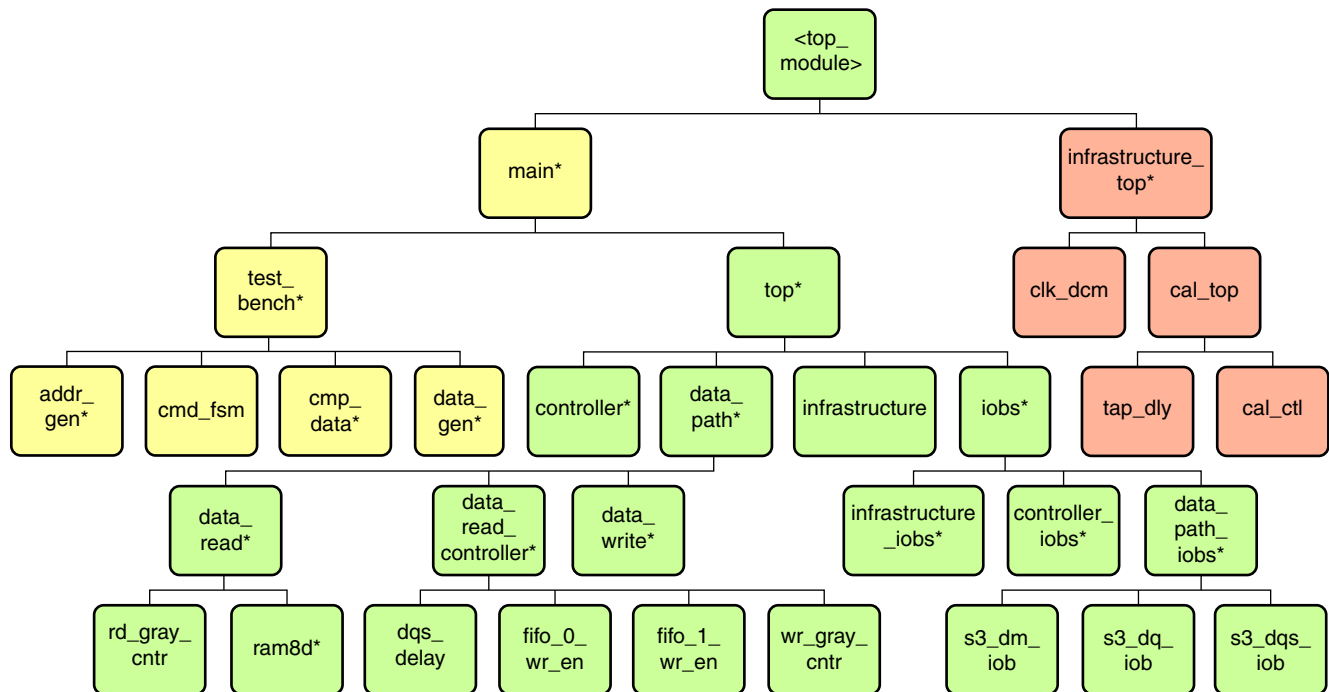


Figure 7-1: Modular Memory Interface Representation

## Hierarchy

Figure 7-2 shows the hierarchical structure of the DDR SDRAM design generated by MIG with a testbench and a DCM. In the figure, the physical and control layers are clearly separated. MIG generates the entire controller, as shown in this hierarchy, including the testbench. The user can replace the testbench with a design that makes use of the DDR SDRAM interface.



- Design Modules
- Test Bench Modules
- Clocks, Reset Generation, and Calibration Modules

Note: A block with a \* has a parameter file included.

UG086\_c7\_02\_010108

Figure 7-2: Hierarchical Structure of the DDR SDRAM Design with a Testbench

The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks, reset generation, and calibration modules

There is a parameter file generated with the design that has all the user input and design parameters selected from MIG.

MIG can generate four different DDR SDRAM designs:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

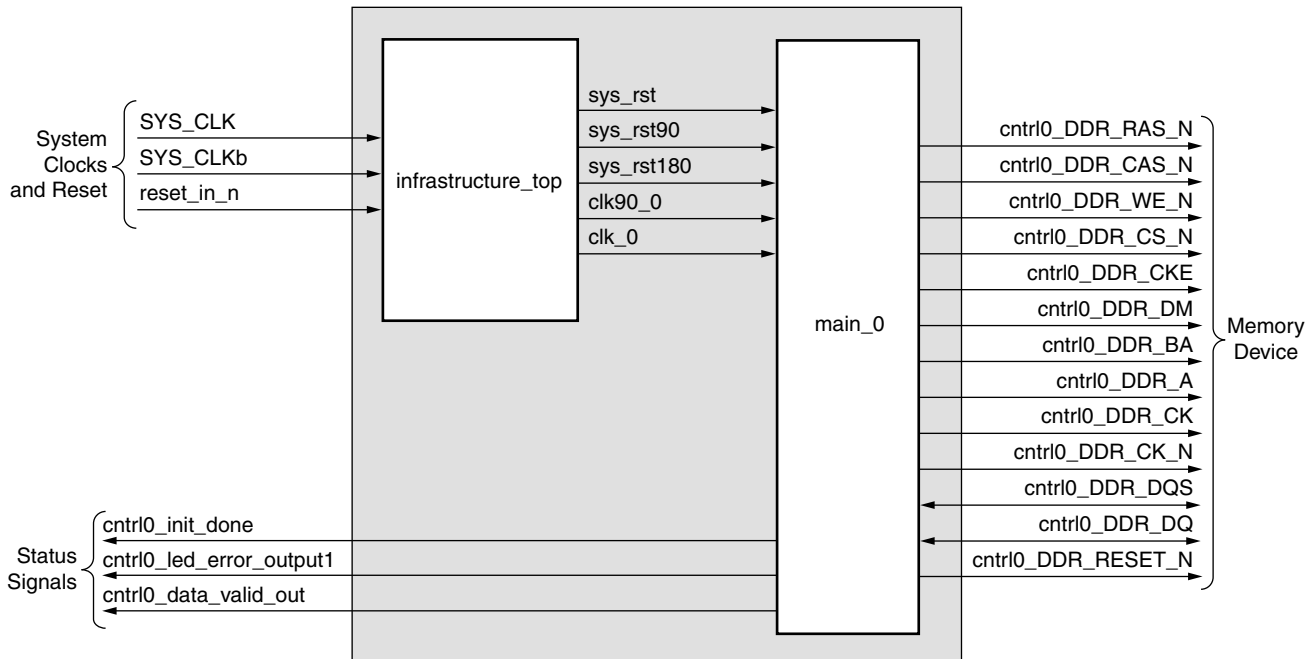
For designs generated without a testbench, the testbench modules in Figure 7-2 are not present in the design. In this case, the user interface signals appear in the <top\_module> module. The list of user interface signals is in Table 7-4.

The infrastructure\_top module has the clock and the reset generation module of the design. It instantiates a DCM in the module when selected by MIG. The differential design clock is an input to this module. A user reset is also input to this module. Using the input clocks and reset signals, system clocks and system reset are generated in this module which is used in the design. Infrastructure\_top also consists of calibration logic.

The DCM primitive is not instantiated in the infrastructure\_top module if the **Use DCM** option is unchecked. Therefore, the system operates on the user-provided clocks. The system reset is generated in the infrastructure module using the DCM\_LOCK input signal.

Figure 7-3 shows a block diagram representation of the top-level module of a DDR SDRAM design with a DCM and a testbench. SYS\_CLK and SYS\_CLKb are differential input system clocks. The DCM clock is instantiated in the infrastructure module that generates the required design clocks. reset\_in\_n is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

The cntrl0\_led\_error\_output1 output signal indicates whether the test passes or fails. When set, this signal indicates that the test has failed. The testbench module does writes and reads, and also compares the read data with the written data. The cntrl0\_data\_valid\_out signal indicates whether the read data is valid or not.



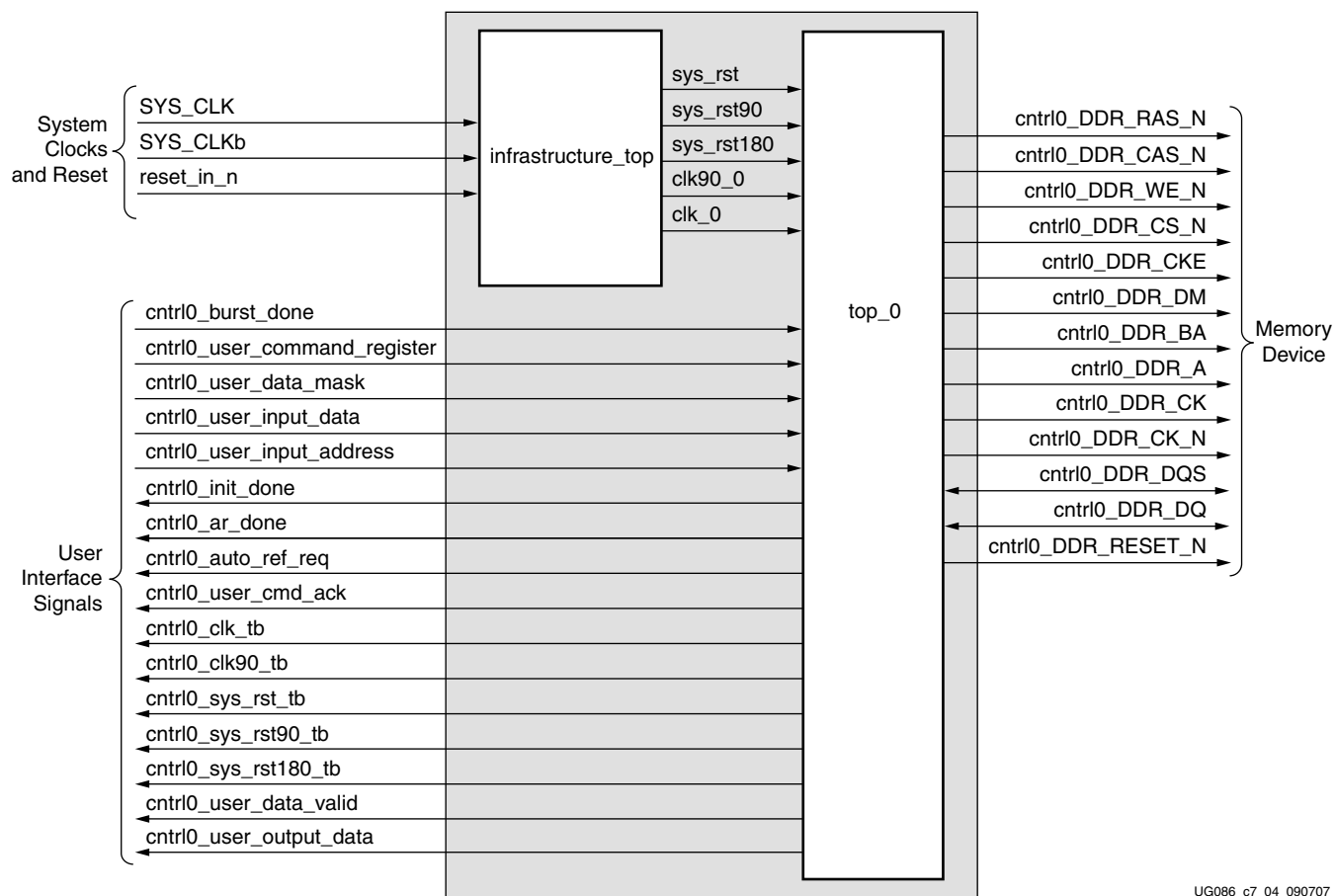
UG086\_c7\_03\_090707

Figure 7-3: MIG Output of the DDR SDRAM Controller Design with a DCM and a Testbench



Figure 7-4 shows a block diagram representation of the top-level module for a DDR SDRAM design with a DCM but without a testbench. SYS\_CLK and SYS\_CLKb are differential input system clocks. The DCM clock is instantiated in the infrastructure module that generates the required design clocks. reset\_in\_n is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

The user interface signals are listed in Figure 7-4. The design provides the clk\_tb, clk90\_tb, sys\_rst\_tb, sys\_rst90\_tb, and sys\_rst180\_tb signals to the user in order to synchronize with the design.

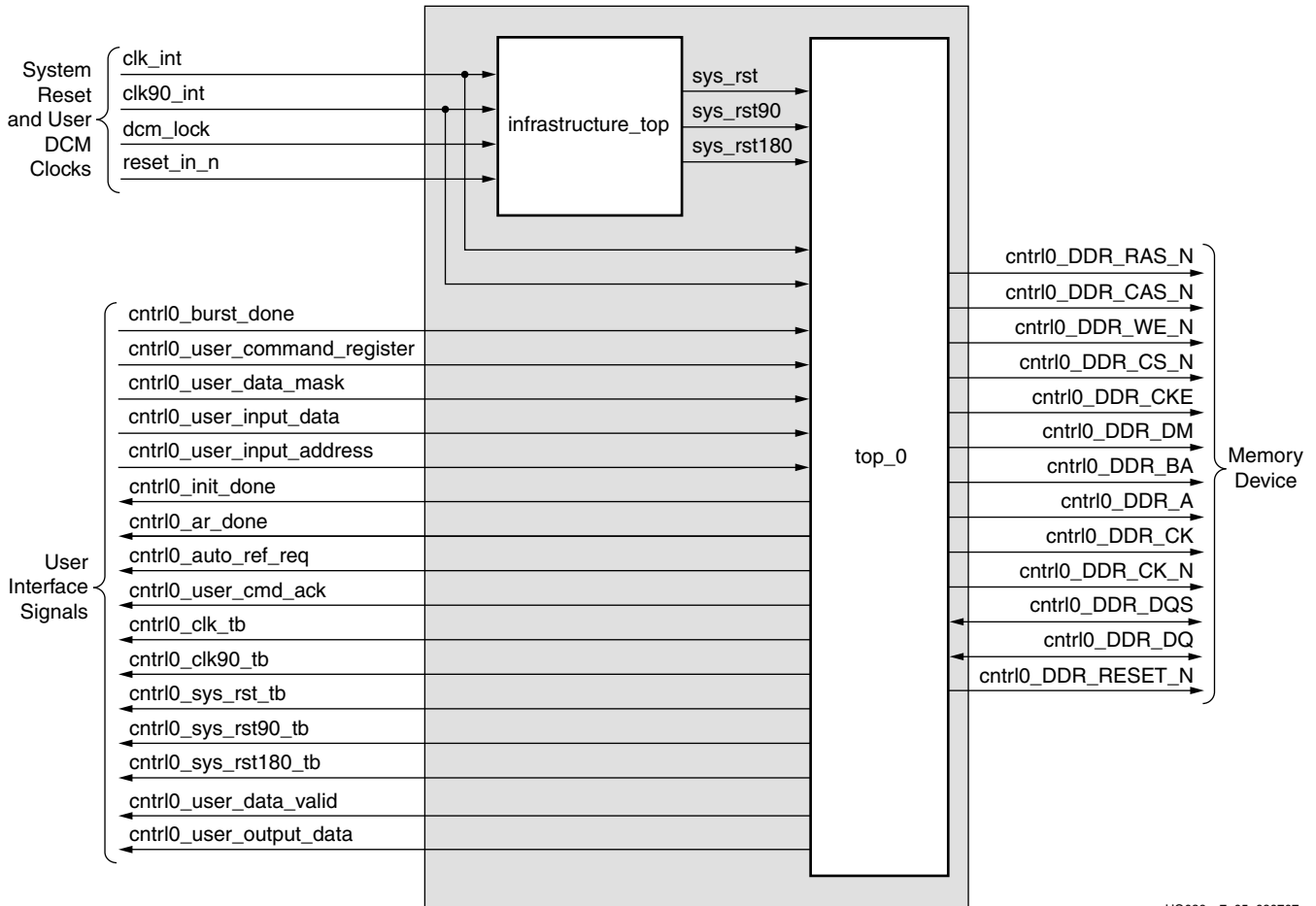


UG086\_c7\_04\_090707

Figure 7-4: MIG Output of the DDR SDRAM Controller Design with a DCM but without a Testbench

Figure 7-5 shows a block diagram representation of the top-level module for a DDR SDRAM design without a DCM or a testbench. The user should provide all the clocks and the dcm\_lock signal. These clocks should be single-ended. reset\_in\_n is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

The user interface signals are listed in Figure 7-5. The design provides the clk\_tb, clk90\_tb, sys\_rst\_tb, sys\_rst90\_tb, and sys\_rst180\_tb signals to the user in order to synchronize with the design.



UG086\_c7\_05\_090707

Figure 7-5: MIG Output of the DDR SDRAM Controller Design without a DCM or a Testbench

Figure 7-6 shows a block diagram representation of the top-level module of a DDR SDRAM design without a DCM but with a testbench. The user should provide all the clocks and the dcm\_lock signal. These clocks should be single-ended. reset\_in\_n is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

The cntrl0\_led\_error\_output1 output signal indicates whether the test passes or fails. The testbench module does writes and reads, and also compares the read data with the written data. The cntrl0\_led\_error\_output1 signal is driven High on data mismatches. The cntrl0\_data\_valid\_out signal indicates whether the read data is valid or not.

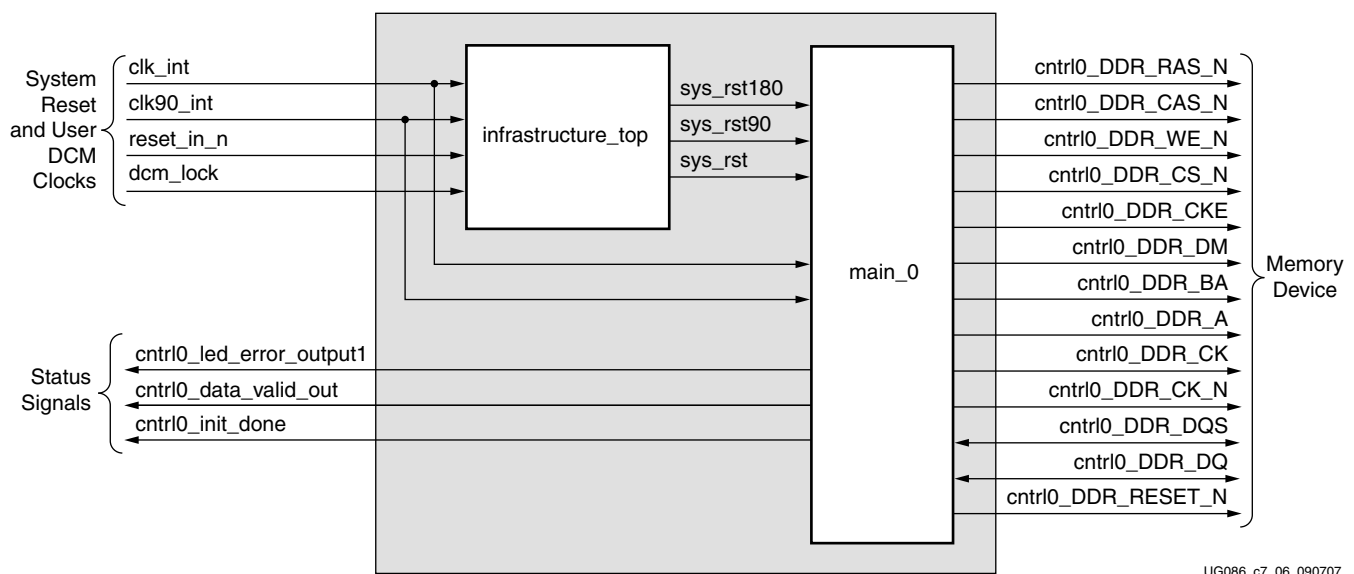


Figure 7-6: MIG Output of the DDR SDRAM Controller Design without a DCM but with a Testbench

All the memory device interface signals shown in Figure 7-3 through Figure 7-6 might not necessarily appear for all designs generated from MIG. For example, the `cntrl0_DDR_RESET_N` port appears in the port list for Registered DIMM designs only. Similarly, `cntrl0_ddr_dm` appears only for parts that have data mask signals. A few RDIMMs do not have data mask, and `cntrl0_DDR_DM` does not appear in the port list for these parts.

Figure 7-7 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.

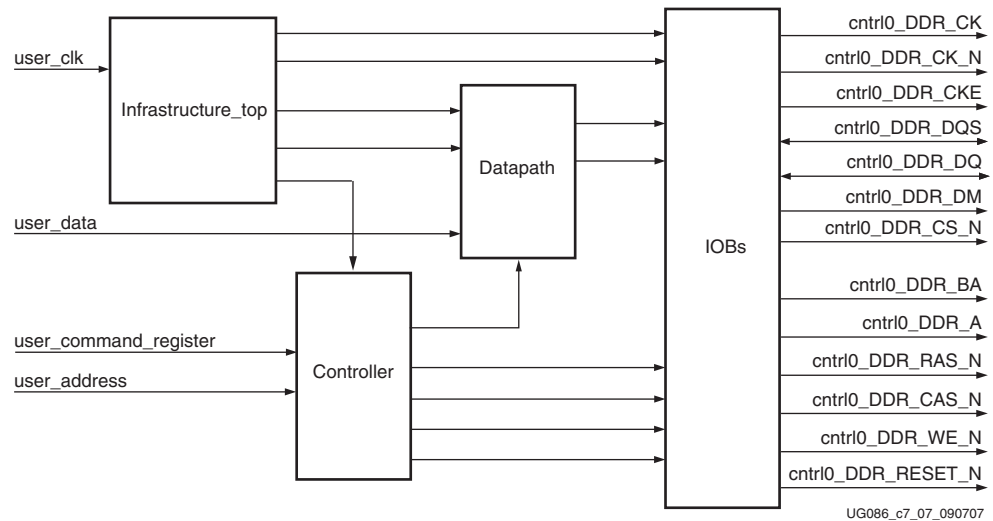


Figure 7-7: Memory Controller Block Diagram

## Controller

The controller module accepts and decodes user commands and generates read, write, and memory initialization 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 23].

## 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 23].

## 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 data\_path\_ioBs module.

## Infrastructure\_top

The infrastructure\_top 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 7-2 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. Active-Low polarity is indicated with \_N appended to the signal name. Table 7-2 is common for designs with and without testbenches. The signal cntrl0\_DDR\_RESET\_N is present only for registered DIMMs.

Table 7-2: **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
cntrl0_DDR_RESET_N	Output	Reset

Table 7-3 lists the DDR SDRAM clock, reset, and status signals for designs with and without testbenches. Except for the cntrl0\_led\_error\_output1 signal, all other signals in Table 7-3 are present in designs either with or without testbenches. The cntrl0\_led\_error\_output1 signal is present only in designs with a testbench.

Table 7-3: DDR SDRAM Clock, Reset, and Status 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. These two clocks should be generated from the same source (DCM output) with a 90° phase shift.
reset_in_n	Input	This signal is the system reset signal. By default, this signal is active Low. The parameter file contains a parameter called RESET_ACTIVE_LOW. An active-High reset input can be selected by changing this parameter to 0.
cntrl0_led_error_output1	Output	This signal is asserted when there is a read data mismatch with the write data. This signal is usually used to connect the LED on the hardware to indicate a data error.
cntrl0_data_valid_out	Output	This signal is asserted when there is valid read data in the read FIFO. The signal LED error output is generated when this signal is High and there is a data mismatch. This signal can be driven to a status LED on the hardware.
cntrl0_rst_dqs_div_in	Input	This loopback signal is connected to the cntrl0_rst_dqs_div_out signal on the board. Refer to XAPP768c [Ref 23] for the functionality of this signal.
cntrl0_rst_dqs_div_out	Output	This loopback signal is connected to the cntrl0_rst_dqs_div_in signal on the board.
dcm_lock	Input	This signal is present only in designs without a DCM.
cntrl0_init_done	Output	The DDR SDRAM controller asserts this signal to indicate that the DDR SDRAM initialization is complete.

Table 7-4 describes the DDR SDRAM controller user interface signals used between the ddr1\_top (design top-level module) and user application modules in designs without a testbench. These signals are buried one level down the hierarchy from memory interface top for with testbench design.

Table 7-4: DDR SDRAM Controller User Interface Signals (without a Testbench)

Signal Names	Direction <sup>(1)</sup>	Description												
cntrl0_user_input_data[(2n-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 $2n$ , the MSB is rising-edge data and the LSB is falling-edge data.												
cntrl0_user_data_mask[(2m-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 $2m$ , the MSB applies to rising-edge data and the LSB applies to falling-edge data.												
cntrl0_user_input_address [(ROW_ADDRESS + COLUMN_ADDRESS + BANK_ADDRESS - 1):0]	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, for a given memory if row_address = 13, column_address = 11, bank_address = 2, and the user_input_address = 26, then: <ul style="list-style-type: none"> <li>Bank Address from the user interface = A[1:0]</li> <li>Column Address from the user interface = A[12:2]</li> <li>Row Address part from the user interface = A[25:13]</li> </ul>												
cntrl0_user_command_register [2:0]	Input	Supported user commands for the DDR SDRAM controller: <table border="1" data-bbox="743 1014 1458 1287"> <thead> <tr> <th>user_command[2:0]</th> <th>User Command Description</th> </tr> </thead> <tbody> <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>110</td> <td>Read</td> </tr> <tr> <td>Others</td> <td>Reserved</td> </tr> </tbody> </table>	user_command[2:0]	User Command Description	000	NOP	010	Memory (DDR SDRAM) initialization	100	Write	110	Read	Others	Reserved
user_command[2:0]	User Command Description													
000	NOP													
010	Memory (DDR SDRAM) initialization													
100	Write													
110	Read													
Others	Reserved													
cntrl0_burst_done	Input	This signal is used to terminate a read or write command. This signal must be asserted after the last address for one clock for BL=2, two clocks for BL=4, and four clocks for BL =8. 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.												
cntrl0_user_output_data [(2n-1):0]	Output	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.												
cntrl0_user_data_valid	Output	When asserted, this signal indicates cntrl0_user_output_data[(2n-1):0] is valid.												

Table 7-4: DDR SDRAM Controller User Interface Signals (without a Testbench) (Continued)

Signal Names	Direction <sup>(1)</sup>	Description
cntrl0_user_cmd_ack	Output	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.
cntrl0_init_done	Output	The DDR SDRAM controller asserts this signal to indicate that the DDR SDRAM initialization is complete.
cntrl0_auto_ref_req	Output	This signal is asserted on every 7.7 $\mu$ s. 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 cntrl0_burst_done signal. To ensure reliable operation, users should terminate the current command within 15 to 20 clock cycles after cntrl0_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.
cntrl0_ar_done	Output	This indicates that the auto-refresh command was completed 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. The Refresh command is completed only after the assertion of the cntrl0_ar_done signal. The user can assert the next command any time after the assertion of the cntrl0_ar_done signal.

**Notes:**

1. All of the signal directions are with respect to the DDR SDRAM controller.



## Resource Utilization

A local inversion clocking technique is used in this design. The DCM generates only clk0 and clk90. One DCM and two BUFGMUXs are used. The Spartan 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  $\mu$ s after power up. For design optimization, a 200  $\mu$ s timer is generated from the refresh counter. The refresh timer is a function of frequency. Therefore, at lower frequencies, the 200  $\mu$ s timer waits more than 200  $\mu$ s. Because wait200 happens only during the power-up sequence, design performance is not degraded. All resets are asserted for 200  $\mu$ s because DDR SDRAM requires a 200  $\mu$ s delay prior to applying an executable command after all power supply and reference voltages are stable. The controller asserts the clock enable to memory after 200  $\mu$ s.

All the load mode register parameters are taken from the Mode Register values in the parameter file. The user has to enter the load mode parameters from the GUI while generating the design from MIG. When the Init command is received from the user interface, the controller starts DDR SDRAM initialization. The controller then writes this data into the Load Mode Register. Once the DDR SDRAM is initialized, the DDR SDRAM controller asserts the init\_done signal.

Figure 7-8 shows the timing for the memory initialization command.

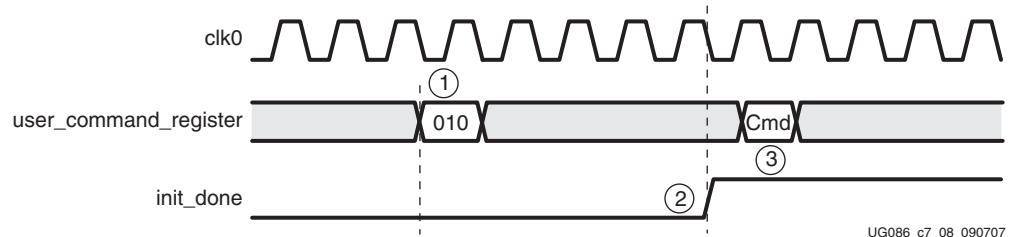


Figure 7-8: DDR SDRAM Initialization

1. 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.
2. The DDR SDRAM controller indicates that the initialization is complete by asserting the init\_done signal on a falling edge of clk0. The init\_done signal is asserted throughout the period.
3. After init\_done is asserted, the user can pass the next command at any time.

### DDR SDRAM Write and Read Operations

In Spartan designs, prior to issuing a read or write operation, the user must assert the first address and command simultaneously and wait for a command acknowledge signal. The assertion time of the command acknowledge varies depending on the controller status. After the command acknowledge is asserted, the user waits for three clock cycles before sending the next address. This three clock cycle time is the Active to Command ( $t_{RCD}$ ) delay for a read or write command as defined in the memory specification. Subsequent addresses are sent once every two clock cycles for a burst length of four.

## Write

Figure 7-9 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 command 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 + bank 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 2n, where n is data width of DDR SDRAM data bus.

For a burst length of four, only two data words (each of 2n) 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, for Burst Length = 4, 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.

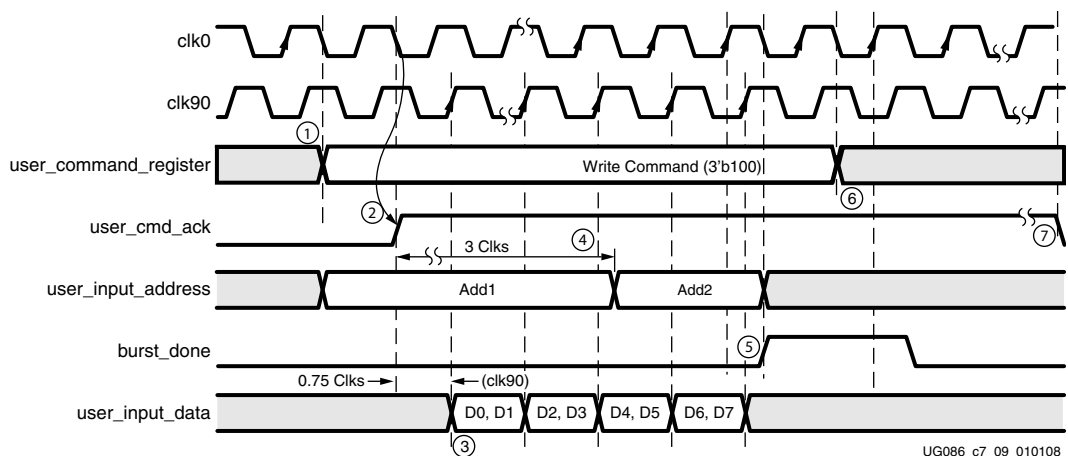


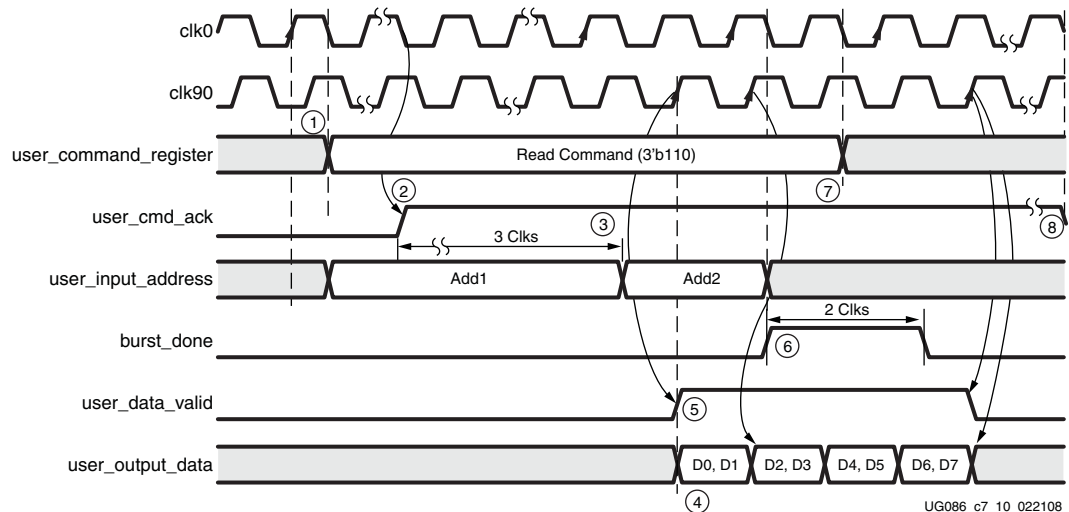
Figure 7-9: DDR SDRAM Write Burst, Burst Lengths of Four and Two Bursts

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 + bank 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.

## Read

The user initiates a memory read with a read command to the DDR SDRAM controller. [Figure 7-10](#) shows the memory read timing diagram for a burst length of four.



**Figure 7-10: 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 falling 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 `usr_cmd_ack` signal on a falling edge of `clk0`. The `usr_cmd_ack` signal is asserted a minimum of one clock cycle after the read command is asserted. 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 + bank 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  $2n$ , where  $n$  is the data width of the DDR SDRAMs. 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 `usr_cmd_ack` signal after completion of precharge to the memory. Any further commands to the DDR SDRAM controller should be given after `usr_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

The DDR SDRAM controller does a memory refresh periodically. Every 7.7  $\mu\text{s}$ , the controller raises an auto-refresh request. The user must terminate any ongoing commands within 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 for terminating the current transaction. 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 `ar_done` signal is asserted by the controller on completion of the auto-refresh command—i.e., after  $T_{\text{RFC}}$  time. The `ar_done` signal is asserted with `clk180` for one clock cycle.

The controller sets the `MAX_REF_CNT` value in the parameter file according to the frequency selected for a refresh interval (7.7  $\mu\text{s}$ ). The `rfc_count_value` value in the parameter file defines  $T_{\text{RFC}}$ , the time between the refresh command to Active or another refresh command.

After completion of the auto-refresh command, the next command can be given any time after `ar_done` is asserted.

## Changing the Refresh Rate

Change the global `define (for Verilog) or constant (for VHDL) variable MAX\_REF\_CNT in mymodule\_parameters\_0.v (or .vhd) so that MAX\_REF\_CNT = (refresh interval in clock periods) = (refresh interval) / (clock period). For example, for a refresh rate of 7.7  $\mu$ s with a memory bus running at 133 MHz:

$$\text{MAX\_REF\_CNT} = 7.7 \mu\text{s} / (\text{clock period}) = 7.7 \mu\text{s} / 7.5 \text{ ns} = 1026 \text{ (decimal)} = 0x402$$

If the above value exceeds  $2^{\text{MAX\_REF\_WIDTH}} - 1$ , the value of MAX\_REF\_WIDTH must be increased accordingly in parameters\_0.v (or .vhd) to increase the width of the counter used to track the refresh interval.

## Load Mode

MIG does not support the user LOAD MODE command. The mode register values from the parameter file are loaded into the Load Mode register during initialization.

## UCF Constraints

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

### 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_dly0/10" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/10" U_SET =
    delay_calibration_chain;
INST "infrastructure_top0/cal_top0/tap_dly0/gen_tap1[0].r" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/gen_tap1[0].r" U_SET =
    delay_calibration_chain;
INST "infrastructure_top0/cal_top0/tap_dly0/gen_tap1[1].r" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/gen_tap1[1].r" U_SET =
    delay_calibration_chain;
```

### 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/gen_delay[0].dqs_delay_col0/one"
LOC = SLICE_X0Y110;
INST "ddr1_top0/data_path0/data_read_controller0/gen_delay[0].dqs_delay_col0/one"
BEL = F;
NET "cntrl0_DDR_DQ[0]" LOC = Y4;
INST "ddr1_top0/data_path0/data_read0/gen_strobe[0].strobe/fifo0_bit0" LOC =
SLICE_X2Y111;
```

The I/O standards for all the memory interface signals are required to be specified.

## MAXDELAY Constraints

The MAXDELAY constraints define the maximum allowable delay on the net. Following are the list of MAXDELAY constraints used in Spartan FPGA designs in the UCF on different nets. The values provided here vary depending on FPGA family and the device type. Some values are dependent on frequency. The constraints shown here are from `example_design`. The hierarchy paths of the nets are different between `example_design` and `user_design`.

```
NET "infrastructure_top0/cal_top0/tap_dly0/tap[7]" MAXDELAY = 350ps;
NET "infrastructure_top0/cal_top0/tap_dly0/tap[15]" MAXDELAY = 350ps;
NET "infrastructure_top0/cal_top0/tap_dly0/tap[23]" MAXDELAY = 350ps;
```

These constraints are used to minimize the tap delay inverter connection wire length. This delay should be minimized to calibrate the delay of a tap (LUT element) accurately. These values are independent of frequency and vary from family to family and device to device. Without these constraints, the tool might synthesize longer routes between the tap connections. Inappropriate delays in this circuit could cause the design to fail in hardware.

```
NET "main_00/top0/dqs_int_delay_in*" MAXDELAY = 675ps;
```

This constraint is used for the DQS nets from the I/O pad to the input of the LUT delay chain. Without this constraint, the nets take unpredictable delays that affect the Data Valid window. In Spartan designs, data is latched using the DQS signal. In order to latch the correct data, DQS is delayed using LUT delay elements to center-align with respect to the input read data. Incorrect data could be latched if the delays on this net are unpredictable. Unpredictable delays might also cause the design to have intermittent failures, which are difficult to debug in hardware.

```
NET "main_00/top0/dqs_div_rst" MAXDELAY = 460ps;
```

The net `dqs_div_rst` is the loopback signal. This signal is used as an enable for read data FIFOs and FIFO write pointers after it is delayed using the LUT delay elements. The overall delay on this net should be comparable with the delay on the DQS signal. This net is constrained to control the overall delay. Both the `dqs_div_rst` and DQS signals take similar paths. If the delay on the `dqs_div_rst` signal is higher, the first read data from memory might be missed.

```
NET
"main_00/top0/data_path0/data_read_controller0/gen_delay*dqs_delay_col
*/delay*" MAXDELAY = 140ps;
NET
"main_00/top0/data_path0/data_read_controller0/rst_dqs_div_delayed/
delay*" MAXDELAY = 140 ps;
```

These constraints are required to minimize the wire delays between the LUT elements of a LUT delay chain that is used to delay the DQS and `rst_dqs_div` loopback signal. Higher wire delays between LUT delay elements can shift the data valid window, which in turn can cause incorrect data to be latched. Therefore, the MAXDELAY constraint is required for these nets.

```
NET "main_00/top0/data_path0/data_read_controller0/rst_dqs_div"
MAXDELAY = 3383 ps;
NET "main_00/top0/data_path0/data_read0/fifo*_wr_en*"
MAXDELAY = 3007ps;
```

These constraints are required because these paths are not constrained otherwise. The total delay on the `rst_dqs_div` and `fifo_wr_en` nets must not exceed the clock period. The total delay on both the nets is set to 85% of the clock period, leaving 15% as margin. These delays vary with frequency.

```
NET "main_00/top0/data_path0/data_read0/fifo*_wr_addr[*]"  
MAXDELAY = 5610ps;
```

The MAXDELAY constraint is required on FIFO write address because this path is not constrained otherwise. This is a single clock cycle path. It is set to 80% of the clock period, leaving 20% as margin because this net generally meets the required constraint.

## 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. Refer to the “Rules Concerning Banks” section for additional information regarding I/O banking rules in DS099 [Ref 27] and DS312 [Ref 28].

## Design Notes

### Spartan-3/3E/3A/3AN/3A DSP Pin Allocation Rules

The pin allocation rules are different for top/bottom and left/right banks because of the local clock structure of Spartan FPGAs.

#### Pin Allocation Rules for Left/Right Banks

1. When a DQS is allocated, its associated DQ bits should be allocated within five tiles above and six tiles below the DQS tile.
2. The DQ bits should not be allocated in the DQS tile.
3. The rst\_dqs\_div signal should be placed in the center of the data bank.

#### Pin Allocation Rules for Top/Bottom Banks

1. All DQ bits corresponding to DQS are required to be placed to the right of its DQS tile.
2. All DQ bits corresponding to the DQS should be within five I/O tiles of the DQS tile.
3. A DQ bit should not be allocated in the same I/O tile where DQS is allocated.

#### Top/Bottom Bank Support

MIG does not support top/bottom banks for Spartan 3E/3A/3AN/3A DSP devices. For some I/O pads, the fabric slices are not located next to the IOBs. These I/O pads cannot be used for pin allocation. By excluding these I/O pins, there are not enough pins to allocate DQ and DQS signals according to the pin allocation rules.



## Supported Devices

This section provides tables for the memory components supported by Spartan-3, Spartan-3A, Spartan-3AN, Spartan-3A DSP, and Spartan-3E devices.

The design generated out of MIG is independent of memory speed grade, hence the package part of the memory component is replaced with X, where X indicates a don't care condition.

The tables below list the components (Table 7-5) and DIMMs (Table 7-6 through Table 7-8) supported by the tool for Spartan-3 FPGA DDR local clocking designs.

**Table 7-5: Supported Components for DDR SDRAM Local Clocking (Spartan-3 FPGAs)**

Components	Packages (XX)	Components	Packages (XX)
MT46V32M4XX-5B	-	MT46V32M4XX-75	P,TG
MT46V64M4XX-5B	BG,FG,P,TG	MT46V64M4XX-75	FG,P,TG
MT46V128M4XX-5B	BN,FN,P,TG	MT46V128M4XX-75	BN,FN,P,TG
MT46V256M4XX-5B	P,TG	MT46V256M4XX-75	P,TG
MT46V16M8XX-5B	TG,P	MT46V16M8XX-75	P,TG
MT46V32M8XX-5B	BG,FG,P,TG	MT46V32M8XX-75	FG,P,TG
MT46V64M8XX-5B	BN,FN,P,TG	MT46V64M8XX-75	BN,FN,P,TG
MT46V128M8XX-5B	-	MT46V128M8XX-75	P,TG
MT46V8M16XX-5B	TG,P	MT46V8M16XX-75	P,TG
MT46V16M16XX-5B	BG,FG,P,TG	MT46V16M16XX-75	BG,FG,P,TG
MT46V32M16XX-5B	BN,FN,P,TG	MT46V32M16XX-75	-
MT46V64M16XX-5B	-	MT46V64M16XX-75	P,TG

**Table 7-6: Supported Unbuffered DIMMs for DDR SDRAM Local Clocking (Spartan-3 FPGAs)**

Unbuffered DIMMs	Packages (X)	Unbuffered DIMMs	Packages (X)
MT4VDDT1664AX-40B	G,Y	MT8VDDT3264AX-40B	G,Y
MT4VDDT3264AX-40B	G,Y	MT9VDDT3272AX-40B	Y

**Table 7-7: Supported Registered DIMMs for DDR SDRAM Local Clocking (Spartan-3 FPGAs)**

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9VDDF3272X-40B	G,Y	MT18VDDF3272X-40B	D,G,Y
MT9VDDF3272X-40B	G,Y	MT18VDDF12872X-40B	DY,G,Y



**Table 7-8: Supported SODIMMs for DDR SDRAM Local Clocking (Spartan-3 FPGAs)**

SODIMMs	Packages (X)	SODIMMs	Packages (X)
MT4VDDT3264HX-40B	G,Y	MT9VDDT3272HX-40B	
MT4VDDT1664HX-40B	Y	MT9VDDT6472HX-40B	G,Y
MT8VDDT3264HX-40B	-	MT9VDDT12872HX-40B	-
MT8VDDT6464HX-40B	DG,DY,G,Y		

The tables below list the components (Table 7-9) and DIMMs (Table 7-10 through Table 7-12) supported by the tool for Spartan-3A/AN DDR local clocking designs.

**Table 7-9: Supported Components for DDR SDRAM Local Clocking (Spartan-3A/AN FPGAs)**

Components	Packages (XX)	Components	Packages (XX)
MT46V32M4XX-5B	-	MT46V32M4XX-75	P,TG
MT46V64M4XX-5B	BG,FG,P,TG	MT46V64M4XX-75	FG,P,TG
MT46V128M4XX-5B	BN,FN,P,TG	MT46V128M4XX-75	BN,FN,P,TG
MT46V256M4XX-5B	P,TG	MT46V256M4XX-75	P,TG
MT46V16M8XX-5B	TG,P	MT46V16M8XX-75	P,TG
MT46V32M8XX-5B	BG,FG,P,TG	MT46V32M8XX-75	FG,P,TG
MT46V64M8XX-5B	BN,FN,P,TG	MT46V64M8XX-75	BN,FN,P,TG
MT46V128M8XX-5B	-	MT46V128M8XX-75	P,TG
MT46V8M16XX-5B	TG,P	MT46V8M16XX-75	P,TG
MT46V16M16XX-5B	BG,FG,P,TG	MT46V16M16XX-75	BG,FG,P,TG
MT46V32M16XX-5B	BN,FN,P,TG	MT46V32M16XX-75	-
MT46V64M16XX-5B	-	MT46V64M16XX-75	P,TG

**Table 7-10: Supported Unbuffered DIMMs for DDR SDRAM Local Clocking (Spartan-3A/AN FPGAs)**

Unbuffered DIMMs	Packages (X)	Unbuffered DIMMs	Packages (X)
MT4VDDT1664AX-40B	G,Y	MT8VDDT3264AX-40B	G,Y
MT4VDDT3264AX-40B	G,Y		

**Table 7-11: Supported Registered DIMMs for DDR SDRAM Local Clocking (Spartan-3A/AN FPGAs)**

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9VDDF3272X-40B	G,Y	MT9VDDF3272X-40B	G,Y

**Table 7-12: Supported SODIMMs for DDR SDRAM Local Clocking (Spartan-3A/AN FPGAs)**

SODIMMs	Packages (X)	SODIMMs	Packages (X)
MT4VDDT3264HX-40B	G,Y	MT8VDDT3264HX-40B	-
MT4VDDT1664HX-40B	Y	MT8VDDT6464HX-40B	DG,DY,G,Y

The tables below list the components (Table 7-13) and DIMMs (Table 7-14 and Table 7-15) supported by the tool for Spartan-3A DSP DDR local clocking designs.

**Table 7-13: Supported Components for DDR SDRAM Local Clocking (Spartan-3A DSP FPGAs)**

Components	Packages (XX)	Components	Packages (XX)
MT46V32M4XX-5B	-	MT46V32M4XX-75	P,TG
MT46V64M4XX-5B	BG,FG,P,TG	MT46V64M4XX-75	FG,P,TG
MT46V128M4XX-5B	BN,FN,P,TG	MT46V128M4XX-75	BN,FN,P,TG
MT46V256M4XX-5B	P,TG	MT46V256M4XX-75	P,TG
MT46V16M8XX-5B	TG,P	MT46V16M8XX-75	P,TG
MT46V32M8XX-5B	BG,FG,P,TG	MT46V32M8XX-75	FG,P,TG
MT46V64M8XX-5B	BN,FN,P,TG	MT46V64M8XX-75	BN,FN,P,TG
MT46V128M8XX-5B	-	MT46V128M8XX-75	P,TG
MT46V8M16XX-5B	TG,P	MT46V8M16XX-75	P,TG
MT46V16M16XX-5B	BG,FG,P,TG	MT46V16M16XX-75	BG,FG,P,TG
MT46V32M16XX-5B	BN,FN,P,TG	MT46V32M16XX-75	-
MT46V64M16XX-5B	-	MT46V64M16XX-75	P,TG

**Table 7-14: Supported Unbuffered DIMMs for DDR SDRAM Local Clocking (Spartan-3A DSP FPGAs)**

Unbuffered DIMMs	Packages (X)	Unbuffered DIMMs	Packages (X)
MT4VDDT1664AX-40B	G,Y	MT8VDDT3264AX-40B	G,Y
MT4VDDT3264AX-40B	G,Y		

**Table 7-15: Supported SODIMMs for DDR SDRAM Local Clocking (Spartan-3A DSP FPGAs)**

SODIMMs	Packages (X)	SODIMMs	Packages (X)
MT4VDDT3264HX-40B	G,Y	MT8VDDT3264HX-40B	-
MT4VDDT1664HX-40B	Y	MT8VDDT6464HX-40B	DG,DY,G,Y

Table 7-16 lists the components supported by the tool for Spartan-3E FPGA DDR local clocking designs.

**Table 7-16: Supported Components for DDR SDRAM Local Clocking (Spartan-3E FPGAs)**

Components	Packages (XX)	Components	Packages (XX)
MT46V32M4XX-5B	-	MT46V32M4XX-75	P,TG
MT46V64M4XX-5B	BG,FG,P,TG	MT46V64M4XX-75	FG,P,TG
MT46V128M4XX-5B	BN,FN,P,TG	MT46V128M4XX-75	BN,FN,P,TG
MT46V256M4XX-5B	P,TG	MT46V256M4XX-75	P,TG
MT46V16M8XX-5B	TG,P	MT46V16M8XX-75	P,TG
MT46V32M8XX-5B	BG,FG,P,TG	MT46V32M8XX-75	FG,P,TG
MT46V64M8XX-5B	BN,FN,P,TG	MT46V64M8XX-75	BN,FN,P,TG
MT46V128M8XX-5B	-	MT46V128M8XX-75	P,TG
MT46V8M16XX-5B	TG,P	MT46V8M16XX-75	P,TG
MT46V16M16XX-5B	BG,FG,P,TG	MT46V16M16XX-75	BG,FG,P,TG
MT46V32M16XX-5B	BN,FN,P,TG	MT46V32M16XX-75	-
MT46V64M16XX-5B	-	MT46V64M16XX-75	P,TG

## Simulating the Spartan-3/3E/3A/3AN/3A DSP FPGA Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for the generated design. The `sim` folder contains the external testbench, memory model, `.do` file, and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more details about the files in the `sim` folder and to simulate the design, see `simulation_help.chm` in the `sim` folder.

## Hardware Tested Configurations

The frequencies shown in [Table 7-17](#) and [Table 7-18](#) were achieved on the Spartan-3 FPGA Memory Interface Board and Spartan-3E FPGA Starter Kit, respectively, under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based on a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 64-bit wide interface.

**Table 7-17: Hardware Tested Configurations for Spartan-3 FPGA DDR SDRAM Designs**

Synthesis Tools	XST
HDL	Verilog and VHDL
FPGA Device	XC3S1500FG676-5
Burst Lengths	2 and 8
CAS Latency (CL)	2 and 2.5
64-bit Design	Tested on 16-bit Component "MT46V16M16XX-75"
	64-bit DIMM "MT4VDDT3264AX"
Frequency Range	67 MHz to 170 MHz for CL = 2
	40 MHz to 190 MHz for CL = 2.5

**Table 7-18: Hardware Tested Configurations for Spartan-3E FPGA DDR SDRAM Designs**

Synthesis Tools	XST
HDL	Verilog and VHDL
FPGA Device	XC3S500EFG320-4
Burst Lengths	2 and 4
CAS Latency (CL)	2 and 2.5
16-bit Design	Tested on 16-bit Component "MT46V32M16XX-6T"
Frequency Range	80 MHz to 170 MHz for CL = 2
	80 MHz to 170 MHz for CL = 2.5

## Implementing DDR2 SDRAM Controllers

---

This chapter describes how to implement DDR2 SDRAM interfaces for Spartan™-3, Spartan-3E, Spartan-3A, Spartan-3AN, and Spartan-3A DSP FPGAs generated by MIG. This design is based on XAPP768c [Ref 23].

### Feature Summary

The DDR2 SDRAM controller design supports:

- Burst lengths of four and eight
- Sequential and interleaved burst types
- 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
- Spartan-3E maximum frequency:
  - ◆ 133 MHz with a -4 speed grade device
  - ◆ 166 MHz with a -5 speed grade device
- Spartan-3A, Spartan-3AN, and Spartan-3A DSP 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
- With and without a testbench
- With or without a DCM

## Design Frequency Ranges

Table 8-1: Design Frequency Range in MHz

FPGA Family	Memory	FPGA Speed Grade			
		-4		-5	
		Min	Max	Min	Max
Spartan-3	Component	125	133	125	166 <sup>(1)</sup>
	DIMM	125	133	125	133
Spartan-3E	Component	125	133	125	166
	DIMM	Not supported			
Spartan-3A/3AN/3A DSP	Component	125	133	125	166
	DIMM	125	133	125	166

**Notes:**

1. Spartan-3 devices support 133 MHz for data widths greater than 32 bits.

## 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.

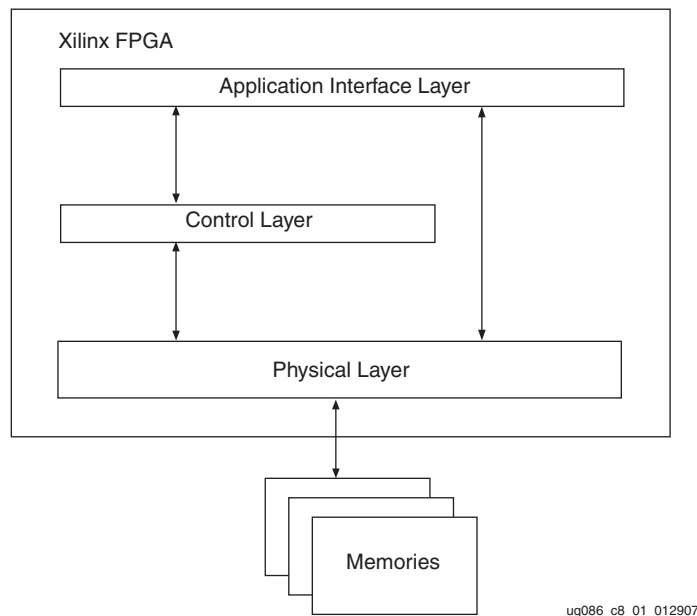
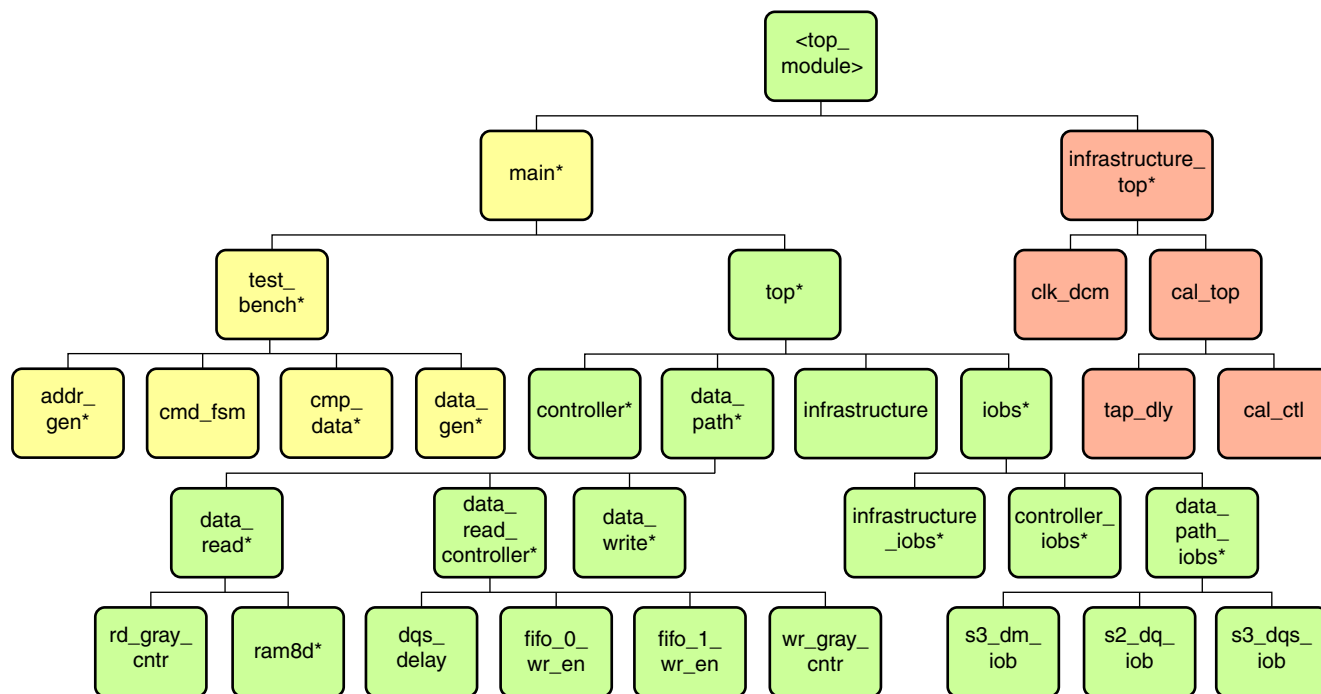


Figure 8-1: Modular Memory Interface Representation

## Hierarchy

Figure 8-2 shows the hierarchical structure of the DDR2 SDRAM design generated by MIG with a testbench and a DCM. In the figure, the physical and control layers are clearly separated. MIG generates the entire controller, as shown in this hierarchy, including the testbench. The user can replace the testbench with a design that makes use of the DDR2 SDRAM interface.



- Design Modules
- Test Bench Modules
- Clocks, Reset Generation, and Calibration Modules

Note: A block with a \* has a parameter file included.

UG086\_c8\_02\_010108

Figure 8-2: Hierarchical Structure of the Design

The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks, reset generation, and calibration modules

There is a parameter file generated with the design that has all the user input and design parameters selected from MIG.

MIG can generate four different DDR2 SDRAM designs:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

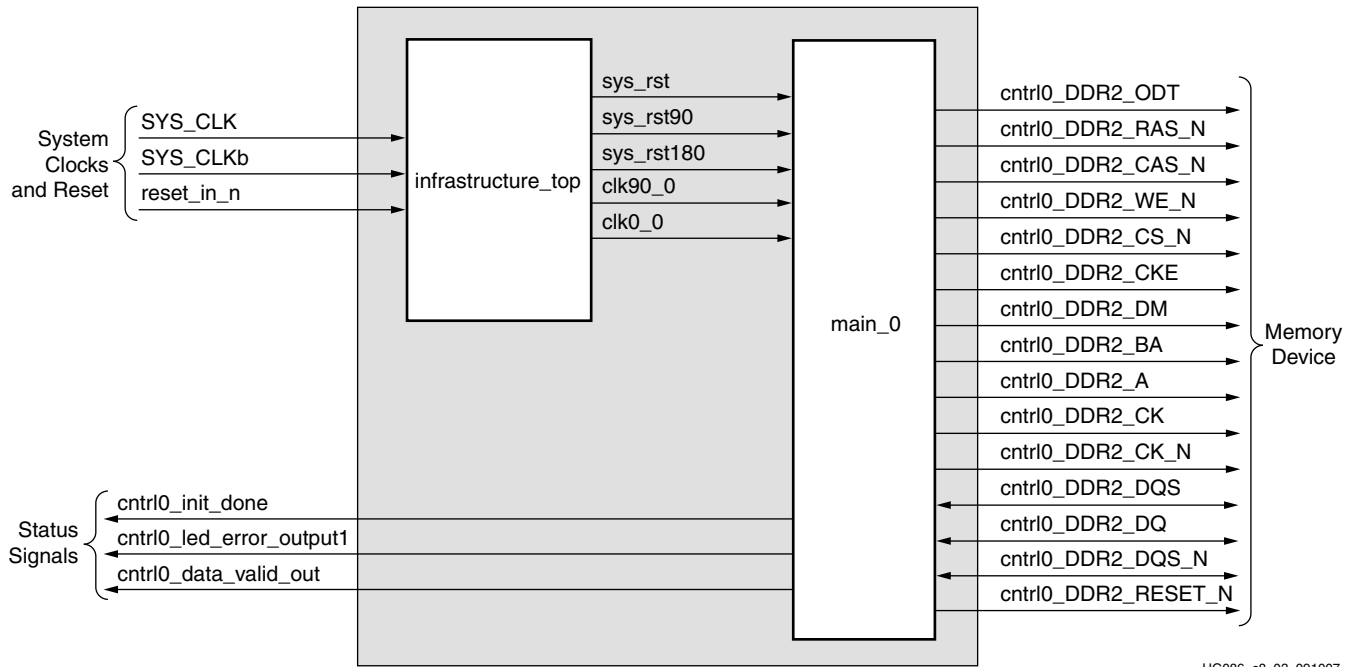
For a design without a testbench (user\_design), the shaded modules in Figure 8-2 are not present in the design. The <top\_module> module has the user interface signals for designs without a testbench. The list of user interface signals is provided in Table 8-4.

The infrastructure\_top module comprises the clock and the reset generation module of the design. It instantiates a DCM in the module when selected by MIG. The differential design clock is an input to this module. A user reset is also input to this module. Using the input clocks and reset signals, system clocks and system reset are generated in this module which is used in the design. Infrastructure\_top also consists of calibration logic.

The DCM primitive is not instantiated in this module if the **Use DCM** option is unchecked. Therefore, the system operates on the user-provided clocks. The system reset is generated in the infrastructure module using the DCM\_LOCK input signal.

Figure 8-3 shows a block diagram representation of the top-level module for a DDR2 SDRAM design with a DCM and a testbench. SYS\_CLK and SYS\_CLKb are differential input system clocks. The DCM clock is instantiated in the infrastructure module that generates the required design clocks. reset\_in\_n is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

The cntrl0\_led\_error\_output1 output signal indicates whether the case passes or fails. The testbench module does writes and reads, and also compares the read data with written data. The cntrl0\_led\_error\_output1 signal is driven High on data mismatches. The cntrl0\_data\_valid\_out signal indicates whether the read data is valid or not.



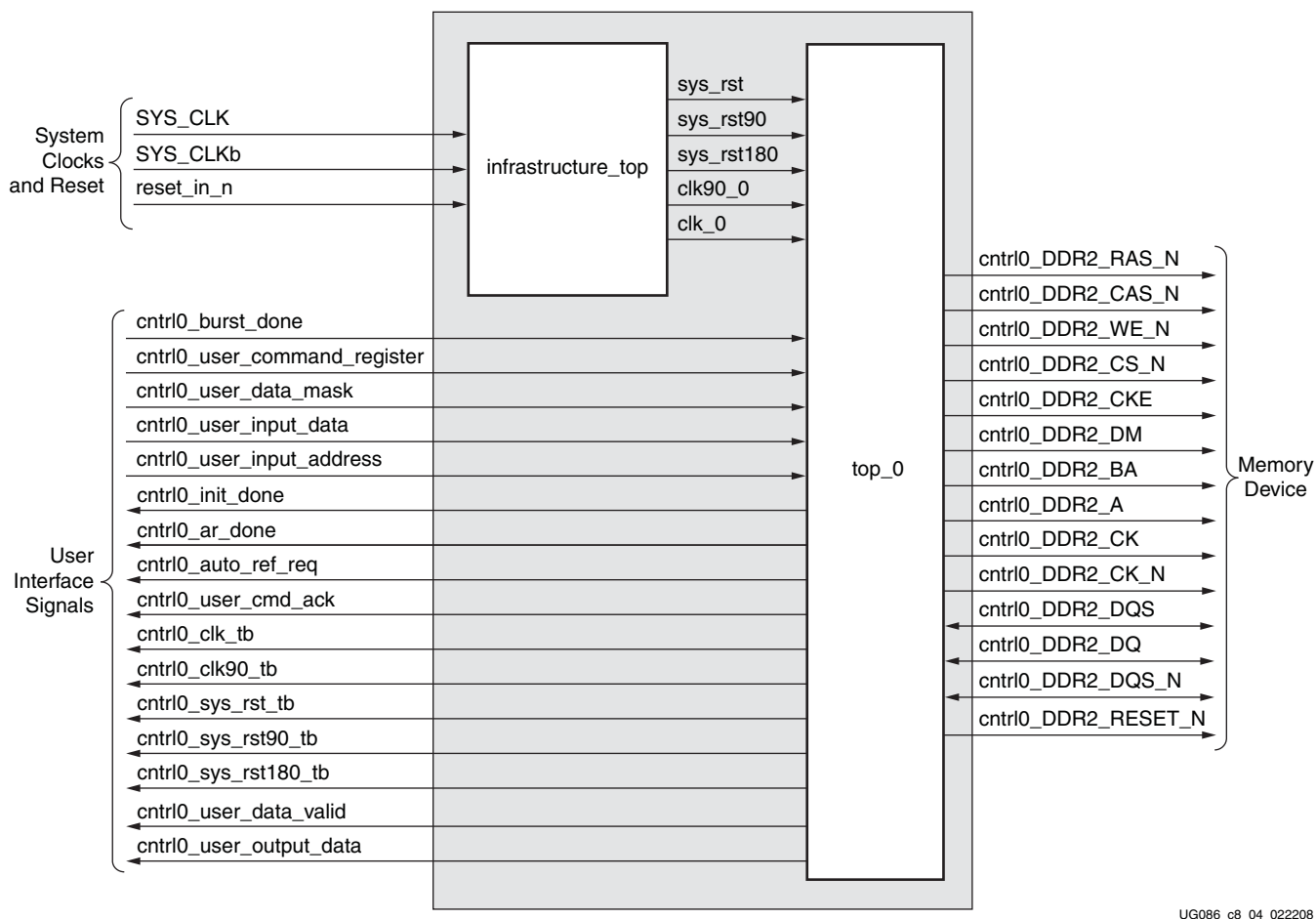
UG086\_c8\_03\_091007

Figure 8-3: MIG Output of the DDR2 SDRAM Controller Design with a DCM and a Testbench



Figure 8-4 shows a block diagram representation of the top-level module for a DDR2 SDRAM design with a DCM but without a testbench. SYS\_CLK and SYS\_CLKb are differential input system clocks. The DCM clock is instantiated in the infrastructure module that generates the required design clocks. reset\_in\_n is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

The user interface signals are listed in Figure 8-4. The design provides the clk\_tb, clk90\_tb, sys\_rst\_tb, sys\_rst90\_tb, and sys\_rst180\_tb signals to the user in order to synchronize with the design.

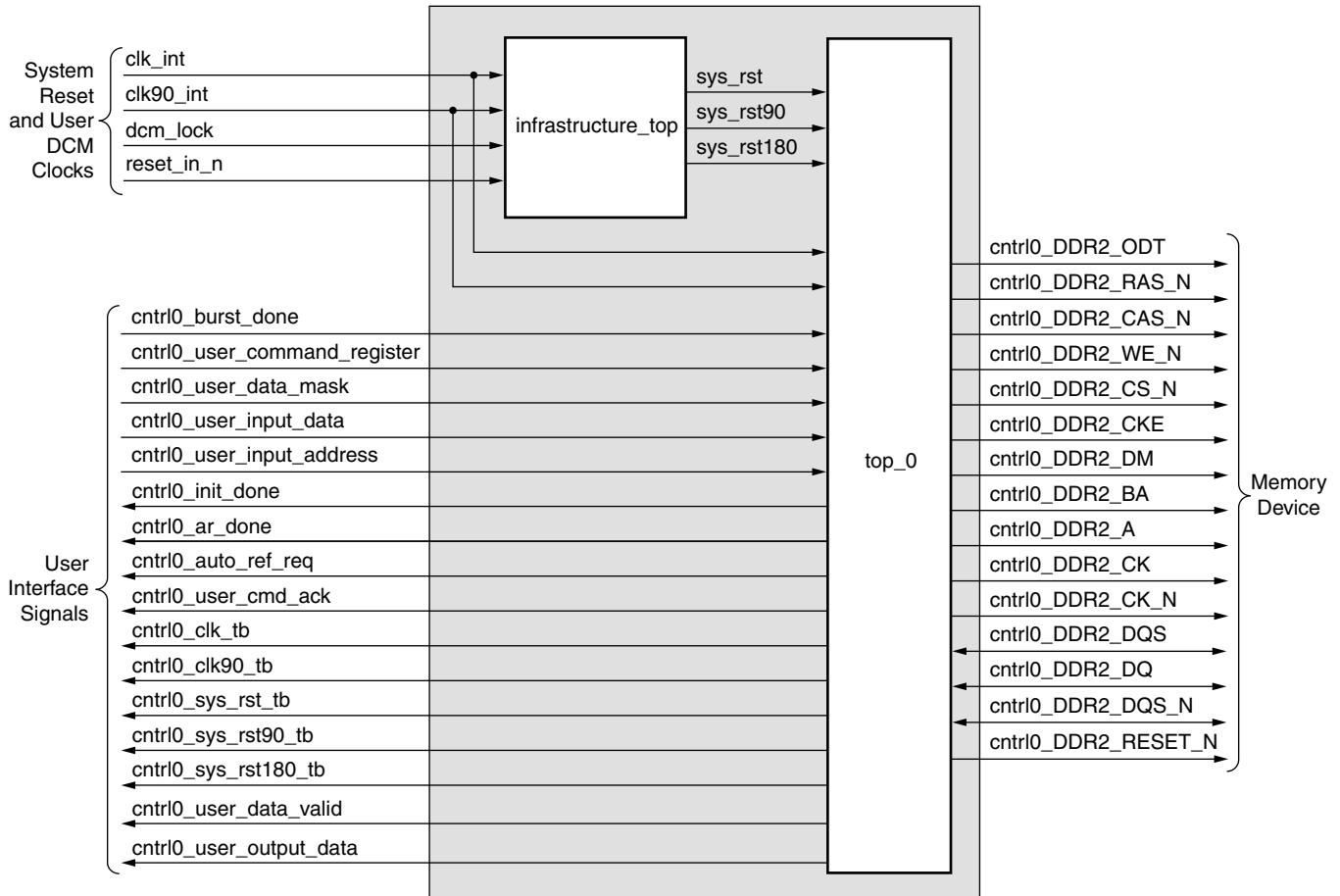


UG086\_c8\_04\_022208

Figure 8-4: MIG Output of the DDR2 SDRAM Controller Design with a DCM but without a Testbench

Figure 8-5 shows a block diagram representation of the top-level module for a DDR2 SDRAM design without a DCM or a testbench. The user should provide all the clocks and the dcm\_lock signal. These clocks should be single-ended. reset\_in\_n is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

The user interface signals are listed in Figure 8-5. The design provides the clk\_tb, clk90\_tb, sys\_rst\_tb, sys\_rst90\_tb, and sys\_rst180\_tb signals to the user in order to synchronize with the design.



UG086\_c8\_05\_091007

Figure 8-5: MIG Output of the DDR2 SDRAM Controller Design without a DCM or a Testbench

Figure 8-6 shows a block diagram representation of the top-level module for a DDR2 SDRAM design without a DCM but with a testbench. The user should provide all the clocks and the dcm\_lock signal. These clocks should be single-ended. reset\_in\_n is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal.

The cntrl0\_led\_error\_output1 output signal indicates whether the case passes or fails. The testbench module does writes and reads, and also compares the read data with the written data. The cntrl0\_led\_error\_output1 signal is driven High on data mismatches. The cntrl0\_data\_valid\_out signal indicates whether the read data is valid or not.

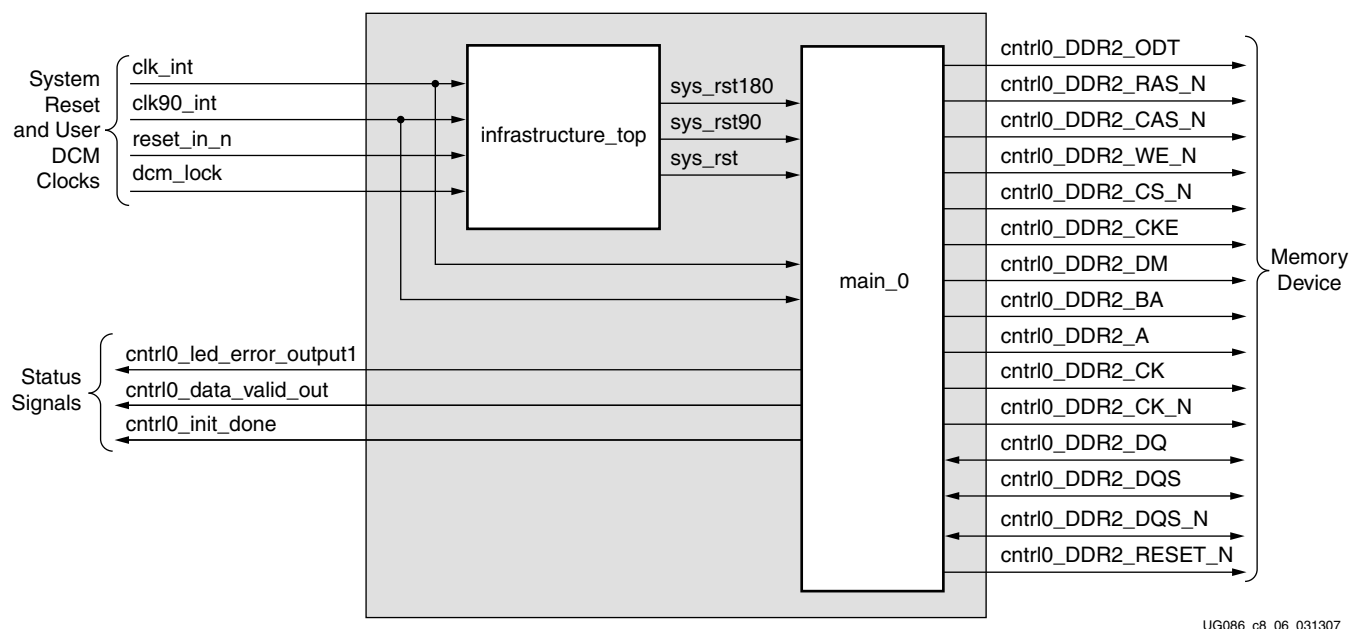


Figure 8-6: MIG Output of the DDR2 SDRAM Controller Design without a DCM but with a Testbench

All the Memory Device interface signals that are shown in Figure 8-3 through Figure 8-6 do not necessarily appear for all designs that are generated from MIG. For example, port cntrl0\_ddr2\_RESET\_N appears in the port list only for Registered DIMM designs. Similarly, cntrl0\_ddr2\_DQS\_N does not appear for single-ended DQS designs. Port cntrl0\_ddr2\_dm appears only for the parts that contain a data mask. A few RDIMMs do not have a data mask, and cntrl0\_ddr2\_dm does not appear in the port list for these parts.

Figure 8-7 shows a detailed block diagram of the DDR2 SDRAM controller. All four blocks shown are sub-blocks of the ddr2\_top module. The functionality of these blocks is explained in following sections.

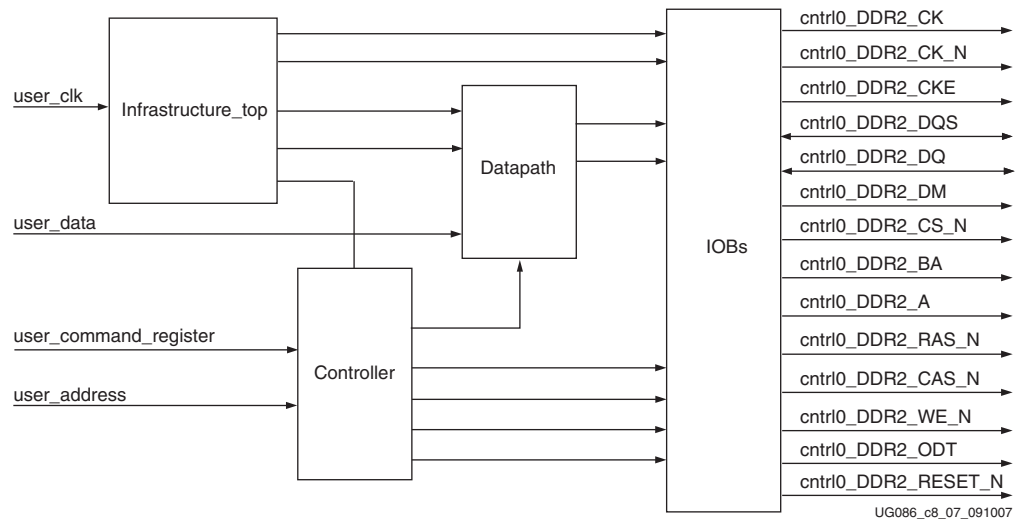


Figure 8-7: Memory Controller Block Diagram

## 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 23].

## 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 23].

## 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\_top

The infrastructure\_top 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 8-2 shows the DDR2 SDRAM interface signals, directions, and descriptions. The signal direction is with respect to the DDR2 SDRAM controller. The cntrl0\_ddr2\_reset\_n signal is present only for registered DIMMs, and the cntrl0\_ddr2\_dqs\_n signal is present when DQS# Enable is selected in the Extended Mode register.

Table 8-2: 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_DQS_N	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
cntrl0_DDR2_RESET_N	Output	Reset

Table 8-3 describes the DDR2 SDRAM controller system interface signals. Except for the `cntrl0_led_error_output1` signal, all other signals in Table 8-3 are present in designs either with or without testbenches. The `cntrl0_led_error_output1` signal is present only in designs with a testbench.

Table 8-3: DDR2 SDRAM Controller System Interface Signals

Signal Names	Direction	Description
<code>SYS_CLK</code> and <code>SYS_CLKb</code>	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.
<code>reset_in_n</code>	Input	This is the system reset signal. By default, this signal is active Low. The parameter file contains a parameter called <code>RESET_ACTIVE_LOW</code> . An active-High reset input can be selected by changing this parameter to 0.
<code>cntrl0_led_error_output1</code>	Output	This signal is asserted when there is a read data mismatch with the write data. This signal is usually used to connect the LED on the hardware to indicate a data error.
<code>cntrl0_data_valid_out</code>	Output	This signal is asserted when there is valid read data in the read FIFO. The signal LED error output is generated when this signal is High and there is a data mismatch. This signal can be driven to a status LED on the hardware.
<code>cntrl0_rst_dqs_div_in</code>	Input	This loopback signal is connected to the <code>cntrl0_rst_dqs_div_out</code> signal on the board. Refer to XAPP768c [Ref 23] for the functionality of this signal.
<code>cntrl0_rst_dqs_div_out</code>	Output	This loopback signal is connected to the <code>cntrl0_rst_dqs_div_in</code> signal on the board.
<code>dcm_lock</code>	Input	This signal is present only in designs without a DCM.
<code>cntrl0_init_done</code>	Output	The DDR2 SDRAM controller asserts this signal to indicate that the DDR2 SDRAM initialization is complete.

Table 8-4 describes the DDR2 SDRAM controller system interface signals in designs without a testbench.

Table 8-4: DDR2 SDRAM Controller User Interface Signals (without a Testbench)

Signal Names	Direction <sup>(1)</sup>	Description												
cntrl0_user_input_data[(2n-1):0]	Input	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.												
cntrl0_user_data_mask[(2m-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 $2m$ , the MSB applies to rising-edge data and the LSB applies to falling-edge data.												
cntrl0_user_input_address [(ROW_ADDRESS + COLUMN_ADDRESS + BANK_ADDRESS - 1):0] <sup>(2)</sup>	Input	This bus consists of the row address, the column address, and the bank address for DDR2 SDRAM writes and reads. The address sequence starting from the LSB is bank address, column address, and row address.												
cntrl0_user_command_register[2:0]	Input	Supported user commands for the DDR2 SDRAM controller:												
		<table border="1"> <thead> <tr> <th>user_command[2:0]</th> <th>User Command Description</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>NOP</td> </tr> <tr> <td>010</td> <td>Initialize memory</td> </tr> <tr> <td>100</td> <td>Write Request</td> </tr> <tr> <td>110</td> <td>Read Request</td> </tr> <tr> <td>Others</td> <td>Reserved</td> </tr> </tbody> </table>	user_command[2:0]	User Command Description	000	NOP	010	Initialize memory	100	Write Request	110	Read Request	Others	Reserved
		user_command[2:0]	User Command Description											
		000	NOP											
		010	Initialize memory											
		100	Write Request											
110	Read Request													
Others	Reserved													
cntrl0_burst_done	Input	This signal is used to terminate read or write command. This signal must be asserted after the last address for two clocks for BL=4 and for four clocks for BL =8. 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.												
cntrl0_user_output_data[(2n-1):0]	Output	This is the read data from the DDR2 SDRAM. The DDR2 SDRAM controller converts the DDR data from the DDR2 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 DDR2 SDRAM data bus.												
cntrl0_user_data_valid	Output	When asserted, this signal indicates user_output_data[(2n-1):0] is valid.												

Table 8-4: DDR2 SDRAM Controller User Interface Signals (without a Testbench) (Continued)

Signal Names	Direction <sup>(1)</sup>	Description
cntrl0_user_cmd_ack	Output	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.
cntrl0_init_done	Output	The DDR2 SDRAM controller asserts this signal to indicate that the DDR2 SDRAM initialization is complete.
cntrl0_auto_ref_req	Output	This signal is asserted on every 7.7 $\mu$ s. 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 cntrl0_burst_done signal. The frequency with which this signal is asserted is determined by the MAX_REF_CNT value in parameter file. cntrl0_auto_ref_req indicates the refresh request to the memory, and cntrl0_ar_done indicates completion of the auto-refresh command.
cntrl0_ar_done	Output	This indicates that the auto-refresh command was completed 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 can assert the next command any time after the assertion of the cntrl0_ar_done signal.

**Notes:**

1. All of the signal directions are with respect to the DDR2 SDRAM controller.



## Resource Utilization

A local inversion clocking technique is used in this design. The DCM generates only clk0 and clk90. One DCM and two BUFGMUXs are used. The Spartan 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  $\mu$ s after power up. For design optimization, a 200  $\mu$ s timer is generated from the refresh counter. The refresh timer is a function of frequency. Therefore, at lower frequencies, the 200  $\mu$ s timer waits more than 200  $\mu$ s. Because wait200 happens only during the power-up sequence, design performance is not degraded. All resets are asserted for 200  $\mu$ s because DDR2 SDRAM requires a 200  $\mu$ s 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  $\mu$ s.

Load mode parameters are to be selected from the GUI while generating the design. These parameters are updated by MIG in the parameter file. When the INIT command is executed, the DDR2 SDRAM controller passes these values to the Memory Load Mode register. When the DDR2 SDRAM is initialized, the DDR2 SDRAM controller asserts the init\_done signal.

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

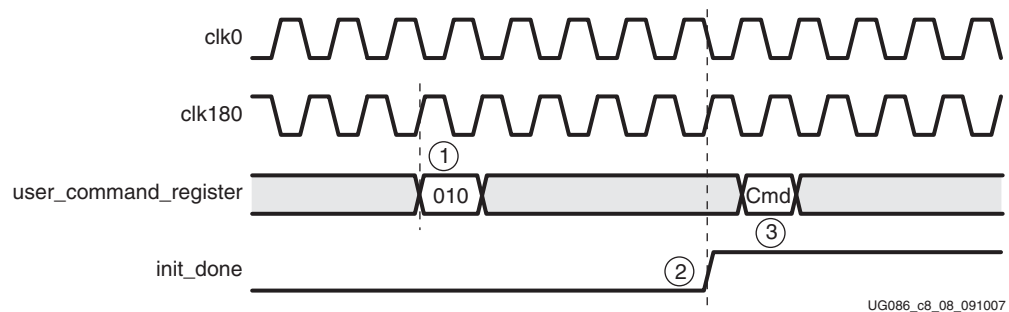


Figure 8-8: DDR2 SDRAM Initialization

1. 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.
2. The DDR2 SDRAM controller indicates that the initialization is complete by asserting the init\_done signal on a falling edge of clk0. The init\_done signal is asserted throughout the period.
3. After init\_done is asserted, the user can pass the next command at any time.

## Write

Figure 8-9 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 falling edge of clk0. In response to a write command, the DDR2 SDRAM controller acknowledges with the usr\_cmd\_ack signal on a falling edge of clk0. 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 + bank address) with the write command and keeps it asserted for three clocks after usr\_cmd\_ack assertion. Any subsequent write addresses are asserted on alternate falling edges of clk0 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 DDR data, the width of this bus is 2n, where n is data width of DDR2 SDRAM data bus.

For a burst length of four, only two data words (each of 2n) 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.

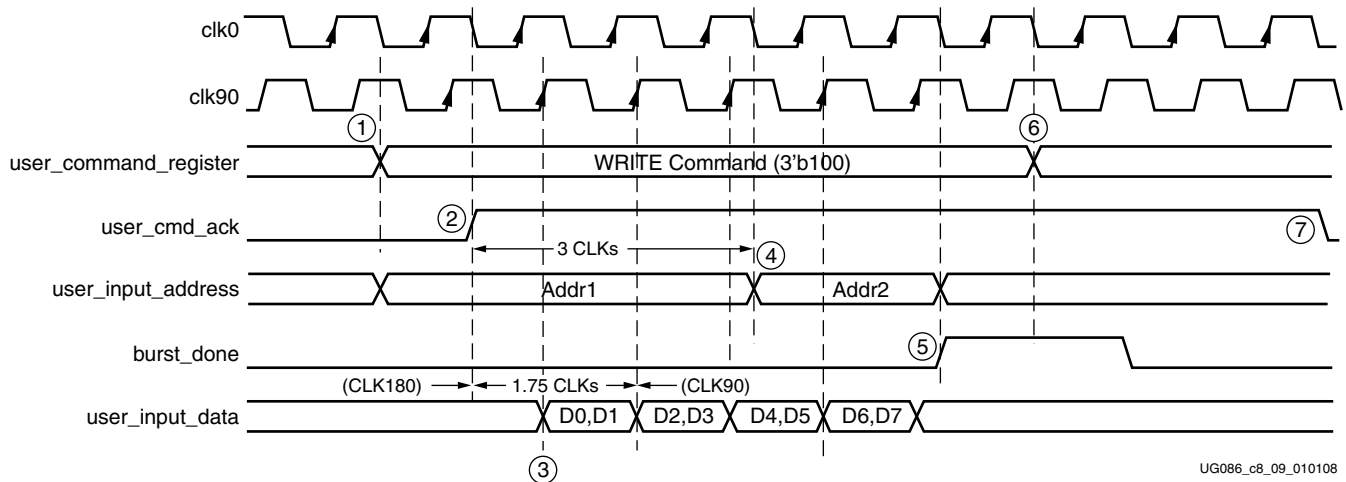


Figure 8-9: DDR2 SDRAM Write Burst, Burst Lengths of Four and Two Bursts

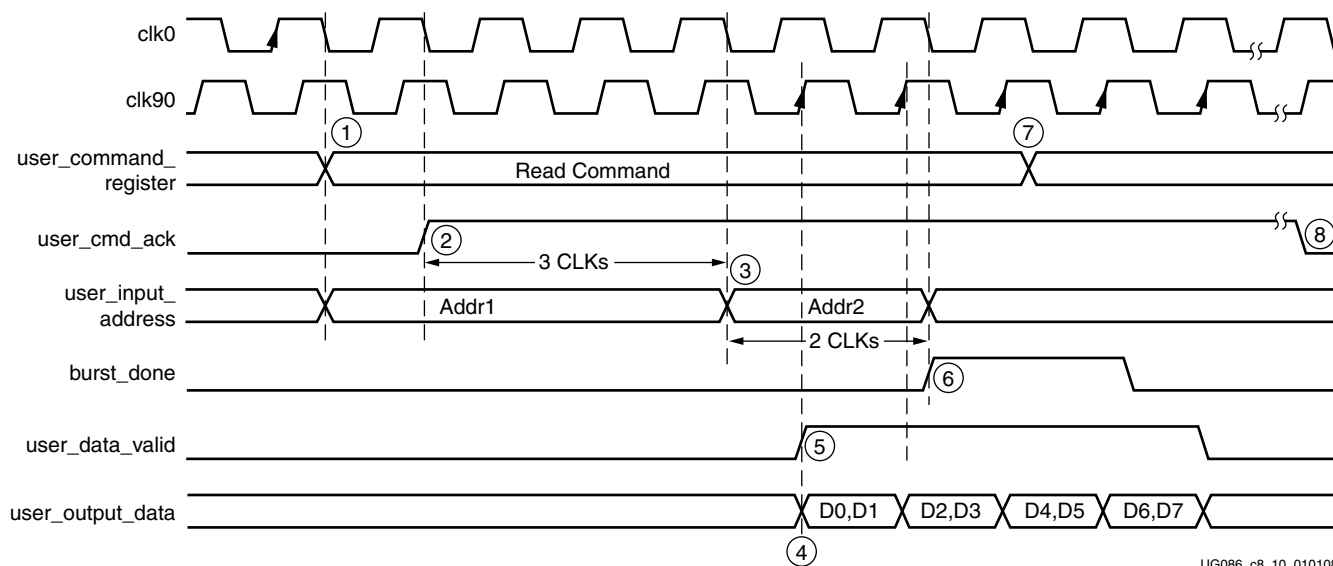
1. A memory write is initiated by issuing a write command to the DDR2 SDRAM controller. The write command must be asserted on a falling edge of clk0.
2. The DDR2 SDRAM controller acknowledges the write command by asserting the user\_cmd\_ack signal on a falling edge of clk0. The user\_cmd\_ack signal is asserted a minimum of one clock cycle after the write command is asserted. If the controller is

busy with a refresh, the `usr_cmd_ack` signal is not asserted until after the refresh command cycle completes.

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 + bank 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`.
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.

## Read

The user initiates a memory read with a read command to the DDR2 SDRAM controller. [Figure 8-10](#) shows the memory read timing diagram for a burst length of four.



**Figure 8-10: 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 falling edge of `clk0` 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 falling edge of `clk0`.
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 falling edge of `clk0`. The `user_cmd_ack` signal is asserted a minimum of one clock cycle after the read command is asserted. If the controller is busy with a refresh, the `usr_cmd_ack` signal is not asserted until after the refresh command cycle completes.
3. The user asserts the first address (row + column + bank 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`.
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  $2n$ , where  $n$  is the data width of the DDR2 SDRAMs. 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 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 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

The DDR2 SDRAM controller does a memory refresh periodically. Every 7.7  $\mu$ s, 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 `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 on the falling edge of `clk0` for one clock cycle.

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  $\mu$ s). The `rfc_count_value` setting in the parameter file defines  $T_{RFC}$ , the time between the refresh command to Active or another refresh command.

After completion of the auto-refresh command, the next command can be given any time after `ar_done` is asserted.

## Changing the Refresh Rate

Change the global ``define` (for Verilog) or constant (for VHDL) variable `MAX_REF_CNT` in `mymodule_parameters_0.v` (or `.vhd`) so that `MAX_REF_CNT = (refresh interval in`

clock periods) = (refresh interval) / (clock period). For example, for a refresh rate of 7.7  $\mu$ s with a memory bus running at 133 MHz:

$$\text{MAX\_REF\_CNT} = 7.7 \mu\text{s} / (\text{clock period}) = 7.7 \mu\text{s} / 7.5 \text{ ns} = 1026 \text{ (decimal)} = 0x402$$

If the above value exceeds  $2^{\text{MAX\_REF\_WIDTH}} - 1$ , the value of MAX\_REF\_WIDTH must be increased accordingly in `parameters_0.v` (or `.vhd`) to increase the width of the counter used to track the refresh interval.

## Load Mode

MIG does not support the LOAD MODE command.

## UCF Constraints

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

### 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_dly0/10" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/10" U_SET =
    delay_calibration_chain;
INST "infrastructure_top0/cal_top0/tap_dly0/gen_tap1[0].r" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/gen_tap1[0].r" U_SET =
    delay_calibration_chain;
INST "infrastructure_top0/cal_top0/tap_dly0/gen_tap1[1].r" RLOC=X0Y6;
INST "infrastructure_top0/cal_top0/tap_dly0/gen_tap1[1].r" U_SET =
    delay_calibration_chain;
```

### 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/gen_delay[0].dqs_delay_col0/
one" LOC = SLICE_X0Y110;
INST "ddr2_top0/data_path0/data_read_controller0/gen_delay[0].dqs_delay_col0/
one" BEL = F;
NET "cntrl0_DDR2_DQ[0]" LOC = Y5;
INST "ddr2_top0/data_path0/data_read0/gen_strobe[0].strobe/fifo0_bit0" LOC =
SLICE_X2Y111;
```

### MAXDELAY Constraints

The MAXDELAY constraints define the maximum allowable delay on the net. Following are the list of MAXDELAY constraints used in Spartan FPGA designs in the UCF on different nets. The values provided here vary depending on FPGA family and the device type. Some values are dependent on frequency. The constraints shown here are from `example_design`. The hierarchy paths of the nets are different between `example_design` and `user_design`.

```
NET "infrastructure_top0/cal_top0/tap_dly0/tap[7]" MAXDELAY = 350ps;
NET "infrastructure_top0/cal_top0/tap_dly0/tap[15]" MAXDELAY = 350ps;
NET "infrastructure_top0/cal_top0/tap_dly0/tap[23]" MAXDELAY = 350ps;
```

These constraints are used to minimize the tap delay inverter connection wire length. This delay should be minimized to calibrate the delay of a tap (LUT element) accurately. These values are independent of frequency and vary from family to family and device to device. Without these constraints, the tool might synthesize longer routes between the tap connections. Inappropriate delays in this circuit could cause the design to fail in hardware.

```
NET "main_00/top0/dqs_int_delay_in*" MAXDELAY = 675ps;
```

This constraint is used for the DQS nets from the I/O pad to the input of the LUT delay chain. Without this constraint, the nets take unpredictable delays that affect the Data Valid window. In Spartan designs, data is latched using the DQS signal. In order to latch the correct data, DQS is delayed using LUT delay elements to center-align with respect to the input read data. Incorrect data could be latched if the delays on this net are unpredictable. Unpredictable delays might also cause the design to have intermittent failures, which are difficult to debug in hardware.

```
NET "main_00/top0/dqs_div_rst" MAXDELAY = 460ps;
```

The net `dqs_div_rst` is the loopback signal. This signal is used as an enable for read data FIFOs and FIFO write pointers after it is delayed using the LUT delay elements. The overall delay on this net should be comparable with the delay on the DQS signal. This net is constrained to control the overall delay. Both the `dqs_div_rst` and DQS signals take similar paths. If the delay on the `dqs_div_rst` signal is higher, the first read data from memory might be missed.

```
NET
"main_00/top0/data_path0/data_read_controller0/gen_delay*dqs_delay_col
*/delay*" MAXDELAY = 140ps;
NET
"main_00/top0/data_path0/data_read_controller0/rst_dqs_div_delayed/
delay*" MAXDELAY = 140 ps;
```

These constraints are required to minimize the wire delays between the LUT elements of a LUT delay chain that is used to delay the DQS and `rst_dqs_div` loopback signal. Higher wire delays between LUT delay elements can shift the data valid window, which in turn can cause incorrect data to be latched. Therefore, the MAXDELAY constraint is required for these nets.

```
NET "main_00/top0/data_path0/data_read_controller0/rst_dqs_div"
MAXDELAY = 3383 ps;
NET "main_00/top0/data_path0/data_read0/fifo*_wr_en*"
MAXDELAY = 3007ps;
```

These constraints are required because these paths are not constrained otherwise. The total delay on the `rst_dqs_div` and `fifo_wr_en` nets must not exceed the clock period. The total delay on both the nets is set to 85% of the clock period, leaving 15% as margin. These delays vary with frequency.

```
NET "main_00/top0/data_path0/data_read0/fifo*_wr_addr[*]"
MAXDELAY = 5610ps;
```

The MAXDELAY constraint is required on FIFO write address because this path is not constrained otherwise. This is a single clock cycle path. It is set to 80% of the clock period, leaving 20% as margin because this net generally meets the required constraint.

## 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. Refer to the “Rules Concerning Banks” section for additional information regarding I/O banking rules in DS099 [Ref 27].

## Design Notes

The DDR2 SDRAM design is not validated on hardware. The MAXDELAY constraints in the UCF are set based on the selected frequency.

Calibration circuit details and data capture techniques are covered in XAPP768c [Ref 23].

## Tool Output

When the design is generated from the tool, it outputs `docs`, `example_design`, and `user_design` folders. The `example_design` consists of the design *with* `test_bench`, and `user_design` consists of the design *without* `test_bench`. Each folder contains `rtl`, `par`, `synth`, and `sim` folders. The `sim` folder contains simulation files for the generated design. The `sim` folder contains the external testbench, memory model, `.do` file, and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more details about the files in the `sim` folder and to simulate the design, see `simulation_help.chm` in the `sim` folder.

## Supported Devices

The design generated out of MIG is independent of memory speed grade, hence the package part of the memory component is replaced with X, where X indicates a don't care condition.

The tables below list the components (Table 8-5) and DIMMs (Table 8-6 through Table 8-8) supported by the tool for Spartan-3 DDR2 local clocking designs.

**Table 8-5: Supported Components for DDR2 SDRAM Local Clocking (Spartan-3 FPGAs)**

Components	Packages (XX)	Components	Packages (XX)
MT47H64M4XX-3	BP	MT47H128M8XX-3	BT,HQ
MT47H64M4XX-37E	BP	MT47H128M8XX-37E	BT,HQ
MT47H64M4XX-5E	BP	MT47H128M8XX-5E	BT,HQ
MT47H128M4XX-3	B6,CB,GB	MT47H256M8XX-3	HG
MT47H128M4XX-37E	B6,CB,GB	MT47H256M8XX-37E	HG
MT47H128M4XX-5E	B6,CB,GB	MT47H256M8XX-5E	HG
MT47H256M4XX-3	BT,HQ	MT47H16M16XX-3	BG
MT47H256M4XX-37E	BT,HQ	MT47H16M16XX-37E	BG
MT47H256M4XX-5E	BT,HQ	MT47H16M16XX-5E	BG
MT47H512M4XX-3	HG	MT47H32M16XX-3	BN,CC,FN,GC
MT47H512M4XX-37E	HG	MT47H32M16XX-37E	BN,CC,FN,GC
MT47H512M4XX-5E	HG	MT47H32M16XX-5E	BN,CC,FN,GC



**Table 8-5: Supported Components for DDR2 SDRAM Local Clocking (Spartan-3 FPGAs) (Continued)**

Components	Packages (XX)	Components	Packages (XX)
MT47H32M8XX-3	BP	MT47H64M16XX-3	BT,HR
MT47H32M8XX-37E	BP	MT47H64M16XX-37E	BT,HR
MT47H32M8XX-5E	BP	MT47H64M16XX-5E	BT,HR
MT47H64M8XX-3	B6,CB,F6,GB	MT47H128M16XX-3	HG
MT47H64M8XX-37E	B6,CB,F6,GB	MT47H128M16XX-37E	HG
MT47H64M8XX-5E	B6,CB,F6,GB	MT47H128M16XX-5E	--

**Table 8-6: Supported Unbuffered DIMMs for DDR2 SDRAM Local Clocking (Spartan-3 FPGAs)**

Unbuffered DIMMs	
MT4HTF1664AY-667	MT8HTF6464AY-53E
MT4HTF1664AY-40E	MT8HTF6464AY-40E
MT4HTF3264AY-667	MT8HTF12864AY-667
MT4HTF3264AY-40E	MT8HTF12864AY-40E
MT4HTF6464AY-667	MT9HTF3272AY-667
MT4HTF6464AY-40E	MT9HTF3272AY-40E
MT8HTF6464AY-667	MT9HTF6472AY-667

**Table 8-7: Supported Registered DIMMs for DDR2 SDRAM Local Clocking (Spartan-3 FPGAs)**

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9HTF3272XX-53E	Y	MT18HTF6472XX-53E	DY,Y
MT9HTF3272XX-40E	Y	MT18HTF6472XX-40E	DY,Y
MT9HTF6472XX-53E	Y	MT18HTF12872XX-53E	DY,MY,NDY, NY,PY,Y
MT9HTF6472XX-40E	Y	MT18HTF12872XX-40E	DY,PY,Y
MT9HTF12872XX-53E	PY,Y	MT18HTF25672XX-53E	PDY,PY,Y
MT9HTF12872XX-40E	Y	MT18HTF25672XX-40E	DY,PDY,Y

**Table 8-8: Supported SODIMMs for DDR2 SDRAM Local Clocking (Spartan-3 FPGAs)**

SODIMMs	
MT4HTF1664HY-667	MT8HTF3264HY-667
MT4HTF1664HY-53E	MT8HTF3264HY-53E
MT4HTF1664HY-40E	MT8HTF3264HY-40E
MT4HTF3264HY-667	MT8HTF6464HY-667
MT4HTF3264HY-53E	MT8HTF6464HY-53E
MT4HTF3264HY-40E	MT8HTF6464HY-40E



The tables below list the components (Table 8-9) and DIMMs (Table 8-10 through Table 8-12) supported by the tool for Spartan-3A/AN DDR2 local clocking designs.

**Table 8-9: Supported Components for DDR2 SDRAM Local Clocking (Spartan-3A/AN FPGAs)**

Components	Packages (XX)	Components	Packages (XX)
MT47H64M4XX-3	BP	MT47H128M8XX-3	BT,HQ
MT47H64M4XX-37E	BP	MT47H128M8XX-37E	BT,HQ
MT47H64M4XX-5E	BP	MT47H128M8XX-5E	BT,HQ
MT47H128M4XX-3	B6,CB,GB	MT47H256M8XX-3	HG
MT47H128M4XX-37E	B6,CB,GB	MT47H256M8XX-37E	HG
MT47H128M4XX-5E	B6,CB,GB	MT47H256M8XX-5E	HG
MT47H256M4XX-3	BT,HQ	MT47H16M16XX-3	BG
MT47H256M4XX-37E	BT,HQ	MT47H16M16XX-37E	BG
MT47H256M4XX-5E	BT,HQ	MT47H16M16XX-5E	BG
MT47H512M4XX-3	HG	MT47H32M16XX-3	BN,CC,FN,GC
MT47H512M4XX-37E	HG	MT47H32M16XX-37E	BN,CC,FN,GC
MT47H512M4XX-5E	HG	MT47H32M16XX-5E	BN,CC,FN,GC
MT47H32M8XX-3	BP	MT47H64M16XX-3	BT,HR
MT47H32M8XX-37E	BP	MT47H64M16XX-37E	BT,HR
MT47H32M8XX-5E	BP	MT47H64M16XX-5E	BT,HR
MT47H64M8XX-3	B6,CB,F6,GB	MT47H128M16XX-3	HG
MT47H64M8XX-37E	B6,CB,F6,GB	MT47H128M16XX-37E	HG
MT47H64M8XX-5E	B6,CB,F6,GB	MT47H128M16XX-5E	--

**Table 8-10: Supported Unbuffered DIMMs for DDR2 SDRAM Local Clocking (Spartan-3A/AN FPGAs)**

Unbuffered DIMMs	
MT4HTF1664AY-667	MT8HTF6464AY-667
MT4HTF1664AY-40E	MT8HTF6464AY-53E
MT4HTF3264AY-667	MT8HTF6464AY-40E
MT4HTF3264AY-40E	MT8HTF12864AY-667
MT4HTF6464AY-667	MT8HTF12864AY-40E
MT4HTF6464AY-40E	

**Table 8-11: Supported Registered DIMMs for DDR2 SDRAM Local Clocking (Spartan-3A/AN FPGAs)**

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9HTF3272XX-53E	Y	MT9HTF6472XX-40E	Y
MT9HTF3272XX-40E	Y	MT9HTF12872XX-53E	PY,Y
MT9HTF6472XX-53E	Y	MT9HTF12872XX-40E	Y

**Table 8-12: Supported SODIMMs for DDR2 SDRAM Local Clocking (Spartan-3A/AN FPGAs)**

SODIMMs	
MT4HTF1664HY-667	MT8HTF3264HY-667
MT4HTF1664HY-53E	MT8HTF3264HY-53E
MT4HTF1664HY-40E	MT8HTF3264HY-40E
MT4HTF3264HY-667	MT8HTF6464HY-667
MT4HTF3264HY-53E	MT8HTF6464HY-53E
MT4HTF3264HY-40E	MT8HTF6464HY-40E

The tables below list the components (Table 8-13 and Table 8-16) and DIMMs (Table 8-14 through Table 8-15) supported by the tool for Spartan-3A DSP and Spartan-3E FPGA DDR2 local clocking designs.

**Table 8-13: Supported Components for DDR2 SDRAM Local Clocking (Spartan-3A DSP FPGAs)**

Components	Packages (XX)	Components	Packages (XX)
MT47H64M4XX-3	BP	MT47H128M8XX-3	BT,HQ
MT47H64M4XX-37E	BP	MT47H128M8XX-37E	BT,HQ
MT47H64M4XX-5E	BP	MT47H128M8XX-5E	BT,HQ
MT47H128M4XX-3	B6,CB,GB	MT47H256M8XX-3	HG
MT47H128M4XX-37E	B6,CB,GB	MT47H256M8XX-37E	HG
MT47H128M4XX-5E	B6,CB,GB	MT47H256M8XX-5E	HG
MT47H256M4XX-3	BT,HQ	MT47H16M16XX-3	BG
MT47H256M4XX-37E	BT,HQ	MT47H16M16XX-37E	BG
MT47H256M4XX-5E	BT,HQ	MT47H16M16XX-5E	BG
MT47H512M4XX-3	HG	MT47H32M16XX-3	BN,CC,FN,GC
MT47H512M4XX-37E	HG	MT47H32M16XX-37E	BN,CC,FN,GC
MT47H512M4XX-5E	HG	MT47H32M16XX-5E	BN,CC,FN,GC
MT47H32M8XX-3	BP	MT47H64M16XX-3	BT,HR
MT47H32M8XX-37E	BP	MT47H64M16XX-37E	BT,HR
MT47H32M8XX-5E	BP	MT47H64M16XX-5E	BT,HR
MT47H64M8XX-3	B6,CB,F6,GB	MT47H128M16XX-3	HG
MT47H64M8XX-37E	B6,CB,F6,GB	MT47H128M16XX-37E	HG
MT47H64M8XX-5E	B6,CB,F6,GB	MT47H128M16XX-5E	--

**Table 8-14: Supported Unbuffered DIMMs for DDR2 SDRAM Local Clocking (Spartan-3A DSP FPGAs)**

Unbuffered DIMMs	
MT4HTF1664AY-667	MT8HTF6464AY-667
MT4HTF1664AY-40E	MT8HTF6464AY-53E
MT4HTF3264AY-667	MT8HTF6464AY-40E
MT4HTF3264AY-40E	MT8HTF12864AY-667
MT4HTF6464AY-667	MT8HTF12864AY-40E
MT4HTF6464AY-40E	

**Table 8-15: Supported SODIMMs for DDR2 SDRAM Local Clocking (Spartan-3A DSP FPGAs)**

SODIMMs	
MT4HTF1664HY-667	MT8HTF3264HY-667
MT4HTF1664HY-53E	MT8HTF3264HY-53E
MT4HTF1664HY-40E	MT8HTF3264HY-40E
MT4HTF3264HY-667	MT8HTF6464HY-667
MT4HTF3264HY-53E	MT8HTF6464HY-53E
MT4HTF3264HY-40E	MT8HTF6464HY-40E

**Table 8-16: Supported Components for DDR2 SDRAM Local Clocking (Spartan-3E FPGAs)**

Components	Packages (XX)	Components	Packages (XX)
MT47H64M4XX-3	BP	MT47H128M8XX-3	BT,HQ
MT47H64M4XX-37E	BP	MT47H128M8XX-37E	BT,HQ
MT47H64M4XX-5E	BP	MT47H128M8XX-5E	BT,HQ
MT47H128M4XX-3	B6,CB,GB	MT47H256M8XX-3	HG
MT47H128M4XX-37E	B6,CB,GB	MT47H256M8XX-37E	HG
MT47H128M4XX-5E	B6,CB,GB	MT47H256M8XX-5E	HG
MT47H256M4XX-3	BT,HQ	MT47H16M16XX-3	BG
MT47H256M4XX-37E	BT,HQ	MT47H16M16XX-37E	BG
MT47H256M4XX-5E	BT,HQ	MT47H16M16XX-5E	BG
MT47H512M4XX-3	HG	MT47H32M16XX-3	BN,CC,FN,GC
MT47H512M4XX-37E	HG	MT47H32M16XX-37E	BN,CC,FN,GC
MT47H512M4XX-5E	HG	MT47H32M16XX-5E	BN,CC,FN,GC
MT47H32M8XX-3	BP	MT47H64M16XX-3	BT,HR
MT47H32M8XX-37E	BP	MT47H64M16XX-37E	BT,HR
MT47H32M8XX-5E	BP	MT47H64M16XX-5E	BT,HR
MT47H64M8XX-3	B6,CB,F6,GB	MT47H128M16XX-3	HG
MT47H64M8XX-37E	B6,CB,F6,GB	MT47H128M16XX-37E	HG
MT47H64M8XX-5E	B6,CB,F6,GB	MT47H128M16XX-5E	--

## Maximum Data Widths

Table 8-17 provides the maximum data widths for Spartan-3 FPGAs. Table 8-18 provides the maximum data widths for Spartan-3E FPGAs. Table 8-21 provides the maximum data widths for Spartan-3A single-ended DQS FPGAs (differential DQS is disabled). Table 8-22 provides the maximum data widths for Spartan-3A differential DQS FPGAs (differential DQS is enabled). Table 8-23 provides the maximum data widths for Spartan-3AN differential DQS FPGAs (single/differential DQS is enabled). Table 8-24 provides the maximum data widths for Spartan-3A DSP differential DQS FPGAs (single/differential DQS is enabled). All the supported data width tables have the Mask Enable option enabled.

Table 8-17: Spartan-3 FPGA Maximum Data Width for DDR and DDR2 Memories

Serial Number	FPGA	Maximum Data Width when Data, Address, and Control are Allocated in...										
		...Different Banks						...the Same Bank				
		Bank 2	Bank 3	Bank 6	Bank 7	Left	Right	Bank 2	Bank 3	Banks 6/7	Left	Right
1	XC3S50CP132	0	0	0	0	8	8	0	0	0	0	0
2	XC3S50PQ208	0	0	0	0	8	8	0	0	0	0	0
3	XC3S50TQ144	0	0	0	0	8	8	0	0	0	0	0
4	XC3S200FT256	8	8	8	8	16	16	0	0	0	8	8
5	XC3S200PQ208	0	8	0	0	16	16	0	0	0	0	0
6	XC3S200TQ144	0	0	0	0	8	8	0	0	0	0	0
7	XC3S400FG320	8	8	8	8	24	24	0	0	0	16	16
8	XC3S400FG456	16	8	16	8	32	24	0	0	0	16	16
9	XC3S400FT256	8	8	8	8	16	16	0	0	0	8	8
10	XC3S400PQ208	0	0	0	0	8	8	0	0	0	0	0
11	XC3S400TQ144	0	0	0	0	8	8	0	0	0	0	0
12	XC3S1000FG320	8	8	8	8	24	24	0	0	0	16	16
13	XC3S1000FG456	16	16	16	16	48	48	8	8	8	32	32
14	XC3S1000FG676	24	24	24	24	48	48	8	8	8	32	32
15	XC3S1000FT256	8	8	8	8	16	16	0	0	0	8	8
16	XC3S1500FG320	8	8	8	8	24	24	0	0	0	16	16
17	XC3S1500FG456	16	16	16	16	48	48	8	8	8	40	40
18	XC3S1500FG676	32	32	32	32	72	72	16	16	16	48	48
19	XC3S2000FG456	16	16	16	16	48	48	8	8	8	32	32
20	XC3S2000FG676	32	32	32	32	72	72	16	16	16	56	56
21	XC3S2000FG900	32	32	32	40	72	72	24	24	24	64	64
22	XC3S4000FG676	24	32	32	32	72	72	16	16	16	56	48
23	XC3S4000FG900	40	40	40	40	72	72	32	32	32	72	72
24	XC3S4000FG1156	48	48	48	48	72	72	32	32	32	72	72
25	XC3S5000FG676	24	24	24	32	64	64	16	16	16	48	48

Table 8-17: Spartan-3 FPGA Maximum Data Width for DDR and DDR2 Memories (Continued)

Serial Number	FPGA	Maximum Data Width when Data, Address, and Control are Allocated in...										
		...Different Banks						...the Same Bank				
		Bank 2	Bank 3	Bank 6	Bank 7	Left	Right	Bank 2	Bank 3	Banks 6/7	Left	Right
26	XC3S5000FGG676	24	24	24	32	64	64	16	16	16	48	48
27	XC3S5000FG900	40	40	40	40	72	72	32	32	32	72	72
28	XC3S5000FG1156	56	56	48	56	72	72	40	40	40	72	72

Table 8-18: Spartan-3E FPGA Maximum Data Width for DDR SDRAMs

Serial Number	FPGA	Maximum Data Width when Data, Address, and Control are Allocated in...		
		...Different Banks		...the Same Bank
		Left	Right	Left/Right
1	XC3S100ECP132	8	8	0
2	XC3S100ETQ144	8	8	0
3	XC3S250ECP132	8	0	0
4	XC3S250EFT256	16	16	0
5	XC3S250EPQ208	16	16	0
6	XC3S250ETQ144	8	8	0
7	XC3S500ECP132	8	0	0
8	XC3S500EFG320	24	24	8
9	XC3S500EFT256	16	16	8
10	XC3S500EPQ208	8	8	0
11	XC3S1200EFG320	16	16	16
12	XC3S1200EFG400	32	32	16
13	XC3S1200EFT256	16	8	8
14	XC3S1600EFG320	16	16	8
15	XC3S1600EFG400	24	32	16
16	XC3S1600EFG484	48	40	32

**Table 8-19: Spartan-3E FPGA Differential DQS Maximum Data Width for DDR SDRAMs (Differential DQS Enabled)**

Serial Number	FPGA	Maximum Data Width When Data, Address, and Control are Allocated in Different Banks	
		Left	Right
1	XC3S100ECP132	0	0
2	XC3S100ETQ144	0	0
3	XC3S250ECP132	0	0
4	XC3S250ETQ144	0	0
5	XC3S250EPQ208	8	16
6	XC3S250EFT256	8	16
7	XC3S500ECP132	0	0
8	XC3S500EPQ208	0	8
9	XC3S500EFT256	0	8
10	XC3S500EFG320	8	16
11	XC3S1200EFT256	0	8
12	XC3S1200EFG320	8	16
13	XC3S1200EFG400	32	32
14	XC3S1600EFG320	8	16
15	XC3S1600EFG400	16	16
16	XC3S1600EFG484	40	40

**Table 8-20: Spartan-3E FPGA Single-Ended DQS Maximum Data Width for DDR SDRAMs (Differential DQS Disabled)**

Serial Number	FPGA	Maximum Data Width When Data, Address, and Control are Allocated in Different Banks	
		Left	Right
1	XC3S100ECP132	8	8
2	XC3S100ETQ144	8	8
3	XC3S250ECP132	8	0
4	XC3S250ETQ144	8	8
5	XC3S250EPQ208	16	16
6	XC3S250EFT256	16	16
7	XC3S500ECP132	8	0
8	XC3S500EPQ208	8	8
9	XC3S500EFT256	16	16

**Table 8-20: Spartan-3E FPGA Single-Ended DQS Maximum Data Width for DDR SDRAMs (Differential DQS Disabled) (Continued)**

Serial Number	FPGA	Maximum Data Width When Data, Address, and Control are Allocated in Different Banks	
		Left	Right
10	XC3S500EFG320	24	24
11	XC3S1200EFT256	8	16
12	XC3S1200EFG320	16	16
13	XC3S1200EFG400	32	32
14	XC3S1600EFG320	16	16
15	XC3S1600EFG400	32	24
16	XC3S1600EFG484	40	48

**Table 8-21: Spartan-3A FPGA Single-Ended DQS Maximum Data Width (Differential DQS Disabled)**

Serial Number	FPGA	Maximum Data Width when Data, Address, and Control are Allocated in...		
		...Different Banks		...the Same Bank
		Left/Right	Left	Right
1	XC3S50ATQ144	8	0	0
2	XC3S50AFT256	8	0	0
3	XC3S200AFT256	16/24	8	8
4	XC3S400AFT256	16	8	8
5	XC3S200AFG320	16	8	16
6	XC3S400AFG320	24	8	16
7	XC3S400AFG400	32	16	16
8	XC3S700AFG400	32	16	16
9	XC3S700AFG484	40	24	32
10	XC3S1400AFG484	40	24	32
11	XC3S1400AFG676	72	48	48

**Table 8-22: Spartan-3A FPGA Differential DQS Maximum Data Width (Differential DQS Enabled)**

Serial Number	FPGA	Maximum Data Width when Data, Address, and Control are Allocated in...			
		...Different Banks		...the Same Bank	
		Left	Right	Left	Right
1	XC3S50ATQ144	8	8	0	0
2	XC3S200AFG320	24	24	8	16
3	XC3S200AFT256	16	24	8	8
4	XC3S400AFG320	24	24	8	16
5	XC3S400AFG400	32	32	16	16
6	XC3S400AFT256	16	16	8	8
7	XC3S700AFG400	24	32	16	16
8	XC3S700AFG484	40	40	24	32
9	XC3S1400AFG484	40	40	24	32
10	XC3S1400AFG676	64	64	48	48
11	XC3S50AFT256	8	8	0	0

**Table 8-23: Spartan-3AN FPGA DQS Maximum Data Width (Single/Differential DQS Enabled)**

Serial Number	FPGA	Maximum Data Width when Data, Address, and Control are Allocated in...			
		...Different Banks		...the Same Bank	
		Left	Right	Left	Right
1	XC3S50ANTQG144	8	8	0	0
2	XC3S50ANFTG256	8	8	0	0
3	XC3S200ANFTG256	16	24	8	8
4	XC3S400ANFGG400	32	32	16	16
5	XC3S700ANFGG484	40	40	24	32
6	XC3S1400ANFGG676 <sup>(1)</sup>	64	64	48	48

**Notes:**

1. For the XC3S1400ANFGG676 part, MIG can generate 72-bit single-ended DQS RDIMM with address and data on different banks.



**Table 8-24: Spartan-3A DSP FPGA DQS Maximum Data Width (Single/Differential DQS Enabled)**

Serial Number	FPGA	Maximum Data Width when Data, Address, and Control are Allocated in...			
		...Different Banks		...the Same Bank	
		Left	Right	Left	Right
1	XC3SD1800A-CS484	32	32	16	16
2	XC3SD3400A-CS484	32	32	16	16
3	XC3SD1800A-FG676	64	64	48	48
4	XC3SD3400A-FG676	64	64	48	48

## DIMM Support for Spartan-3 Generation Devices

**Table 8-25: DIMM Support for Spartan-3 Devices**

Serial Number	FPGA	64-bit DIMM			72-bit DIMM		
		x4	x8	x16	x4	x8	x16
1	XC3S1500FG676	No	Yes	Yes	No	Yes	Yes
2	XC3S2000FG676	No	Yes	Yes	No	Yes	Yes
3	XC3S4000FG676	No	Yes	Yes	No	Yes	Yes
4	XC3S5000FG676	No	Yes	Yes	No	No	No
5	XC3S2000FG900	Yes	Yes	Yes	Yes	Yes	Yes
6	XC3S4000FG900	Yes	Yes	Yes	Yes	Yes	Yes
7	XC3S5000FG900	Yes	Yes	Yes	Yes	Yes	Yes
8	XC3S4000FG1156	Yes	Yes	Yes	Yes	Yes	Yes
9	XC3S5000FG1156	Yes	Yes	Yes	Yes	Yes	Yes
10	XC3S1500LFG676	No	Yes	Yes	No	Yes	Yes
11	XC3S4000LFG900	Yes	Yes	Yes	Yes	Yes	Yes

**Table 8-26: DIMM Support for Spartan-3A and Spartan-3AN Devices**

Serial Number	FPGA	64-bit DIMM			72-bit RDIMM		
		x4	x8	x16	x4	x8	x16
1	XC3S1400AFG676	No	Yes	Yes	No	Yes	Yes
2	XC3S1400ANFGG676	No	Yes	Yes	No	Yes	Yes

Table 8-27: DIMM Support for Spartan-3A DSP Devices

Serial Number	FPGA	64-bit DIMM			72-bit RDIMM		
		x4	x8	x16	x4	x8	x16
1	XC3SD1800AFG676	No	Yes	Yes	No	No	No
2	XC3SD3400AFG676	No	Yes	Yes	No	No	No

**Note:** Spartan-3E devices do not support 64-bit or 72-bit DIMMs.

## Design Frequency Range in MHz for Spartan-3 Generation Devices

Table 8-28: Spartan-3 Generation Component Controllers

FPGA Family	DDR SDRAM		DDR2 SDRAM	
	≤ 32-bit	> 32-bit	≤ 32-bit	> 32-bit
Spartan-3A/3AN/3A DSP	166	166	166	166
Spartan-3E	166	166	166	166
Spartan-3	166	133	166	133

Table 8-29: Spartan-3 Generation DIMM Controllers

FPGA Family	DDR SDRAM		DDR2 SDRAM	
Spartan-3A/3AN/3A DSP	166	166	166	166
Spartan-3E	NS	NS	NS	NS
Spartan-3	133	133	133	133

**Note:** NS = Not Supported.

## Hardware Tested Configurations

The frequencies shown in [Table 8-30](#) were achieved on the Spartan-3A FPGA Starter Kit under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based on a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 64-bit interface.

**Table 8-30: Hardware Tested Configurations for Spartan-3A FPGA DDR2 SDRAM Designs**

Synthesis Tools	XST
HDL	Verilog and VHDL
FPGA Device	XC3S700AFG484-4
Burst Lengths	4 and 8
CAS Latency (CL)	3
16-bit Design	Tested on 16-bit Component "MT47H32M16XX-5E"
Frequency Range	25 MHz to 225 MHz

The frequency shown in [Table 8-31](#) was achieved on the Spartan-3A DSP 3400A Development Board under nominal conditions. This frequency should not be used to determine the design frequency. The maximum design frequency supported in the MIG wizard is based a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 64-bit interface.

**Table 8-31: Hardware Tested Configurations for Spartan-3A DSP FPGA DDR2 SDRAM Designs**

Synthesis Tools	XST
HDL	Verilog and VHDL
FPGA Device	XC3SD3400AFG676-4
Burst Lengths	4 and 8
CAS Latency (CL)	3
32-bit Design	Tested on 64-bit SO DIMM "MT4HTF6464HY-667"
Frequency	133 MHz





## *Section IV: Virtex-5 FPGA to Memory Interfaces*

*Chapter 9, "Implementing DDR2 SDRAM Controllers"*

*Chapter 10, "Implementing QDRII SRAM Controllers"*

*Chapter 11, "Implementing DDR SDRAM Controllers"*



## Implementing DDR2 SDRAM Controllers

This chapter describes how to implement DDR2 SDRAM interfaces for Virtex™-5 FPGAs generated by MIG. The DDR2 SDRAM design supports frequencies up to 333 MHz. This design is based on XAPP858 [Ref 26].

### 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 9-1. 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.

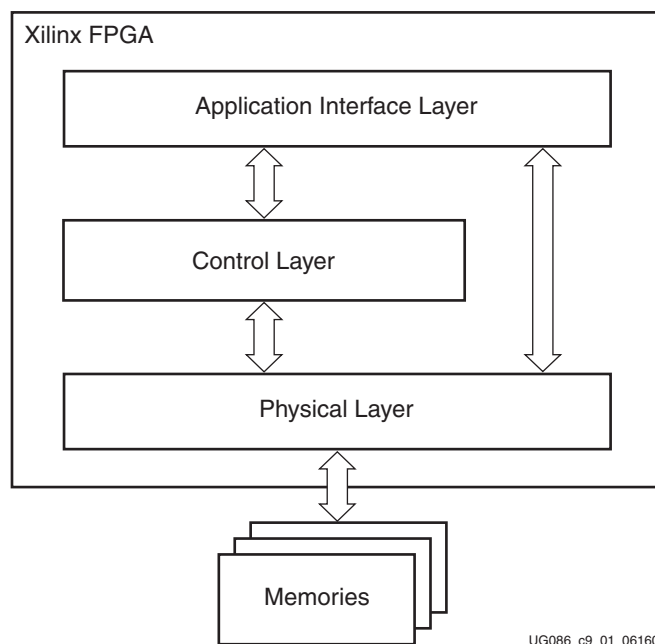


Figure 9-1: Modular Memory Interface Representation

## Feature Summary

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

### Supported Features

The DDR2 SDRAM controller design supports:

- Burst lengths of four and eight
- Sequential and interleaved burst types
- CAS latencies of 3, 4, and 5
- Additive latencies of 0, 1, 2, 3, and 4
- Differential DQS
- ODT
- Verilog and VHDL
- Byte wise data masking
- Precharge and auto refresh
- Bank management
- Linear addressing
- ECC
- Different memories (density/speed)
- Memory components, registered DIMMs, unbuffered DIMMs, and SODIMMs
- With and without a testbench
- With and without a DCM

The supported features are described in more detail in [“Architecture.”](#)

### Design Frequency Ranges

Table 9-1: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-1		-2		-3	
	Min	Max	Min	Max	Min	Max
Component	125	266	125	300	125	333
RDIMM	125	266	125	300	125	333
UDIMM or SODIMM <sup>(1)</sup>	125	266	125	266	125	266

**Notes:**

1. It is possible to go faster than 266 MHz, but it requires care and IBIS simulations and possibly using the parameter to send the CS out earlier depending on the load. For more details, see XAPP858 [Ref 26].

### Unsupported Features

The DDR2 SDRAM controller design does not support:

- Dual-rank DIMMs
- Single-ended DQS



- Redundant DQS (RDQS)
- Deep memories
- Multicontrollers

## Architecture

### Implemented Features

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

#### Burst Length

The DDR2 SDRAM controller supports burst lengths of four and eight. Through the “Set mode register(s)” option, the burst length can be selected. For a design without a testbench (user\_design), the user has to provide bursts of the input data based on the chosen burst length. Bits M2:M0 of the Mode Register define the burst length, and bit M3 indicates the burst type (see the Micron data sheet). Read and write accesses to the DDR2 SDRAM are burst-oriented. It determines the maximum number of column locations accessed for a given READ or WRITE command.

#### CAS Latency

The DDR2 SDRAM controller supports CAS latencies of 3, 4, and 5. The CAS latency (CL) can be selected in the “Set mode register(s)” option. CL is implemented in the phy\_write module. During data write operations, the generation of the dqs\_oe\_n and dqs\_rst\_n signals varies according to the CL in the phy\_write module. During read data operations, the generation of the ctrl\_rden signal varies according to the CL in the ctrl module. Bits M4:M6 of the Mode Register define the CL (see the Micron data sheet). CL is the delay in clock cycles between the registration of a READ command and the availability of the first bit of output data.

#### Additive Latency

DDR2 SDRAM devices support a feature called posted CAS additive latency (AL). The DDR2 SDRAM supports ALs of 0, 1, 2, 3, and 4. AL can be selected in the “Set mode register(s)” option. AL is implemented in the DDR2 SDRAM ctrl module. The ctrl module issues READ/WRITE commands prior to  $t_{RCD}$  (minimum) depending on the user-selected AL value in the Extended Mode Register. 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 (see the Micron data sheet).

#### Data Masking

DDR2 SDRAM design supports data masking per byte. Masking per nibble is not supported due to the limitation of the internal block RAM based FIFOs. So, the masking of data can be done on per byte basis. The mask data is stored in the Data FIFO along with the actual data.

## Precharge

The PRECHARGE command is used to close the open row in a bank if there is a command to be issued in the same bank. The DDR2 Virtex-5 controller issues a PRECHARGE command only if there is already an open row in the particular bank where a read or write command is to be issued, thus increasing the efficiency of the design. The auto precharge function is not supported in this design. The design ties the A10 bit Low during normal reads and writes.

## Auto Refresh

The auto refresh command is issued to the memory at specified intervals of time. The memory issues an auto refresh command to refresh the charge to retain the data.

## Bank Management

A Virtex-5 DDR2 SDRAM controller design supports bank management that increases the efficiency of the design. The controller keeps track of whether the bank being accessed already has an open row or not and also decides whether a PRECHARGE command should be issued or not to that bank. When bank management is enabled via the MULTI\_BANK\_EN parameter, a maximum of four banks/rows can open at any one time. A least recently used (LRU) algorithm is employed to keep the three most recently used banks and to close the least recently used bank when a new bank/row location needs to be accessed. The bank management feature can also be disabled by clearing MULTI\_BANK\_EN.

## Linear Addressing

The DDR2 SDRAM controller supports linear addressing. Linear addressing refers to the way the user provides the address of the memory to be accessed. For Virtex-5 DDR2 SDRAM controllers, the user provides the address information through the app\_af\_addr signal. As the densities of the memory devices vary, the number of column address bits and row address bits also change. In any case, the row address bits in the app\_af\_addr signal always start from the next higher bit, where the column address ends. This feature increases the number of devices that can be supported with the design.

## Different Memories (Density/Speed)

The DDR2 SDRAM controller supports different densities. For DDR2 components shown in MIG, densities vary from 256 Mb to 2 Gb, and the DIMM densities vary from 256 Mb to 2 Gb. The user can select the various configurations from the “Create new memory part” option. The supported maximum column address is 13, the maximum row address is 15, and the maximum bank address is 3. The design can decode write and read addresses from the user in the DDR2 SDRAM ctrl module. The user address consists of column, row, and bank addresses.

## On-Die Termination

The DDR2 SDRAM controller supports on-die termination (ODT). Through the “Set mode register(s)” option from the GUI, the user can disable ODT or can choose 75, 150, or 50. ODT can turn the termination on and off as needed to improve signal integrity in the system.

ODT is only enabled on writes to DDR2 memory. It is disabled on read operations.

**Note:** The Virtex-5 DDR2 interface requires that if parallel termination is used at the memory end, it must be ODT rather than external termination resistor(s). This is a requirement of the read capture scheme used.

## Generic Parameters

The DDR2 SDRAM design is a generic design that works for most of the features mentioned above. User input parameters are defined as parameters for Verilog and generics in VHDL in the design modules and are passed down the hierarchy. For example, if the user selects a burst length of 4, then it is defined as follows in the <top\_module> module:

```
parameter BURST_LEN = 4,           // burst length (in doublewords)
```

The user can change this parameter in <top\_module> for various burst lengths to get the desired output. Same concept holds for all the other parameters listed in the <top\_module> module. [Table 9-2](#) lists the details of all parameters.

Table 9-2: Parameterization of DDR2 SDRAM Virtex-5 Design

Category	Parameter Name	Description	Other Notes	Value Restrictions
Memory Width	BANK_WIDTH	Number of memory bank address bits		
	CKE_WIDTH	Number of memory clock enable outputs		
	CLK_WIDTH	Number of differential clock outputs	Determined by the number of components/modules (one pair per component)	
	COL_WIDTH	Number of memory column bits		
	CS_BITS	$\log_2(\text{CS\_NUM})$	Used for chip-select related address decode. See notes for CS_NUM and CS_WIDTH.	
	CS_NUM	Number of separate chip selects	Different from CS_WIDTH. For example, for a 32-bit data bus consisting of 2 x16 parts, CS_NUM = 1, but CS_WIDTH = 2 (that is, a single chip select drives two separate outputs, one for each component)	$\text{CS\_WIDTH} / \text{CS\_NUM} = \text{integer}$
	CS_WIDTH	Number of memory chip selects	Determined by the number of components/modules (one per component)	$\text{CS\_WIDTH} / \text{CS\_NUM} = \text{integer}$
	DM_WIDTH	Number of data mask bits	Can be different value than DQS_WIDTH if x4 components are used	$(\text{DQS\_WIDTH} * \text{DQ\_PER\_DQS}) / 8$
	DQ_BITS	$\log_2(\text{DQS\_WIDTH} * \text{DQ\_PER\_DQS})$	Used for data bus calibration decode	$(\text{DQ\_WIDTH}) / \text{Number of data bits}$
	DQ_WIDTH	Number of data bits	Must set to $\text{DQS\_WIDTH} * \text{DQ\_PER\_DQS}$ . Equal to total number of data bits, including ECC bits.	$\text{DQS\_WIDTH} * \text{DQ\_PER\_DQS}$
	DQ_PER_DQS	Number of memory DQ data bits per strobe		
	DQS_BITS	$\log_2(\text{DQS\_WIDTH})$		
	DQS_WIDTH	Number of memory DQS strobes		
	ODT_WIDTH	Number of ODT control outputs	Determined by the number of components/modules (one per component)	
	ROW_WIDTH	Number of memory address bits		
APPDATA_WIDTH	Number of data bits at user backend interface			If ECC Disabled: $2 * (\text{DQ\_WIDTH})$ If ECC Enabled: $2 * (\text{DQ\_WIDTH} - 8 * (\text{DQ\_WIDTH} / 72))$

Table 9-2: Parameterization of DDR2 SDRAM Virtex-5 Design (Continued)

Category	Parameter Name	Description	Other Notes	Value Restrictions
Memory Options	ADDITIVE_LAT	Additive latency		(0,1,2,3,4)
	BURST_LEN	Burst length		(4,8) for DDR2, (2,4,8) for DDR
	BURST_TYPE	burst type (0: sequential, 1: interleaved)		(0,1)
	CAS_LAT	CAS latency (equal to 6 for CL = 2.5)		(3,4,5) for DDR2, (2,3,6) for DDR
	ECC_ENABLE	Enable ECC		Set to 0
	MULTI_BANK_EN	Bank management enable	If enabled, up to 4 banks are kept open; otherwise, one bank is kept open	(0,1)
	ODT_TYPE	ODT termination value	0: ODT disabled 1: 75 $\Omega$ 2: 150 $\Omega$ 3: 50 $\Omega$	(0,1,2,3)
	REDUCE_DRV	Reduced strength memory I/O enable. Set (1) for reduced I/O drive strength.	Not supported for all DDR/DDR2 widths	(0,1)
	REG_ENABLE	Set for registered memory module	Accounts for an extra clock cycle delay on address/control for registered module	(0,1)
	TWO_T_TIME_EN	Enable "2T" timing for control/address signals	0: Disable 2T timing 1: Enable 2T timing	(0,1)
Memory Timing	TREFI_NS	Auto refresh interval (in ns)	Take directly from memory datasheet	
	T <sub>RAS</sub>	Active to precharge delay (in ps)	Take directly from memory datasheet	
	T <sub>RCD</sub>	Active to read/write delay (in ps)	Take directly from memory datasheet	
	T <sub>RFC</sub>	Refresh to refresh, refresh to active delay (in ps)	Take directly from memory datasheet	
	T <sub>RP</sub>	Precharge to command delay (in ps)	Take directly from memory datasheet	
	T <sub>RTP</sub>	Read to precharge delay (in ps)	Take directly from memory datasheet	
	T <sub>WR</sub>	Used to determine write to precharge (in ps)	Take directly from memory datasheet	
	T <sub>WTR</sub>	Write to read (in ps)	Take directly from memory datasheet	

Table 9-2: Parameterization of DDR2 SDRAM Virtex-5 Design (Continued)

Category	Parameter Name	Description	Other Notes	Value Restrictions
Miscellaneous	CLK_PERIOD	Memory clock period (in ps)	Used for PHY calibration and DCM (if applicable) setting	
	DLL_FREQ_MODE	DCM Frequency Mode	Determined by CLK_PERIOD. Needed only if the DCM option is selected.	("LOW", "HIGH")
	DDR2_TYPE	Select either DDR or DDR2 interface	0: DDR 1: DDR2 Provided from the mem_if_top level and below	(0,1)
	SIM_ONLY	Enable to bypass initial 200 µs power-on delay. Abbreviated calibration sequence (only one bit for Stage 1, one strobe for Stages 2–4).		(0,1)
	RST_ACT_LOW	Indicates the polarity of input reset signal (sys_rst_n)	1: Reset is active Low. 0: Reset is active High.	(0,1)
	DQS_IO_COL	Placement parameter specifying I/O column locations for each DQS in interface	For each DQS, set to: 00: Left 01: Center 10: Right	Array size = 2 * DQS_WIDTH. Each array element must be = (00, 01, 10)
	DQ_IO_MS	Placement parameter specifying master/slave I/O placement for each DQ in interface	For each DQ, set to: 0: Slave I/O used 1: Master I/O used	Array size = DQ_WIDTH. Each array element must be = (0, 1)
	DEBUG_EN	Enable Calibration Debug Port	See Appendix D for details	(0,1)

## Hierarchy

Figure 9-2 shows the hierarchical structure of the DDR2 SDRAM design generated by MIG with a testbench and a DCM.

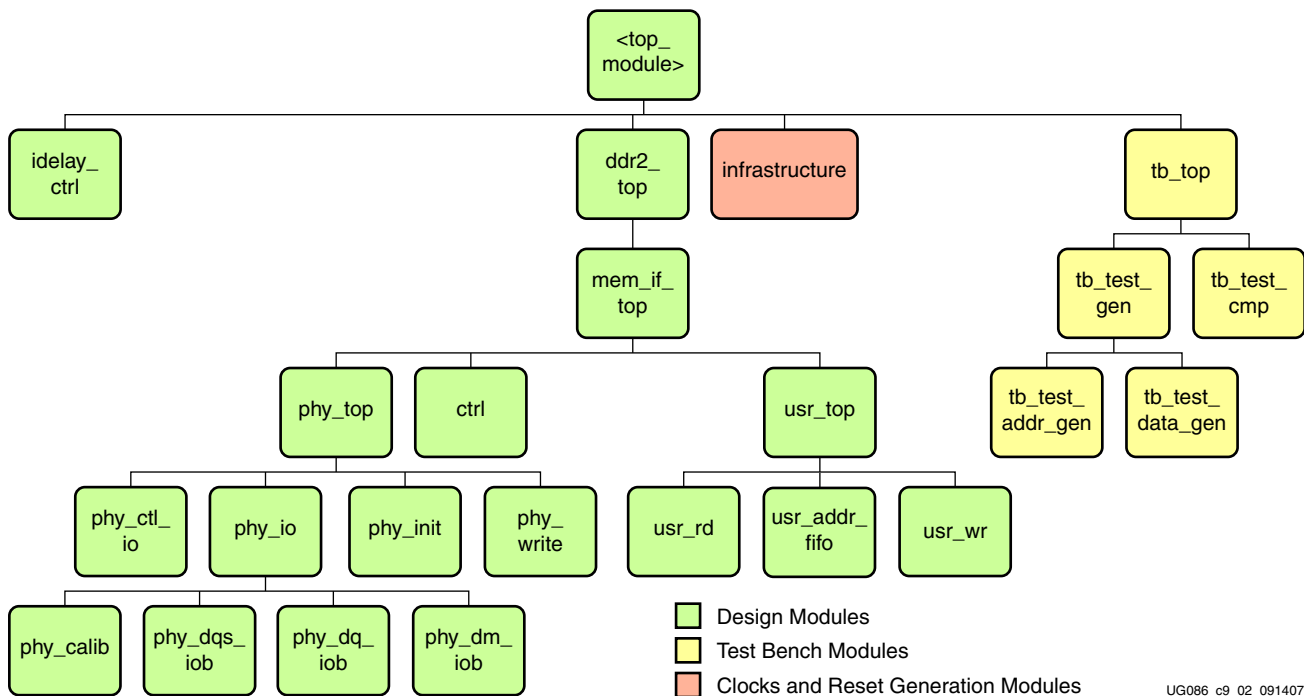


Figure 9-2: Hierarchical Structure of the Virtex-5 DDR2 Design

UG086\_c9\_02\_091407

The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks and reset generation modules

MIG can generate four different DDR2 SDRAM designs:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

For a design without a testbench (user\_design), the shaded modules in [Figure 9-2](#) are not present in the design. The <top\_module> module has the user interface signals for designs without a testbench. The list of user interface signals is provided in [Table 9-5](#).

Design clocks and resets are generated in the infrastructure module. The DCM is instantiated in infrastructure module when selected by MIG. The inputs to this module are the differential design clock and a 200 MHz differential clock for the IDELAYCTRL module. A user reset is also input to this module. Using the input clocks and reset signals, system clocks and system reset are generated in this module which is used in the design.

The DCM primitive is not instantiated in this module if the **No DCM** option is selected. So, the system operates on the user-provided clocks. The system reset is generated in the infrastructure module using the dcm\_lock input signal.

## Constraints

The Virtex-5 FPGA DDR2 design uses a combination of the IOB flop (IDDR) and fabric-based flops for read data capture. This requires the use of pinout-dependent directed-routing and location constraints. For more details, see [Appendix B, "Required UCF and HDL Modifications for Pinout Changes."](#)

## MIG Tool Design Options

MIG provides various options to generate the design with or without a testbench or with or without a DCM. This section provides detailed descriptions of the type of design generated by the user using various options.

Figure 9-3 shows a top-level block diagram of a DDR2 SDRAM design with a DCM and a testbench. `sys_clk_p` and `sys_clk_n` are differential input system clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. `clk200_p` and `clk200_n` are used for the `idelay_ctrl` element. `sys_rst_n` is an active-Low system reset signal. All design resets are generated using it. The error output signal indicates whether the case passes or fails. The testbench module does writes and reads, and also compares the read data with written data. The error signal is driven High on data mismatches. The `phy_init_done` signal indicates the completion of initialization and calibration of the design.

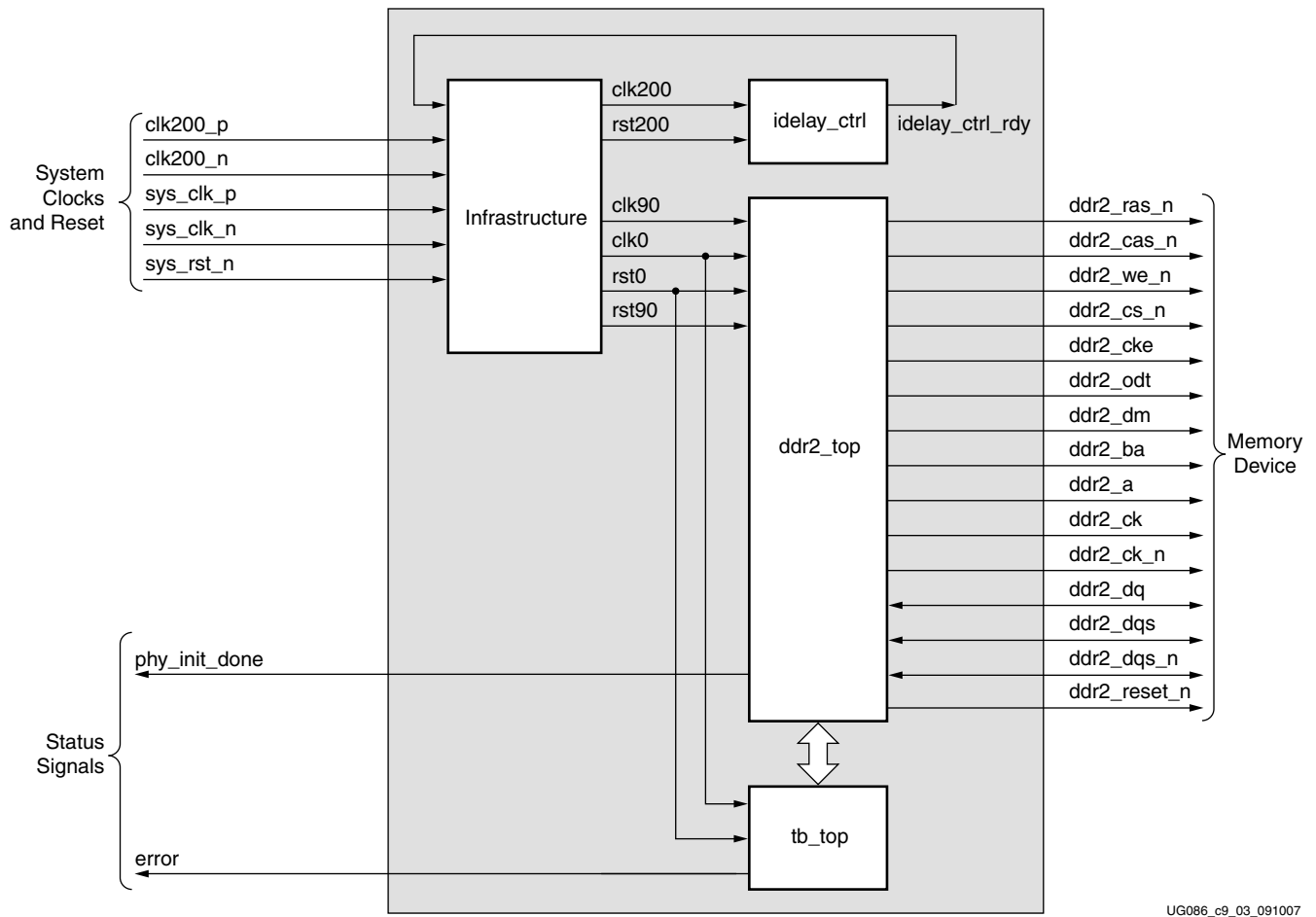
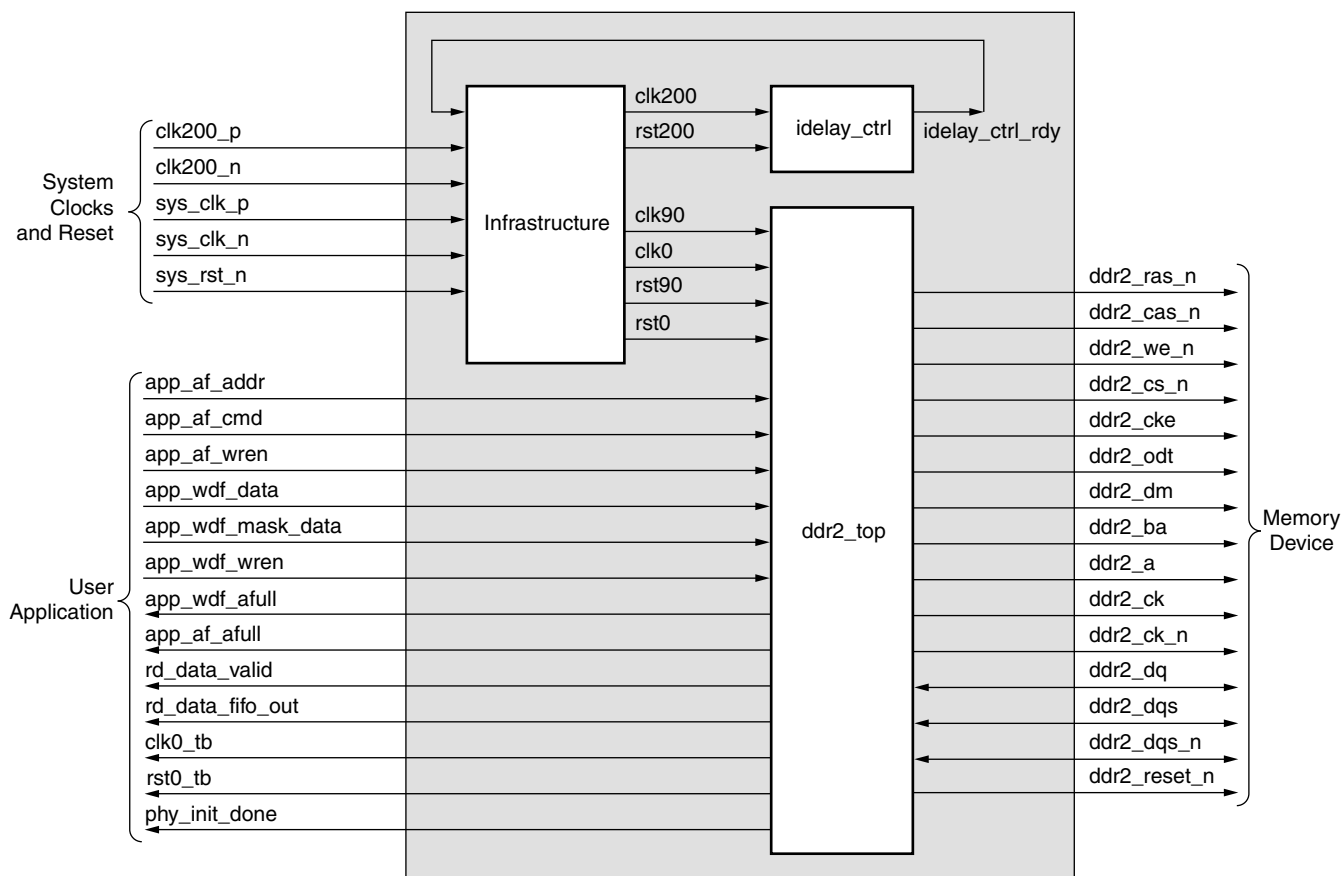


Figure 9-3: Top-Level Block Diagram of the DDR2 SDRAM Design with a DCM and a Testbench



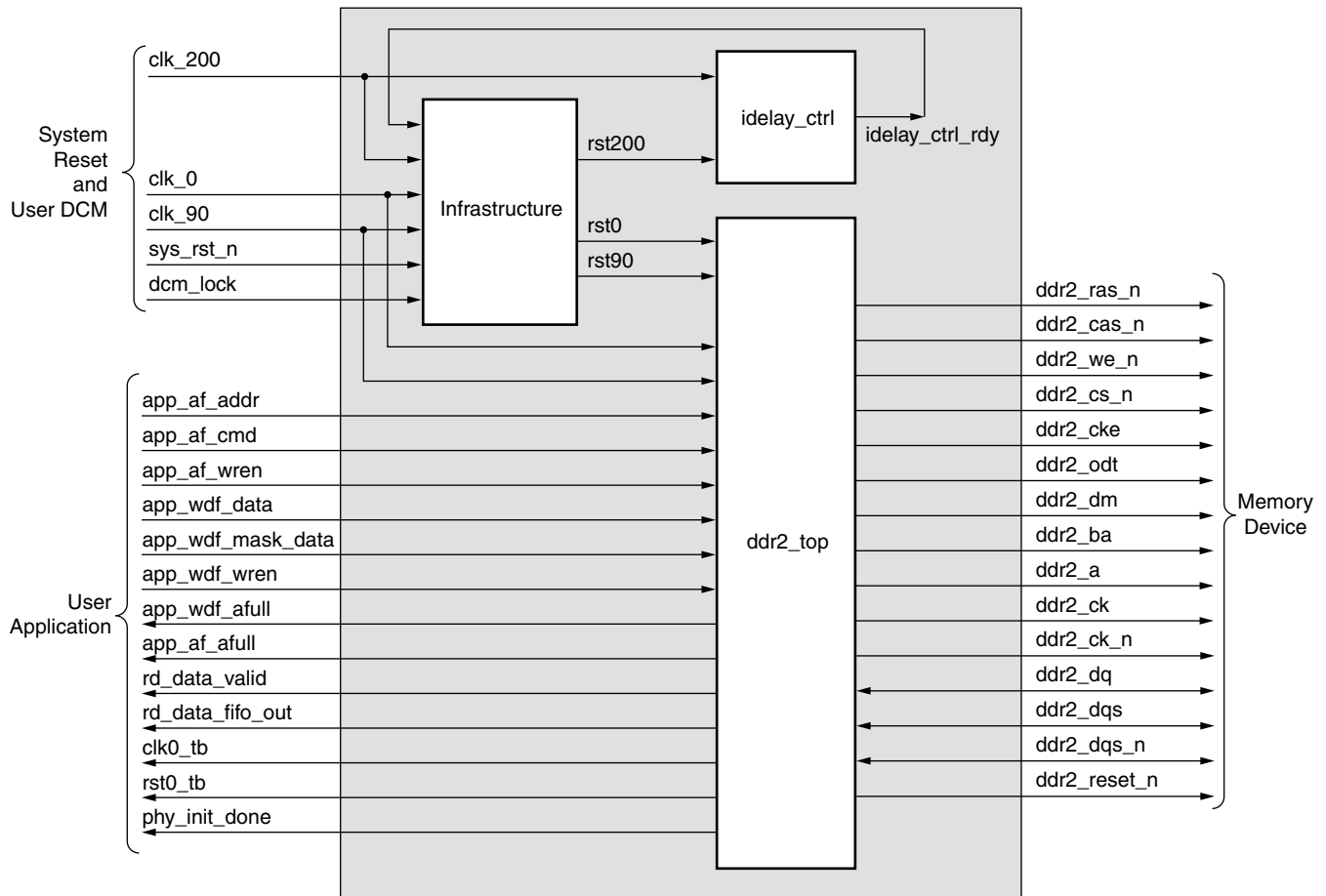
Figure 9-4 shows a top-level block diagram of a DDR2 SDRAM design with a DCM but without a testbench. The `sys_clk_p` and `sys_clk_n` signals are differential input system clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. The `clk200_p` and `clk200_n` signals are used for the `idelay_ctrl` element. The `sys_rst_n` signal is the active-Low system reset signal. All design resets are gated by the `dcm_lock` signal. The user has to drive the user application signals. The design provides the `clk_tb` and `reset_tb` signals to the user in order to synchronize with the design. The `phy_init_done` signal indicates the completion of initialization and calibration of the design.



UG086\_c9\_04\_091007

Figure 9-4: Top-Level Block Diagram of the DDR2 SDRAM Design with a DCM but without a Testbench

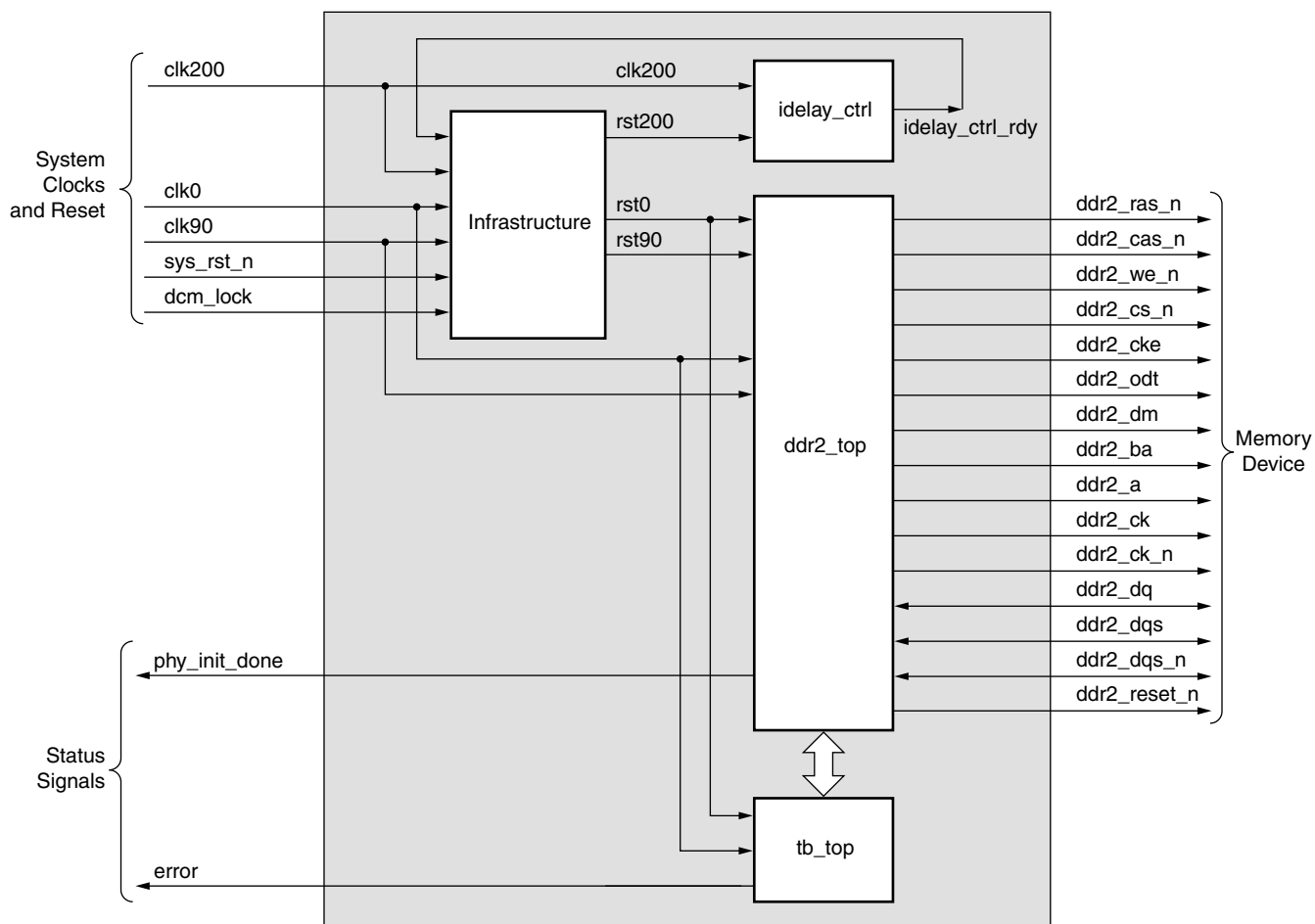
Figure 9-5 shows a top-level block diagram of a DDR2 SDRAM design without a DCM or a testbench. The user should provide all the clocks and the dcm\_lock signal. These clocks should be single-ended. The sys\_rst\_n signal is the active-Low system reset signal. All design resets are gated by the dcm\_lock signal. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFs. The user has to drive the user application signals. The design provides the clk\_tb and reset\_tb signals to the user in order to synchronize with the design. The phy\_init\_done signal indicates the completion of initialization and calibration of the design.



UG086\_e9\_05\_091007

Figure 9-5: Top-Level Block Diagram of the DDR2 SDRAM Design without a DCM or a Testbench

Figure 9-6 shows a top-level block diagram of a DDR2 SDRAM design without a DCM but with a testbench. The user should provide all the clocks and the `dcm_lock` signal. These clocks should be single-ended. `sys_rst_n` is the active-Low system reset signal. All design resets are gated by the `dcm_lock` signal. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFPGs. The error output signal indicates whether the case passes or fails. The testbench module does writes and reads, and also compares the read data with the written data. The error signal is driven High on data mismatches. The `phy_init_done` signal indicates the completion of initialization and calibration of the design.



UG086\_c9\_06\_013107

Figure 9-6: Top-Level Block Diagram of the DDR2 SDRAM Design without a DCM but with a Testbench

## DDR2 Controller Submodules

Figure 9-7 is a detailed block diagram of the DDR2 SDRAM controller. The design top module is expanded to show various internal blocks. The functions of these blocks are explained in the subsections following the figure.

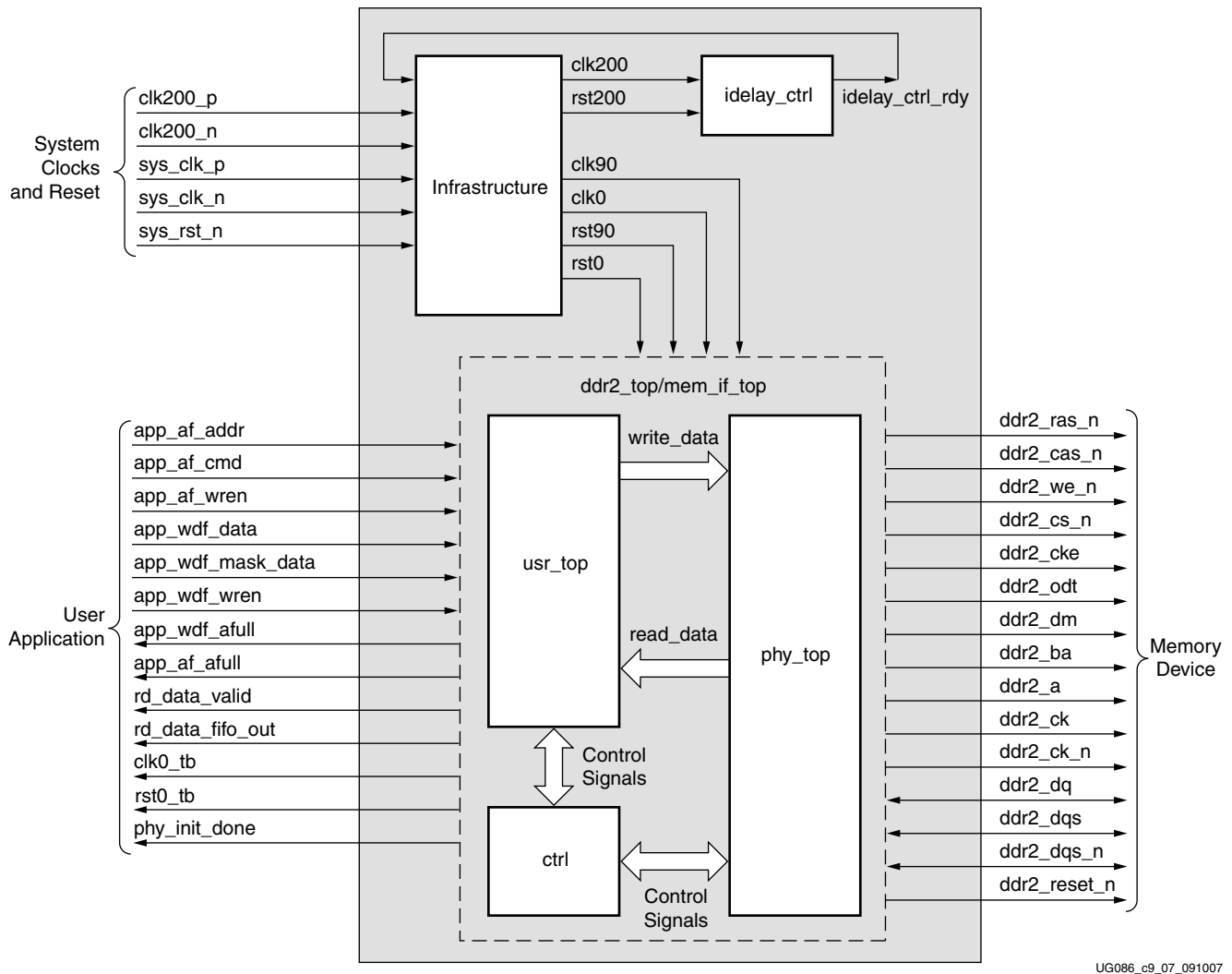


Figure 9-7: DDR2 Memory Controller Block Diagram

### Infrastructure

The infrastructure module generates the clock and reset signals for the design. The user clocks and user reset are input to this module. In designs generated with a DCM, the input clocks are differential. There are clocks for design use and also a 200 MHz clock for the idelayctrl primitive. These differential clocks are first passed through the buffers, and the single-ended output of the buffers is used. The single-ended output of sys\_clk\_p and sys\_clk\_n is then given to the DCM input. The clock outputs of the DCM are clk0 and clk90. After the DCM is locked, the design is in the reset state for at least 25 clocks.

For designs without a DCM, the user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFs.

## ldelay\_ctrl

This module instantiates the IDELAYCTRL primitive of the Virtex-5 FPGA. The IDELAYCTRL primitive is used to continuously calibrate the individual delay elements in its region to reduce the effect of process, temperature, and voltage variations. A 200 MHz clock has to be fed to this primitive.

## Ctrl

The ctrl module is the main controller of the Virtex-5 DDR2 SDRAM controller design. It generates all the control signals required for the DDR2 memory interface and the user interface. During the normal operation, this module toggles the memory address and control signals.

The ctrl module decodes the user command and issues the specified command to the memory. The app\_af\_cmd signal is decoded as a write command when it equals 3'b000, and app\_ar\_cmd is decoded as a read command when it equals 3'b001. The commands and control signals are generated based on the input burst length and CAS latency. The controller state machine issues the commands in the correct sequence while determining the timing requirements of the memory.

In the multi-bank mode (MULTIBANK\_EN = 1), the controller has the ability to keep four banks open at a time. The banks are opened in the order of the commands that are presented to the controller. In the event that four banks are already opened and an access arrives to the fifth bank, the least recently used bank is closed and the new bank is opened. All the banks are closed during auto refresh and are opened as commands are presented to the controller. Depending on the traffic pattern, the multi-bank enable mode can increase the efficiency of the design.

In the single-bank mode (MULTIBANK\_EN = 0), the controller keeps one bank open at a time. When there is an access to a different bank or to a different row in the current bank, the controller closes the current row and bank and opens the new row and bank.

## phy\_top

The phy\_top module is the top level of the physical interface of the design. The physical layer includes the input/output blocks (IOBs) and other primitives used to read and write the double data rate signals to and from the memory, such as IDDR and ODDR. This module also includes the IODELAY elements of the Virtex-5 FPGA. These IODELAY elements are used to delay the data signals to capture the valid data into the Read Data FIFO.

The memory control signals, such as RAS\_N, CAS\_N, and WE\_N, are driven from the buffers in the IOBs. All the input and output signals to and from the memory are referenced from the IOB to compensate for the routing delays inside the FPGA.

The phy\_init module, which is instantiated in the phy\_top module, is used to initialize the DDR2 memory in a predefined sequence according to the JEDEC standard for DDR2 SDRAM.

The phy\_calib module calibrates the design to align the strobe signal such that it always captures the valid data in the FIFO. This calibration is needed to compensate for the trace delays between the memory and the FPGA devices.

The phy\_write module splits the user data into rise data and fall data to be sent to the memory as a double data rate signal using ODDR. Similarly, while reading the data from memory, the data from IDDR is combined to get a single vector that is written into the read FIFO.

usr\_top

The `usr_top` module is the user interface block of the design. It receives and stores the user data, command, and address information in respective FIFOs. The `ctrl` module generates the required control signals for this module. During a write operation, the data stored in the `usr_wr_fifo` is read and given to the physical layer to output to the memory. Similarly, during a read operation, the data from the memory is read via IDDR and written into the FIFOs. This data is given to the user with a valid signal (`rd_data_valid`), which indicates valid data on the `rd_data_fifo_out` signal. Table 9-3 lists the user interface signals.

## DDR2 SDRAM Initialization

DDR2 memory is initialized through a specified sequence as per both Micron and JEDEC specifications. Initialization logic is implemented in the physical layer.

## DDR2 SDRAM Design Calibration

Before issuing user read and write commands, the read data path is calibrated to ensure that correct data is captured into the CLK0 domain of the FPGA. Calibration logic is implemented in the physical layer of the design. Figure 9-8 shows overall calibration sequence.

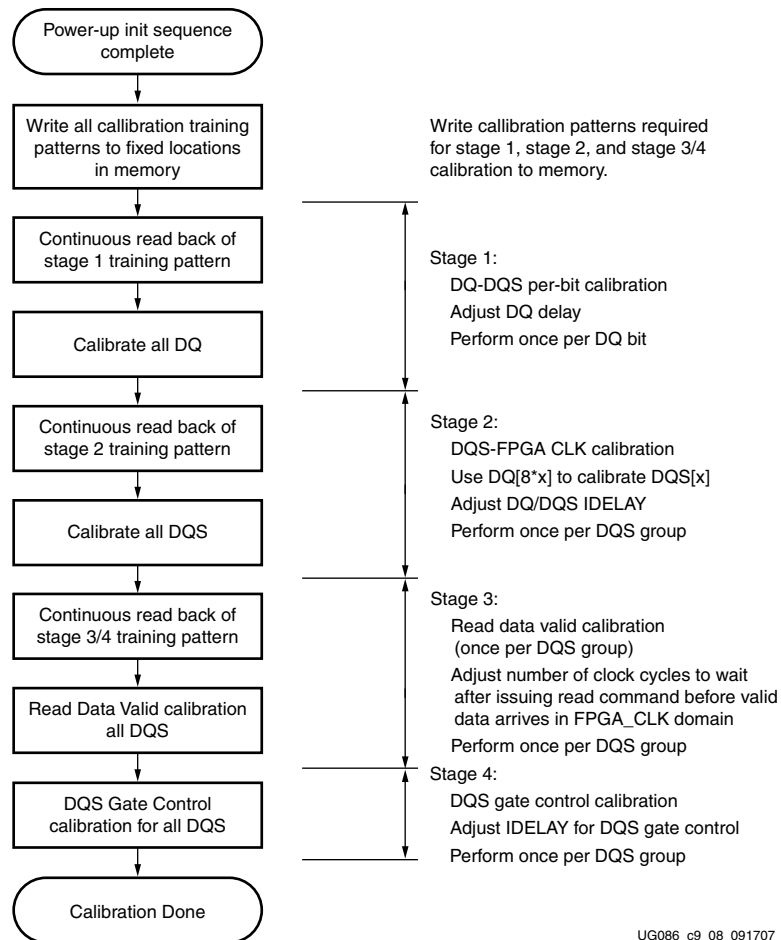


Figure 9-8: Overall Design Calibration Sequence

The first calibration stage is used to position the DQS in the DQ valid window. This synchronizes the capture of DQ using DQS in the IDDR flop. A training pattern of 1 for rise and 0 for fall data is written into the memory and is continuously read back. The DQ and IDELAYs are adjusted depending upon the DQ to DQS relationship. Per-bit deskew is performed on the DQ bits.

The second calibration stage is between the DQS and the FPGA clock. This synchronizes the transfer of data between the IDDR flop and flip-flops located in the FPGA fabric. The DQ and DQS IDELAY taps are incremented together to align to the FPGA clock domain.

The third calibration stage is the read-enable calibration, which is used to generate a read valid signal. The memory devices do not provide a signal indicating when the read data is valid. The read data is delayed by CAS latency, additive latency, the PCB trace, and the I/O buffer delays. The read-enable calibration is used to determine the delay between issuing a read command and the arrival of the read data.

The fourth calibration stage is used to align the DQS Gate signal from the controller to the falling edge of DQS. The DQS Gate controls the clock enable to the DQ IDDRs. It is used to prevent clocking of invalid data into the IDDR after the read postamble. This can happen because the DQS is 3-stated by the memory at the end of a read. The DQS can then go into an indeterminate value, causing false clocking of the IDDR.

After initialization and calibration is done, the controller is signaled to start normal operation of the design. Now, the controller can start issuing user write and read commands to the memory.

## DDR2 SDRAM System and User Interface Signals

Table 9-3 and Table 9-4 describe the system interface signals for designs generated with and without a DCM, respectively.

Table 9-3: DDR2 SDRAM Controller System Interface Signals (with a DCM)

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	200 MHz input differential clock for the IDELAYCTRL primitive of Virtex-5 FPGAs.
sys_rst_n	Input	Active-Low reset to the DDR2 controller.

Table 9-4: DDR2 SDRAM Controller System Interface Signals (without a DCM)

Signal	Direction	Description
clk0	Input	The DDR2 SDRAM controller and memory operate on this clock.
clk90	Input	90° phase-shifted clock with the same frequency as clk0.
clk200	Input	200 MHz input differential clock for the IDELAYCTRL primitive of Virtex-5 FPGAs.
sys_rst_n	Input	Active-Low reset to the DDR2 SDRAM controller. This signal is used to generate the synchronous system reset.
dcm_lock	Input	The status signal indicating whether the DCM is locked or not. This signal is used to generate the synchronous system reset.

Table 9-5 describes the user interface signals.

Table 9-5: DDR2 SDRAM Controller User Interface Signals

Signal	Direction <sup>(1)</sup>	Description
app_af_cmd[2:0]	Input	3-bit command to the Virtex-5 DDR2 SDRAM design. app_af_cmd = 3'b000 for write command app_af_cmd = 3'b001 for read command Other combinations are invalid. Functionality of the controller is unpredictable for unimplemented commands.
app_af_addr[30:0] <sup>(2)</sup>	Input	Gives information about the address of the memory location to be accessed. This bus contains the bank address, the row address, and the column address. Column address = app_af_addr[COL_WIDTH-1: 0] Row address = app_af_addr[ROW_WIDTH+COL_WIDTH-1: COL_WIDTH] Bank address = app_af_addr[BANK_WIDTH+ROW_WIDTH+COL_WIDTH-1: ROW_WIDTH+COL_WIDTH]
app_af_wren	Input	Write enable to the User Address FIFO. This signal should be synchronized with the app_af_addr and app_af_cmd signals.
app_wdf_data[2*DQ_WIDTH-1:0]	Input	User input data. It should contain the fall data and the rise data. Rise data = app_wdf_data[DQ_WIDTH-1: 0] Fall data = app_wdf_data[2*DQ_WIDTH-1: DQ_WIDTH]
app_wdf_mask_data[2*DM_WIDTH-1: 0]	Input	User mask data. It should contain the masking information for both rise and fall data. Rise mask data = app_wdf_mask_data[DM_WIDTH-1: 0] Fall mask data = app_wdf_mask_data[2*DM_WIDTH-1: DM_WIDTH]
app_wdf_wren	Input	Write enable for the User Write FIFO. This signal should be synchronized with the app_wdf_data and app_wdf_mask_data signals.
app_af_afull	Output	Almost Full status of the Address FIFO. When this signal is asserted, the user can write 12 more locations into the FIFO.
app_wdf_afull	Output	Almost Full status of the User Write FIFO. When this signal is asserted, the user can write 12 more locations into the FIFO.
rd_data_valid	Output	Status signal indicating read data is valid on the read data bus.
rd_data_fifo_out[2*DQ_WIDTH-1: 0]	Output	Read data from the memory.
phy_init_done	Output	Indicates the completion of initialization and calibration of the design.
clk0_tb	Output	Clock output to the user. All user interface signals must be synchronized with this clock.
rst0_tb	Output	Active-High reset for the user interface.

**Notes:**

1. Direction indicated in the table is referenced from the design perspective. For example, input here indicates that the signal is input to the design.
2. Addressing in Virtex-5 is linear addressing i.e. the row address immediately follows the column address bits, and the bank address follows the row address bits, thus supporting more devices.



## User Interface Accesses

The user backend logic communicates with the memory controller through a FIFO-based user interface. This interface consists of three related buses: (1) a command/address FIFO bus accepts write/read commands as well as the corresponding memory address from the user, (2) a write data FIFO bus accepts the corresponding write data when the user issues a write command on the command/address bus, and (3) a read bus on which the corresponding read data for an issued read command is returned.

The user interface has the following timing and signaling restrictions:

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

## Write Interface

Figure 9-9 shows the user interface block diagram for write operations.

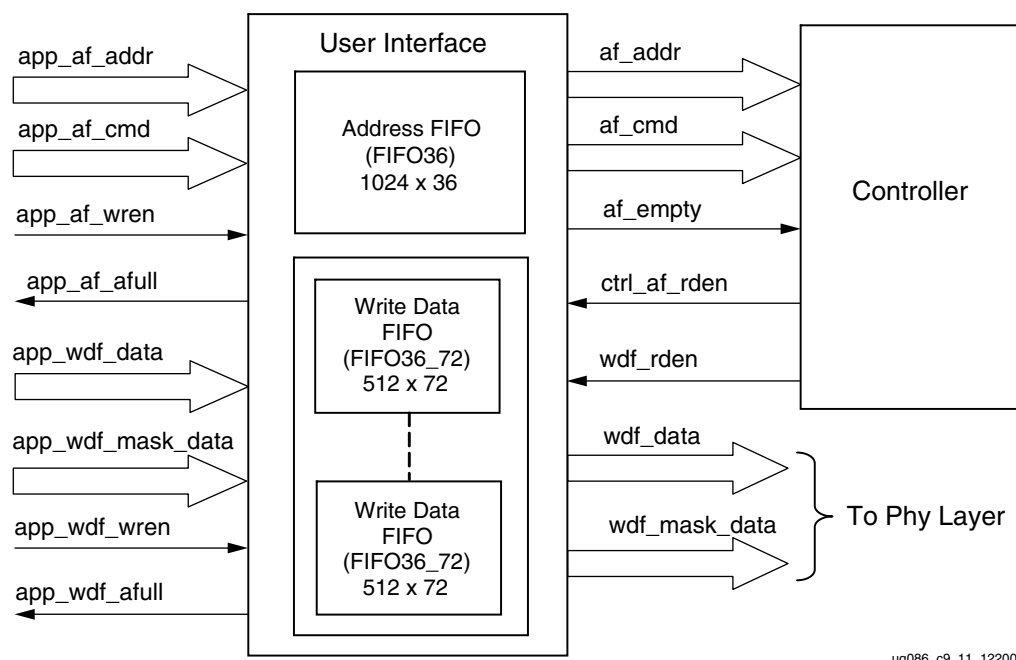


Figure 9-9: User Interface Block Diagram for Write Operation

The following steps describe the architecture of the Address and Write Data FIFOs and show how to perform a write burst operation to DDR2 SDRAM from the user interface.

1. The user interface consists of an Address FIFO and a Write Data FIFO. The Write Data FIFO is constructed using the Virtex-5 FIFO36\_72 primitive with a 512 x 72 configuration. The 72-bit architecture comprises one 64-bit port and one 8-bit port. For Write Data FIFOs, the 64-bit port is used for data bits and the 8-bit port is used for mask bits for ECC-disabled designs. Mask bits are available only when supported by the memory part and when Data Mask is enabled in the MIG GUI. Some memory parts, such as Registered DIMMs of x4 parts, do not support mask bits.

2. In ECC-enabled designs, the 64-bit port is used for data bits and the 8-bit port is used for ECC data. The attributes passed to the Virtex-5 FIFO36\_72 primitive are different for ECC-enabled designs; attribute EN\_ECC\_WRITE is set to TRUE for ECC-enabled designs to enable the generation of ECC data.
3. The Address FIFO is constructed using the Virtex-5 FIFO36 primitive with a 1024 x 36 configuration. The 36-bit architecture comprises one 32-bit port and one 4-bit port. The 32-bit port is used for the address (app\_af\_addr) and the 4-bit port is used for the command (app\_af\_cmd).
4. The Address FIFO is common for both Write and Read commands. It comprises an address part and a command part. Command bits discriminate between write and read commands.
5. User interface data width app\_wdf\_data is twice that of the memory data width. For an 8-bit memory width, the user interface is 16 bits consisting of rising-edge data and falling-edge data. There is a mask bit for every 8 bits of data. For 72-bit memory data, the user interface data width app\_wdf\_data is 144 bits, and the mask data app\_wdf\_mask\_data is 18 bits.
6. The minimum configuration of the Write Data FIFO is 512 x 72 for a memory data width of 8 bits. For an 8-bit memory data width, the least-significant 16 bits of the data port are used for write data and the least-significant two bits of the 8-bit port are used for mask bits. The controller internally pads all zeros for the most-significant 48 bits of the 64-bit port and the most-significant 6 bits of the 8-bit port.
7. Depending on the memory data width, MIG instantiates multiple FIFO36\_72s to gain the required width. For designs using 8-bit to 32-bit data width, one FIFO36\_72 is instantiated; for 72-bit data width, a total of three FIFO36\_72s are instantiated. The bit architecture comprises 32 bits of rising-edge data, 4 bits of rising-edge mask, 32 bits of falling-edge data, and 4 bits of falling-edge mask, which are all stored in a FIFO36\_72. MIG routes app\_wdf\_data and app\_wdf\_mask\_data to FIFO36\_72s accordingly.
8. The user can initiate a write to memory by writing to the Address FIFO and the Write Data FIFO when the FIFO full flags are deasserted. Status signal app\_af\_afull is asserted when the Address FIFO is full; similarly, app\_wdf\_afull is asserted when the Write Data FIFO is full.
9. At power on, both the Address FIFO and Write Data FIFO full flags are deasserted.
10. The user should assert Address FIFO write-enable signal app\_af\_wren along with address app\_af\_addr and command app\_af\_cmd to store the address and command into Address FIFO.
11. The user data should be synchronized to the clk\_tb clock. The user should assert the Data FIFO write-enable signal app\_wdf\_wren along with write data app\_wdf\_data and mask data app\_wdf\_mask\_data to store the write data and mask data into the Write Data FIFOs. The user should provide both rising-edge and falling-edge data together for each write to the Data FIFO. The Virtex-5 DDR2 SDRAM controller design supports byte-wise masking of data only.
12. The write command should be given by keeping app\_af\_cmd = 3'b000 and asserting app\_af\_wren. Address information is given on the app\_af\_addr signal. Address and command information is written into the User Address FIFO.
13. After the completion of the initialization and calibration process and when the User Address FIFO empty signal is deasserted, the controller reads the Command and Address FIFO and issues a write command to the DDR2 SDRAM.
14. The write timing diagram in Figure 9-10 is derived from the MIG-generated test bench for a burst length of 4. As shown, each write to the Address FIFO should have two

writes to the Data FIFO. The phy\_init\_done signal indicates memory initialization and calibration completion.

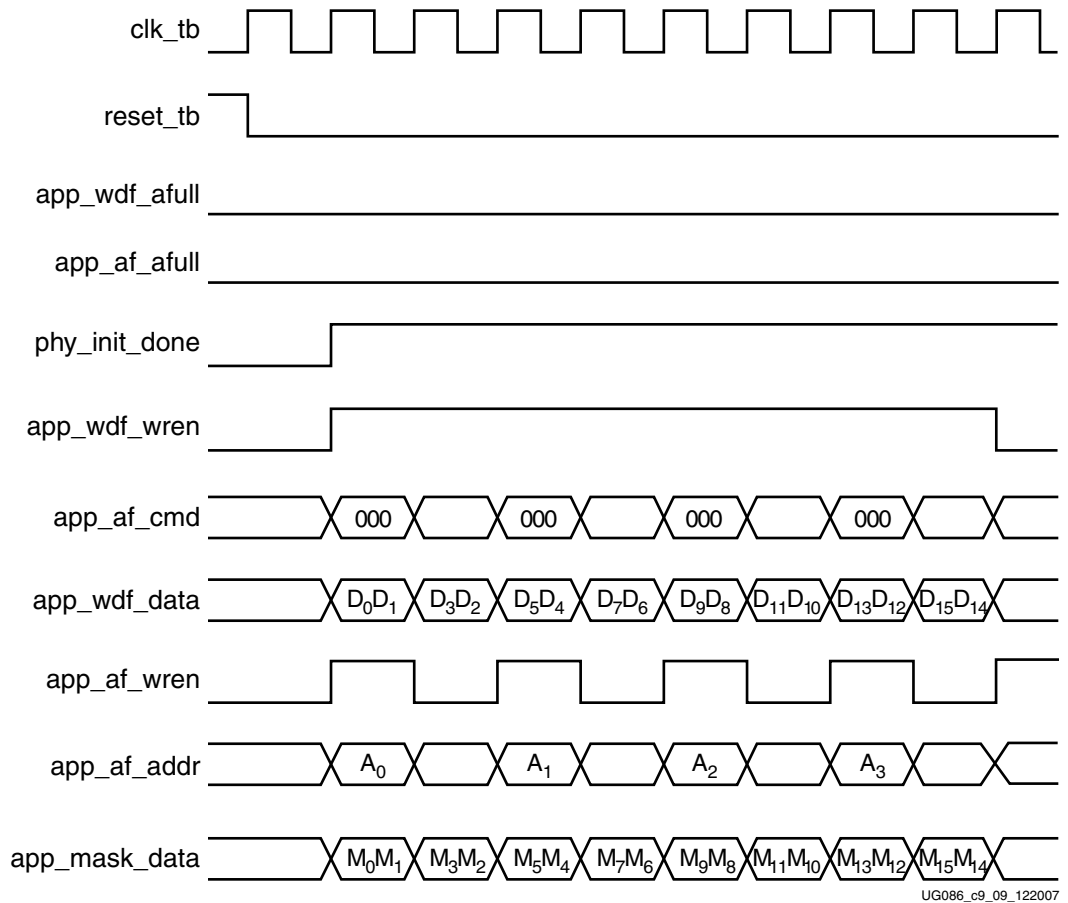
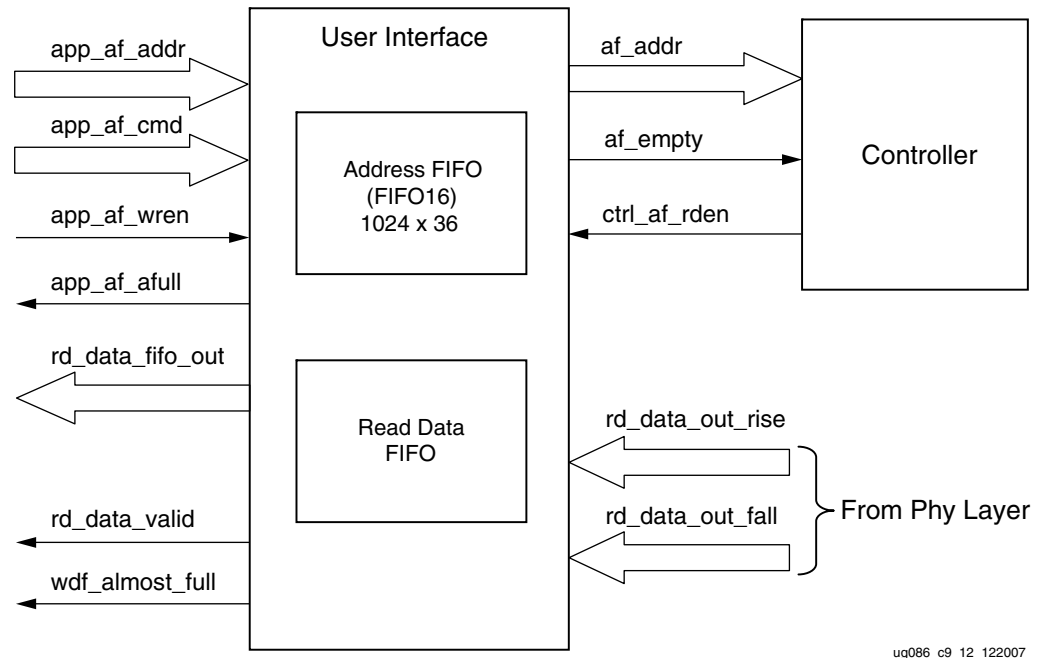


Figure 9-10: DDR2 SDRAM Write Burst for Four Bursts (BL = 4)

## Read Interface

Figure 9-11 shows the block diagram of the read interface.



ug086\_c9\_12\_122007

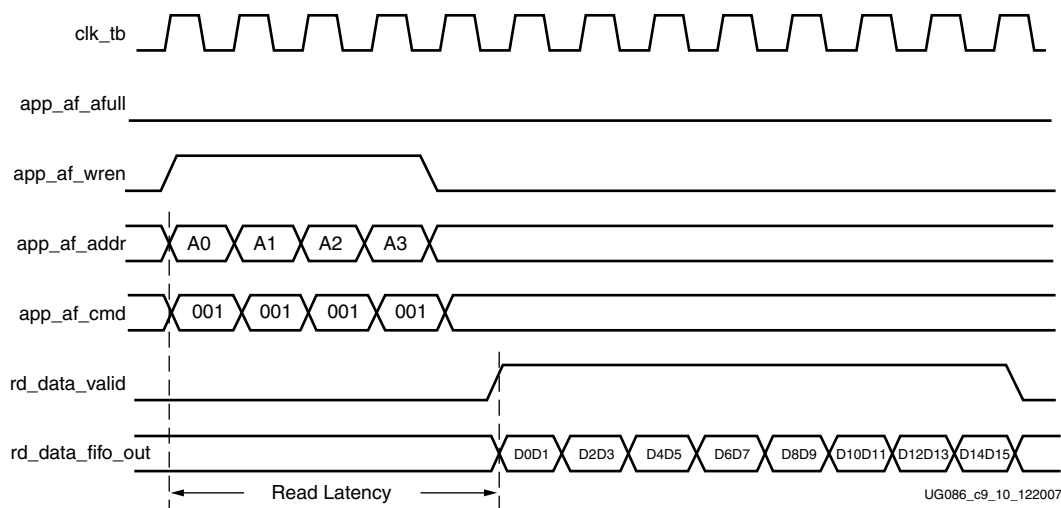
Figure 9-11: User Interface Block Diagram for Read Operation

The following steps describe the architecture of the Read Data FIFO and show how to perform a read burst operation from DDR2 SDRAM from user interface.

1. The Read Data FIFOs are constructed using the Virtex-5 FIFO36\_72 primitive with a 512 x 72 configuration for ECC-enabled designs. For non-ECC designs, read data is latched using the flops.
2. In ECC-enabled designs, the 64-bit port is used for data bits and the 8-bit port is used for ECC data. The Virtex-5 FIFO36\_72 performs ECC comparison when the attribute EN\_ECC\_READ is set during read operation. MIG instantiates the FIFOs appropriately for ECC or non-ECC designs.
3. The user can initiate a read to memory by writing to the Address FIFO when the FIFO full flag app\_af\_afull is deasserted.
4. To write the read address and read command into the Address FIFO, the user should issue the Address FIFO write-enable signal app\_af\_wren along with read address app\_af\_addr and app\_af\_cmd is the command (set to 001 for a read command).
5. The controller reads the Address FIFO and generates the appropriate control signals to memory. After decoding app\_af\_cmd, the controller issues a read command to the memory at the specified address.
6. Prior to the actual read and write commands, the design calibrates the latency in number of clock cycles from the time the read command is issued to the time the data is received. Using this precalibrated delay information, the controller stores the read data in the Read Data FIFOs.
7. The read\_data\_valid signal is asserted when data is available in the Read Data FIFOs.
8. When the calibration is completed, the controller generates the control signals to capture the read data from the FIFO according to the CAS latency selected by the user.

The `rd_data_valid` signal is asserted when the read data is available to the user, and `rd_data_fifo_out` is the read data from the memory to the user.

9. [Figure 9-12](#) shows the user interface timing diagram for burst length of four.



**Figure 9-12: DDR2 SDRAM Read Burst (BL = 4) for Four Bursts**

Read latency is defined as the time between when the read command is written to the user interface bus until when the corresponding first piece of data is available on the user interface bus (see [Figure 9-12](#)).

When benchmarking read latencies, it is important to specify the exact conditions under which the measurement occurs.

Read latency varies based on the following parameters:

- Number of commands already in the FIFO pipeline before the read command is issued
- Whether an ACTIVATE command needs to be issued to open the new bank/row
- Whether a PRECHARGE command needs to be issued to close a previously opened bank
- Specific timing parameters for the memory, such as  $T_{RAS}$  and  $T_{RCD}$  in conjunction with the bus clock frequency
- Commands can be interrupted, and banks/rows can forcibly be closed when the periodic AUTO REFRESH command is issued
- CAS latency
- Board-level and chip-level (for both memory and FPGA) propagation delays

[Table 9-6](#) and [Table 9-7](#) show read latencies for the Virtex-5 DDR2 interface for two different conditions. [Table 9-6](#) shows the case where a row activate is not required prior to issuing a read command on the DDR bus. This situation is possible, for example, when bank management is enabled, and the read targets an already opened bank. [Table 9-7](#) shows the case when a read results in a bank/row conflict. In this case, a precharge of the previous row must be followed by an activation of the new row, which increases read latency. Other specific conditions are noted in the footnotes for each table.

**Table 9-6: Read Latency without Precharge and Activate**

Parameter	Number of Clocks
User READ command to empty signal deassertion (using FIFO36)	1 clock
Empty signal to READ command on DDR2 bus	8.5 clocks
READ command to read valid assertion	8.5 clocks
<b>Total</b>	<b>18 clocks</b>

**Notes:**

1. Test conditions: Clock frequency = 333 MHz, CAS latency = 5, DDR2 -3E speed grade device.
2. Access conditions: Read to an already open bank/row is issued to an empty control/address FIFO.
3. Some entries have fractional clock cycles because the inverted version of CLK0 is used to drive the DDR2 memory.
4. The Virtex-5 FPGA DDR2 interface uses a FIFO36 for the address/control FIFO. It is possible to shorten the READ command to empty signal deassertion latency by implementing the FIFO as a distributed RAM FIFO or removing the FIFO altogether, as the application requires.

**Table 9-7: Read Latency with Precharge and Activate**

Parameter	Number of Clocks
User READ command to empty signal deassertion (using FIFO36)	1 clock
Empty signal to PRECHARGE command on DDR2 bus	8.5 clocks
PRECHARGE to ACTIVE command to DDR2 memory	4 clocks
ACTIVE to READ command to DDR2 memory	4 clocks
READ command to read valid assertion	8.5 clocks
<b>Total</b>	<b>26 clocks</b>

**Notes:**

1. Test conditions: Clock frequency = 333 MHz, CAS latency = 5, DDR2 -3E speed grade device.
2. Access conditions: Read that results in a bank/row conflict is issued to an empty control/address FIFO. This requires that the previous bank/row be closed first.
3. Some entries have fractional clock cycles because the inverted version of CLK0 is used to drive the DDR2 memory.
4. The Virtex-5 FPGA DDR2 interface uses a FIFO36 for the address/control FIFO. It is possible to shorten the READ command to empty signal deassertion latency by implementing the FIFO as a distributed RAM FIFO or removing the FIFO altogether, as the application requires.

## Simulating the DDR2 SDRAM Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains the external testbench, memory model, `.do` file, and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more details about the files in the `sim` folder and to simulate the design, see `simulation_help.chm` in the `sim` folder.

## Supported Devices

The design generated out of MIG is independent of memory package, hence the package part of the memory component is replaced with XX or XXX, where XX or XXX indicates a don't care condition. The tables below list the components (Table 9-8) and DIMMs

(Table 9-9) supported by the tool for the DDR2 design. In supported devices, X in the components column denotes a single alphanumeric character. For example, MT47H128M4XX-3 can be either MT47H128M4BP-3 or MT47H128M4B6-3. XX for Registered DIMMs denotes a single or two alphanumeric characters. For example, MT9HTF3272XX-667 can be either MT9HTF3272Y-667 or MT9HTF3272DY-667.

Table 9-8: Supported Components for DDR2 SDRAM (Virtex-5 FPGAs)

Components	Packages (XX)	Components	Packages (XX)
MT47H64M4XX-3	BP	MT47H128M8XX-3	BT,HQ
MT47H64M4XX-37E	BP	MT47H128M8XX-37E	BT,HQ
MT47H64M4XX-5E	BP	MT47H128M8XX-5E	BT,HQ
MT47H128M4XX-3	B6,CB,GB	MT47H256M8XX-3	HG
MT47H128M4XX-37E	B6,CB,GB	MT47H256M8XX-37E	HG
MT47H128M4XX-5E	B6,CB,GB	MT47H256M8XX-5E	HG
MT47H256M4XX-3	BT,HQ	MT47H16M16XX-3	BG
MT47H256M4XX-37E	BT,HQ	MT47H16M16XX-37E	BG
MT47H256M4XX-5E	BT,HQ	MT47H16M16XX-5E	BG
MT47H512M4XX-3	HG	MT47H32M16XX-3	BN,CC,FN,GC
MT47H512M4XX-37E	HG	MT47H32M16XX-37E	BN,CC,FN,GC
MT47H512M4XX-5E	HG	MT47H32M16XX-5E	BN,CC,FN,GC
MT47H32M8XX-3	BP	MT47H64M16XX-3	BT,HR
MT47H32M8XX-37E	BP	MT47H64M16XX-37E	BT,HR
MT47H32M8XX-5E	BP	MT47H64M16XX-5E	BT,HR
MT47H64M8XX-3	B6,CB,F6,GB	MT47H128M16XX-3	HG
MT47H64M8XX-37E	B6,CB,F6,GB	MT47H128M16XX-37E	HG
MT47H64M8XX-5E	B6,CB,F6,GB	MT47H128M16XX-5E	--

Table 9-9: Supported Registered DIMMs for DDR2 SDRAM (Virtex-5 FPGAs)

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9HTF3272XX-667	--	MT18HTF6472XX-667	--
MT9HTF3272XX-53E	Y	MT18HTF6472XX-53E	DY,Y
MT9HTF3272XX-40E	Y	MT18HTF6472XX-40E	DY,Y
MT9HTF6472XX-667	PY,Y	MT18HTF12872XX-667	DY,PDY,PY,Y
MT9HTF6472XX-53E	Y	MT18HTF12872XX-53E	DY,MY,NDY, NY,PY,Y
MT9HTF6472XX-40E	Y	MT18HTF12872XX-40E	DY,PY,Y
MT9HTF12872XX-667	PY	MT18HTF25672XX-667	PDY,PY,Y
MT9HTF12872XX-53E	PY,Y	MT18HTF25672XX-53E	PDY,PY,Y
MT9HTF12872XX-40E	Y	MT18HTF25672XX-40E	DY,PDY,Y
MT18HTF6472G-53E	--	--	--

**Table 9-10: Supported UDIMMs for DDR2 SDRAM (Virtex-5 FPGAs)**

MT4HTF1664AY-667	MT8HTF6464AY-53E
MT4HTF1664AY-40E	MT8HTF6464AY-40E
MT4HTF3264AY-667	MT8HTF12864AY-667
MT4HTF3264AY-40E	MT8HTF12864AY-40E
MT4HTF6464AY-667	MT9HTF3272AY-667
MT4HTF6464AY-40E	MT9HTF3272AY-40E
MT8HTF6464AY-667	MT9HTF6472AY-667

**Table 9-11: Supported SODIMMs for DDR2 SDRAM (Virtex-5 FPGAs)**

MT4HTF1664HY-667	MT8HTF3264HY-667
MT4HTF1664HY-53E	MT8HTF3264HY-53E
MT4HTF1664HY-40E	MT8HTF3264HY-40E
MT4HTF3264HY-667	MT8HTF6464HY-667
MT4HTF3264HY-53E	MT8HTF6464HY-53E
MT4HTF3264HY-40E	MT8HTF6464HY-40E

## Hardware Tested Configurations

The frequencies shown in [Table 9-12](#) were achieved on the Virtex-5 FPGA ML561 Memory Interfaces Development Board under nominal conditions. These frequencies should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based on a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 72-bit wide interface.

**Table 9-12: Hardware Tested Configurations**

Synthesis Tools	XST and Synplicity
HDL	Verilog and VHDL
FPGA Device	XC5VLX50T-FF1136-2
Burst Lengths	4, 8
CAS Latency (CL)	3, 4, 5
Additive Latency	0, 1, 2, 3, 4
32-bit Design	Tested on 16-bit Component MT47H32M16XX-3
72-bit RDIMM Design	Tested on 72-bit DIMM MT9HTF6472XX-667
72-bit UDIMM Design	Tested on 72-bit DIMM MT9HTF6472AY-667
ECC verified	72-bit RDIMM and UDIMM design
Component, CL=3, 4, 5	100 MHz to 400 MHz
DIMM, CL=3	100 MHz to 280 MHz
DIMM, CL=4, 5	100 MHz to 400 MHz



## Implementing QDRII SRAM Controllers

This chapter describes how to implement QDRII SRAM interfaces for Virtex™-5 FPGAs generated by MIG. This design is based on XAPP853 [Ref 25].

### Feature Summary

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

#### Supported Features

The QDRII controller design supports the following:

- A maximum frequency of 300 MHz
- 18-bit, 36-bit, and 72-bit data widths
- Burst lengths of four and two
- Implementation using different Virtex-5 devices
- Support for DCI Cascading
- Operation with 18-bit and 36-bit memory components
- Verilog and VHDL
- With and without a testbench
- With and without a DCM

#### Design Frequency Ranges

Table 10-1: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-1		-2		-3	
	Min	Max	Min	Max	Min	Max
Component	120	250	120	300	120	300

#### Unsupported Features

The QDRII controller design does not support:

- 9-bit data widths
- 9-bit memory components

## Architecture

Figure 10-1 shows a top-level block diagram of the QDRII memory controller. One side of the QDRII memory controller connects to the user interface denoted as User Interface. The other side of the controller interfaces to QDRII memory. The memory interface data width is selectable from MIG.

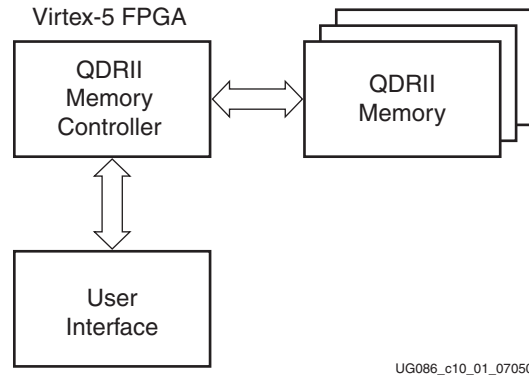


Figure 10-1: QDRII Memory Controller

The QDR operation can support double data rated read and write operations through separate data output and input ports with the same cycle. Memory bandwidth is maximized because data can be transferred into SRAM on every edge of the clock and transferred out of SRAM on every edge of the read clock. Independent read and write ports eliminate the need for high-speed bus turnaround.

Read and write addresses are latched on positive edges of the input clock K. A common address bus is used to access the addresses for both read and write operations. The key advantage to QDRII devices is they have separate data buses for reads and writes to SRAM.

## Interface Model

The QDRII memory interface is layered to simplify the design and make the design modular. Figure 10-2 shows the layered memory interface in the QDRII memory controller. The two layers are the application layer and the implementation and physical layer.

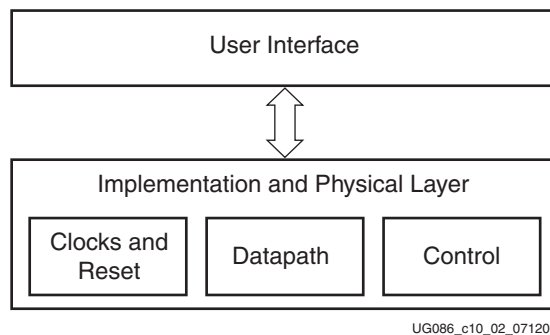


Figure 10-2: Interface Layering Model

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 implementation and physical layer comprises:



The modules are classified as follows:

- Design modules
- Testbench modules
- Clocks and reset generation modules

MIG can generate four different QDRII SRAM designs:

- With a testbench and a DCM
- Without a testbench and with a DCM
- With a testbench and without a DCM
- Without a testbench and without a DCM

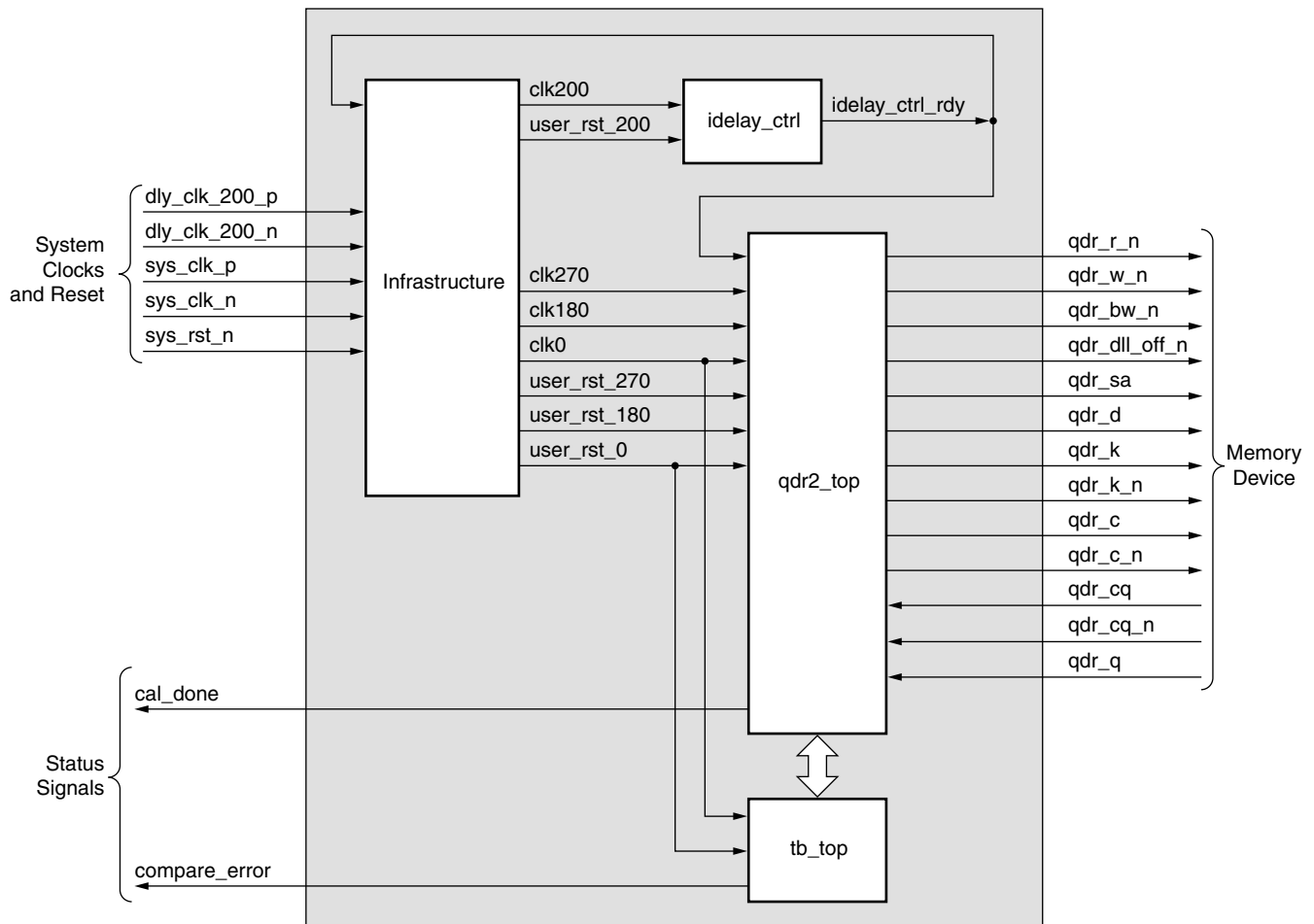
For designs without a testbench (*user\_design*), testbench modules are not present in the design. The `<top_module>` (top level) module has the user interface signals for designs without a testbench. The list of user interface signals is provided in [Table 10-4](#).

Design clocks and resets are generated in the infrastructure module. The DCM clock is instantiated in the infrastructure module for designs with a DCM. The inputs to this module are the differential design clock and a 200 MHz differential clock for the IDELAYCTRL module. A user reset is also input to this module. Using the input clocks and reset signal, system clocks and system reset signals are generated in this module, which are used in the design.

The DCM primitive is not instantiated in this module if the “No DCM” option is selected. So, the system operates on the user-provided clocks. The system reset signals are generated in the infrastructure module using the `dcm_lock` input signal, the input reset signal, and the idelay control element’s ready signal.

The QDRII design is generated in two configurations with and without a testbench (*example\_design* and *user\_design* respectively). The top-level module with testbench (*example\_design*) has the design top, testbench, IDELAY control, and clock and reset modules. Without a testbench (*user\_design*), the `mem_test_bench` module is removed from the top-level module. By default, MIG outputs both designs (*example\_design* and *user\_design*) in two separate RTL folders, and the user can choose the appropriate design.

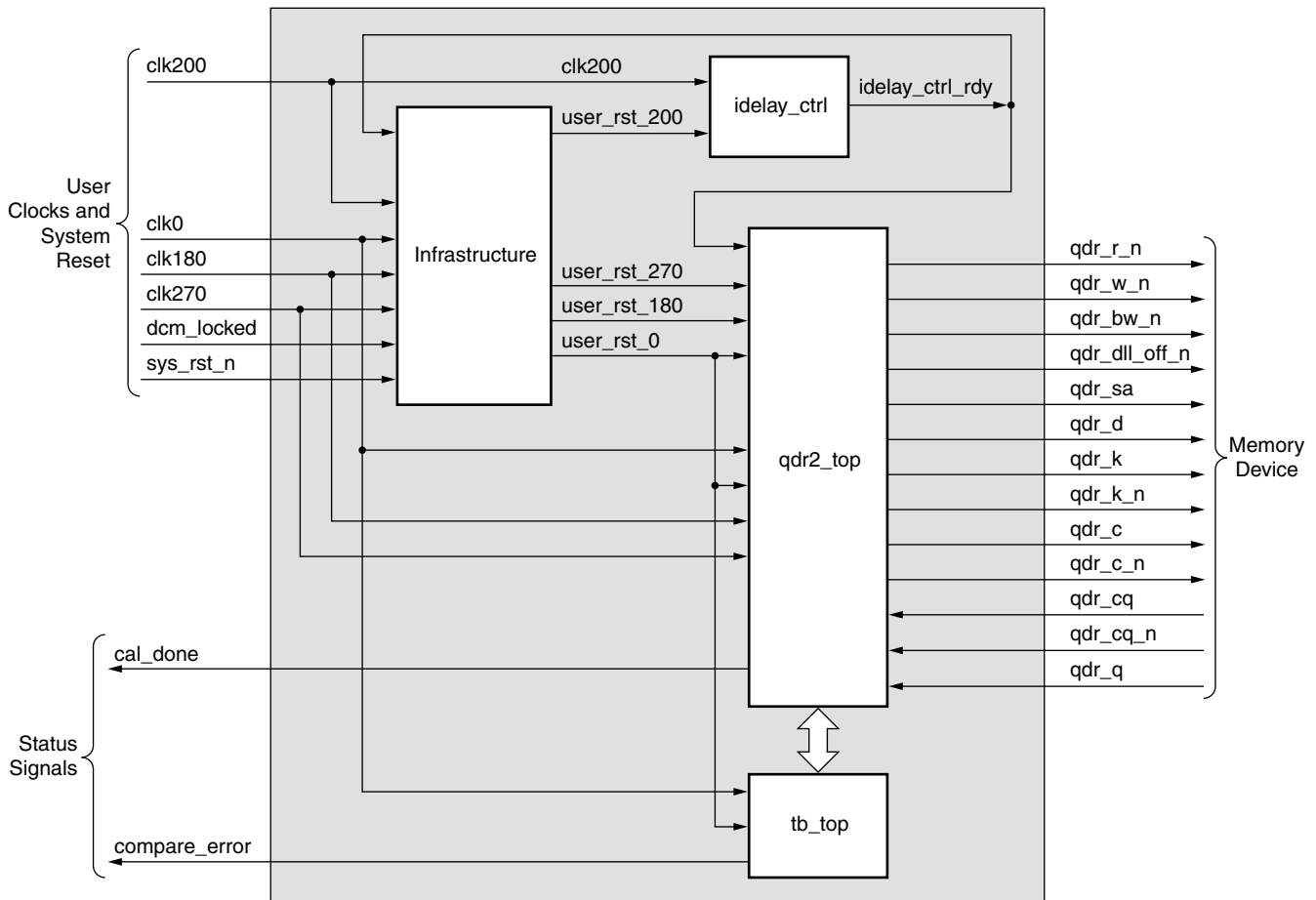
Figure 10-4 shows a top-level block diagram of a QDRII SRAM design with a DCM and a testbench. `sys_clk_p` and `sys_clk_n` are differential input system clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. `dly_clk_200_p` and `dly_clk_200_n` are used for the `idelay_ctrl` element. `sys_rst_n` is an active-Low system reset signal. All design resets are generated using the `sys_rst_n` signal, the `dcm_locked` signal, and the `dly_ready` signal of the `IDELAYCTRL` element. The `compare_error` output signal indicates whether the design passes or fails. The testbench module called “`tb_top`” generates the user interface data, address, and command signals. The user data bits and address bits are stored in the corresponding User Interface FIFOs. The `compare_error` signal is driven High on data mismatches. The `cal_done` signal indicates the completion of initialization and calibration of the design.



UG086\_c10\_04\_091707

Figure 10-4: Top-Level Block Diagram of the QDRII SRAM Design with a DCM and a Testbench

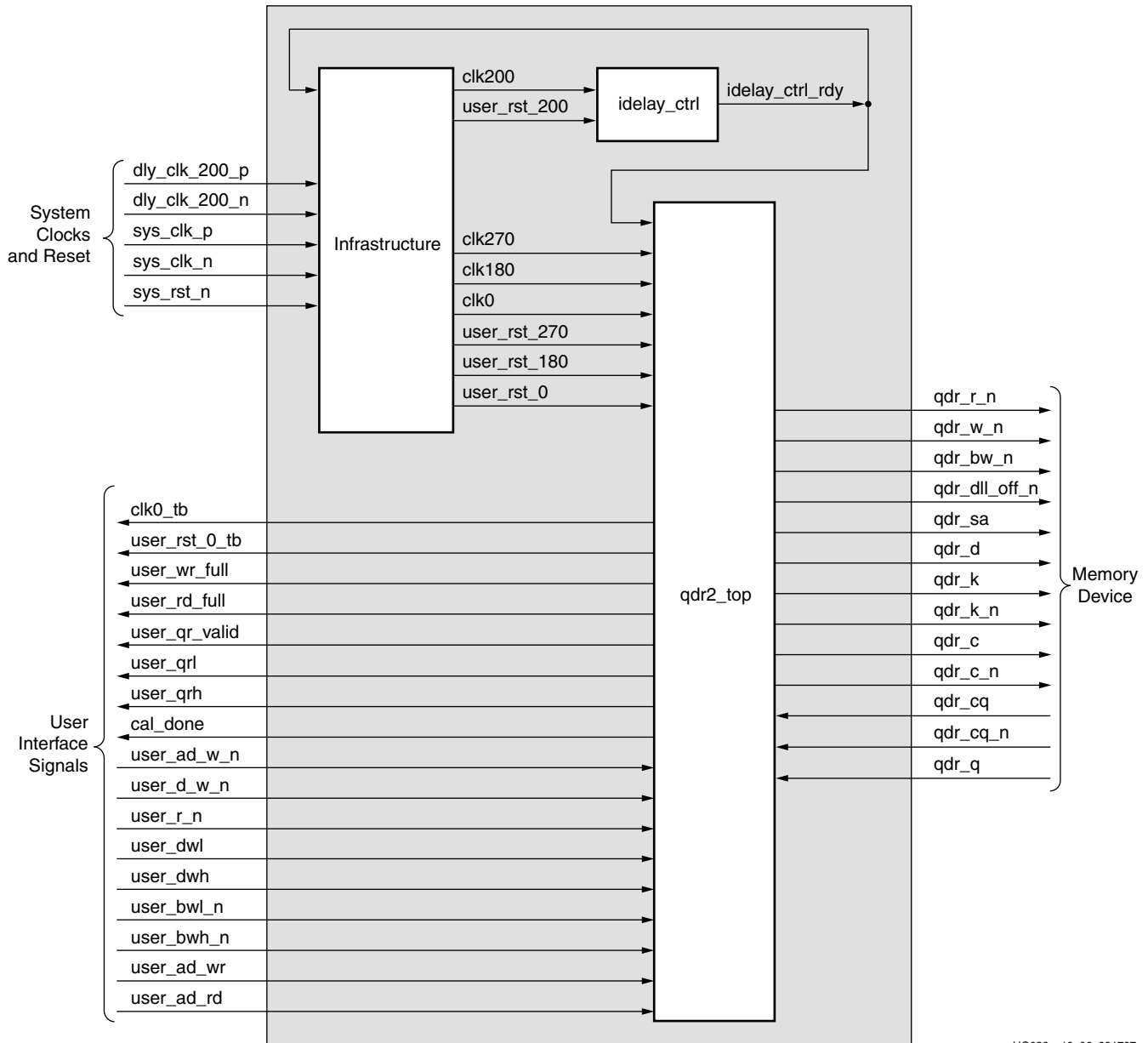
Figure 10-5 shows a top-level block diagram of a QDRII SRAM design without a DCM but with a testbench. The user should provide all the clocks and the dcm\_locked signal. These clocks should be single-ended. sys\_rst\_n is the active-Low system reset signal. All design resets are generated using the sys\_rst\_n signal, the dcm\_locked signal, and the dly\_rdy signal of the IDELAYCTRL element. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFPGs. The compare\_error output signal indicates whether the case passes or fails. The testbench module called "tb\_top" generates the user interface data, address, and command signals. The user data bits and address bits are stored in the corresponding User Interface FIFOs. The compare\_error signal is driven High on data mismatches. The cal\_done signal indicates the completion of initialization and calibration of the design.



UG086\_c10\_05\_091707

Figure 10-5: Top-Level Block Diagram of the QDRII SRAM Design without a DCM but with a Testbench

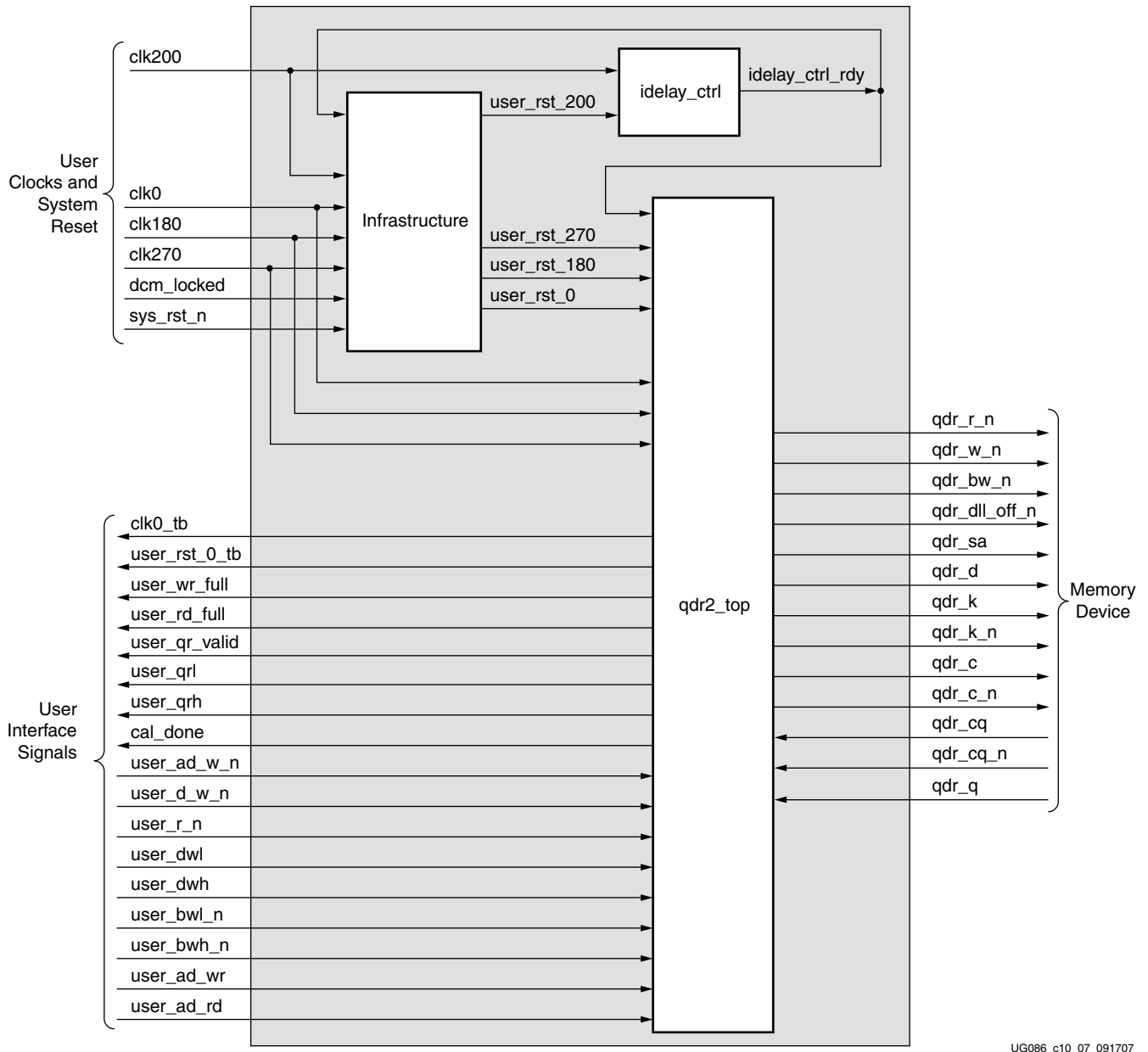
Figure 10-6 shows a top-level block diagram of a QDRII SRAM design with a DCM but without a testbench. `sys_clk_p` and `sys_clk_n` are differential input system clocks. The DCM is instantiated in the infrastructure module that generates the required design clocks. `dly_clk_200_p` and `dly_clk_200_n` are used for the `idelay_ctrl` element. `sys_rst_n` is an active-Low system reset signal, and all design resets are generated using the `sys_rst_n` signal, the `dcm_locked` signal, and the `dly_ready` signal of the `IDELAYCTRL` element. The user has to drive the user application signals. The design provides the `clk0_tb` and `user_rst_0_tb` signals to the user in order to synchronize the user application signals with the design. The `cal_done` signal indicates the completion of initialization and calibration of the design.



UG086\_c10\_06\_091707

Figure 10-6: Top-Level Block Diagram of the QDRII SRAM Design with a DCM but without a Testbench

Figure 10-7 shows a top-level block diagram of a QDRII SRAM design without a DCM or a testbench. The user should provide all the clocks and the dcm\_locked signal. These clocks should be single-ended. sys\_rst\_n is the active-Low system reset signal. All design resets are generated using the sys\_rst\_n signal, the dcm\_locked signal, and the dly\_ready signal of the IDELAYCTRL element. The user application must have a DCM primitive instantiated in the design, and all user clocks should be driven through BUFPGs. The user has to drive the user application signals. The design provides the clk0\_tb and user\_rst\_0\_tb signals to the user in order to synchronize the user application signals with the design. The cal\_done signal indicates the completion of initialization and calibration of the design.



UG086\_c10\_07\_091707

Figure 10-7: Top-Level Block Diagram of the QDRII SRAM Design without a DCM or a Testbench



## QDRII Memory Controller Modules

Figure 10-8 shows a detailed block diagram of the QDRII memory controller.

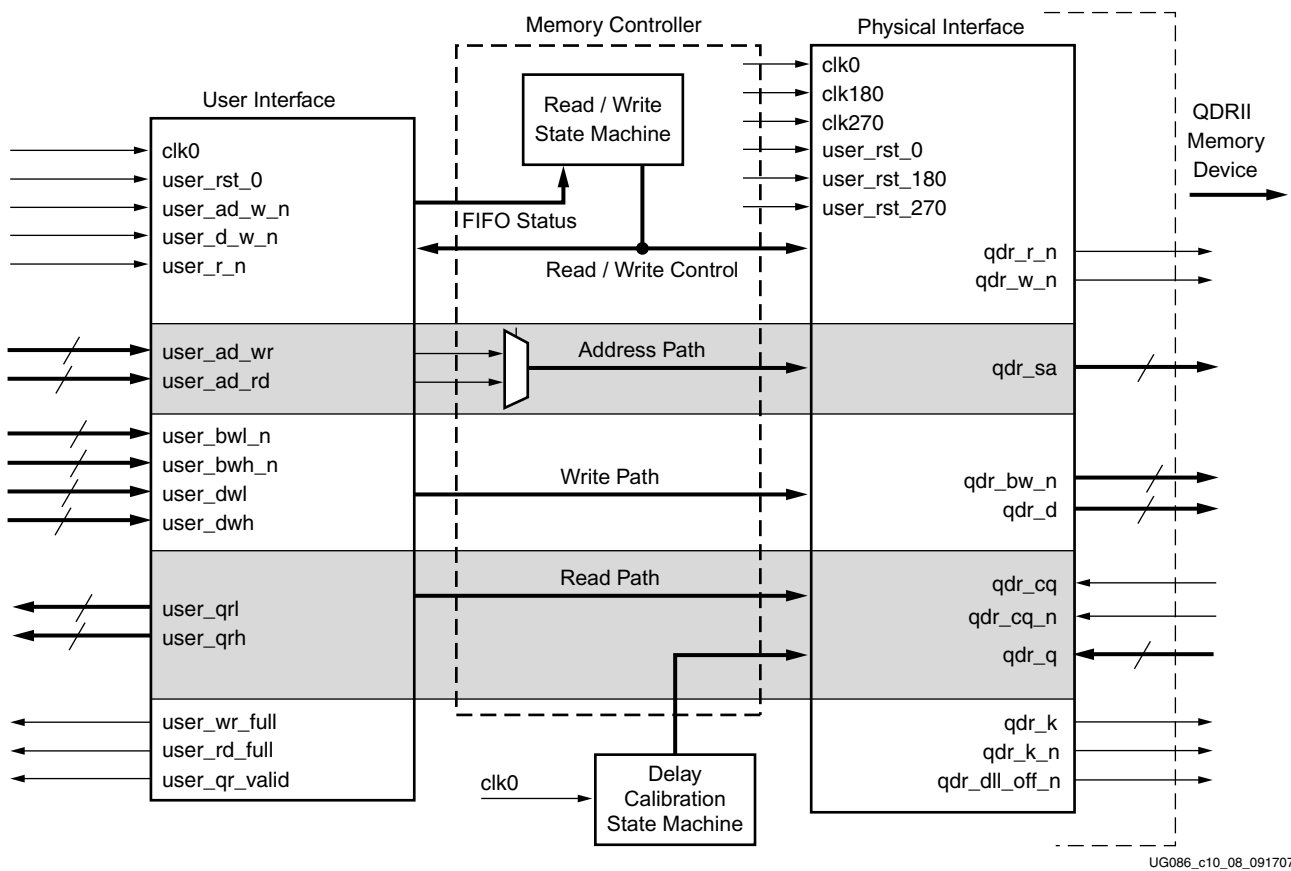


Figure 10-8: QDRII Memory Controller Modules

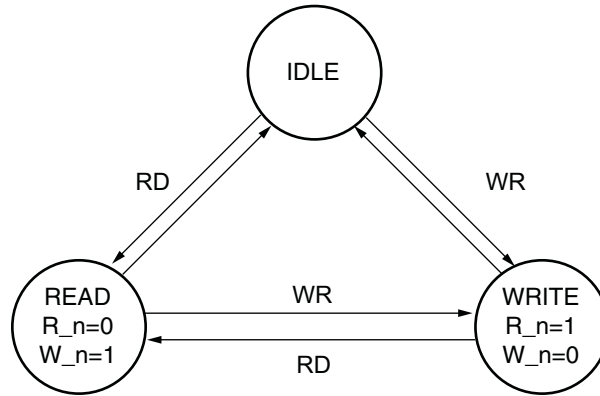
### Controller

The QDRII memory controller initiates alternate WRITE and READ commands to the memory as long as the User Write Address FIFO and the User Read Address FIFO are not empty.

The user writes the write data, its corresponding byte write enable, and the Write Address bits into the User Write Data FIFOs, the User Byte Write FIFO, and the User Write Address FIFOs, respectively. When the User Write Address FIFO is not empty, the QDRII controller generates a write-enable signal to the memory. When the write enable is asserted, the write data, the byte write enable, and the write address bits are transferred to memory from the User Write Data FIFOs, the User Byte Write FIFO, and the User Write Address FIFO, respectively.

The read address from where the data is to be read from the memory is stored by the user in the User Read Address FIFO. The QDRII memory controller generates a read-enable signal to the memory when the User Read Address FIFO is not empty. When the read enable is asserted, the read address from the Read Address FIFO is transferred to memory. When the read data from the memory corresponding to the read address is captured correctly, a valid user\_qr\_valid signal is asserted High. The user can access the read data corresponding to the read address only when the data valid signal user\_qr\_valid is asserted High.

Figure 10-9 shows a state machine of the QDR II memory controller for burst lengths of four. When calibration is complete (that is, when the cal\_done signal is asserted), the state machine is in the IDLE state. When the User Write Address FIFO is not empty (that is, when the user has written the write data, the byte write enable, and the write address bits into their corresponding FIFOs, respectively), the state machine goes to the WRITE state, initiating a memory write of one burst.



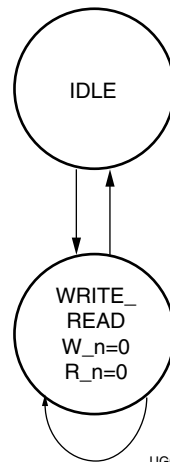
UG086\_c10\_09\_013107

Figure 10-9: QDR II Memory Controller State Machine with Burst Lengths of 4

When the User Read Address FIFO is not empty (that is, the user has written read address bits 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 QDR II memory controller can go to either the WRITE or the READ state depending on the status of the User FIFOs. Writes are given priority. In the WRITE state, a memory write is initiated, and the User Read Address Not Empty status is checked in order 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 Address FIFO Not Empty status is checked before going to the WRITE state. If the User Address FIFO is empty, the state machine goes to the IDLE state.



UG086\_c10\_14\_122007

Figure 10-10: QDR II Memory Controller State Machine with Burst Lengths of 2

Figure 10-10 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 Address FIFO is not empty (that is, when the user has written the write data, the byte write enable, and the write address bits into their corresponding FIFOs), the state machine goes to the WRITE\_READ state, initiating a memory write of one complete burst. When the User Read Address FIFO is not empty (that is, the user has written read address bits into the User Read Address FIFO), the state machine goes to the READ\_WRITE state, initiating a memory read of one complete burst.

From the IDLE state, the QDR II memory controller goes to WRITE\_READ state if either:

- the User Write Address FIFO is not empty, or
- the User Read Address FIFO is not empty.

In the WRITE\_READ state, the User Read Address Not Empty status is checked to initiate a memory read. To initiate a memory write in the WRITE\_READ state, the User Write Address FIFO not empty status is checked. If both the User Write Address FIFO and the User Read Address FIFO are empty, the state machine goes to the IDLE state. If either the User Write Address FIFO or the User Read Address FIFO is not empty, the state machine remains in the WRITE\_READ state to issue memory writes or reads.

Refer to XAPP853 [Ref 25] for data capture techniques and timing analysis of the QDR II memory controller module.

## Infrastructure

The Infrastructure (infrastructure\_top) module comprises the reset generation logic and instantiates a DCM primitive for clock signal generation. Inputs to the infrastructure\_top module are sys\_clk\_p and sys\_clk\_n (the differential clock pair from which the design clocks are generated), dly\_clk\_200\_p and dly\_clk\_200\_n (the differential clock pair for the IDELAYCTRL elements), and sys\_rst\_n (the user reset signal). sys\_clk\_p and sys\_clk\_n are used by the DCM primitive to generate the clock, the 180° phase-shifted version of the clock, and the 270° phase-shifted version of the clock. The QDR II controller works using these clocks. This module even generates reset signals using the sys\_rst\_n signal, the dcm\_lock signal, and the ready signal from the idelay control element for different clock domains that are used by the controller design.

## top\_phy

This module is the interface between the controller and the memory. It consists of the following:

- Control logic that generates READ/WRITE commands and address signals to the memory.
- Write Data logic that associates the write data, the byte enable, and the write address with the WRITE commands and the read address with the READ commands. It also generates the write data pattern for calibration purposes.
- Read Data logic that comprises the read data capturing scheme and calibration logic.

## DCI Cascading

In Virtex-5 family devices, I/O banks that need DCI reference voltage can be cascaded with other DCI I/O banks. One set of VRN/VRP pins can be used to provide reference voltage to several I/O banks. With DCI cascading, one bank (the master bank) must have its VRN/VRP pins connected to external reference resistors. Other banks in the same column (slave banks) can use DCI standards with the same impedance as the master bank, without connecting the VRN/VRP pins on these banks to external resistors. DCI impedance control

in cascaded banks is received from the master bank. This results in more usable pins and in reduced power usage because fewer VR pins and DCI controllers are used.

The syntax for representing the DCI Cascading in the UCF is:

```
CONFIG DCI_CASCADE = "<master> <slave1> <slave2> ...";
```

There are certain rules that need to be followed in order to use DCI Cascade option:

1. The master and slave banks must all reside on the same column (left, center, or right) on the device.
2. Master and slave banks must have the same  $V_{CCO}$  and  $V_{REF}$  (if applicable) voltages.

This feature enables placing all 36 bits of read data, as well as the CQ and CQ# clocks, in the same bank when interfacing with 36-bit QDRII components.

MIG supports DCI Cascading. Following are the possibilities for generating the designs with DCI support using the DCI Cascade option.

- For x36 component designs, the DCI Cascade option is always enabled. This feature cannot be disabled if DCI support is needed.
- For x18 component designs, DCI Cascade is optional. DCI support for these designs can be selected with or without the DCI Cascade selection.
- For x18 component with 18-bit data width designs, the DCI Cascade option is disabled and cannot be utilized.

When DCI Cascade option is selected, MIG displays the master bank selection box for each column of the FPGA in the bank selection page.

- If an FPGA has no banks or has only non-DCI banks in a particular column, the master bank selection box for that column is not displayed.
- All the data read banks are treated as slave banks.
- When a data read bank is selected in a particular column, the master bank selection box for that particular column is activated and the rest of the master bank selection boxes for other columns are deactivated.
- In a particular column, when a data read bank is selected and there are no DCI banks left in that column for master banks selection, then the design cannot be generated. The data read banks must be moved to the other columns in order to select the master banks.
- The master bank selection box shows all the bank numbers in that particular column other than the data read banks and non-DCI banks in that column.
- There can be only one master bank selected for each column of banks.
- MIG utilizes VRN/VRP pins in the slave banks for pin allocation.
- For each master bank, VRN/VRP pins are reserved, and a dummy input pin `masterbank_sel_pin` is allocated and assigned the HSTL\_I\_DCI\_18 I/O standard. This helps to enable the DCI standard for the read data banks. The number of dummy input pins is equal to the number of master banks allocated by MIG.
- The dummy input pin is required to satisfy the requirement of the master bank. Any master bank should have at least one input pin to program the DCI option.
- When all the banks in a particular column are allocated with data read pins, MIG chooses only the required banks for data read pins depending upon the design data width and leaves rest of the banks for master bank selection.

The center column banks of all the FPGAs are divided into two sections, top-column banks and bottom-column banks. Top-column banks are the banks available above the 0th bank,

and the bottom column banks are the banks available below 0th bank. Therefore, there are two master bank selection boxes for the center column.

The VRN/VRP pins for a master bank do not need to be reserved in the reserve pins page.

Once the design is ready with the valid master and slave bank selection, the same master and slave bank information (along with the DCI Cascading syntax) is provided in the UCF when the design is generated.

For more information about DCI Cascade, refer to DCI Cascading in the *Virtex-5 FPGA User Guide* and the *Xilinx Constraints Guide*.

## CQ/CQ\_n Implementation

Controller uses CQ and CQ\_n for capturing read data of a 36-bit component. CQ and CQ\_n are placed on the P pins of the clock-capable I/Os. For a 36-bit component, CQ is used to capture the first 18 bits of the read data, and CQ\_n is used to capture the second 18 bits of the read data. For an 18-bit component, only CQ is used for capturing the read data. CQ\_n is not used, and it is connected to a dummy logic. This dummy logic is used just to retain CQ\_n pin during PAR. Users can use the CQ\_n pin if needed.

## Pinout Considerations

It is recommended to select banks within the same column in MIG. This helps to avoid the clock tree skew that the design would incur while crossing from one column to another.

When the Data Read, Data Write, Address, and System Control pins are allocated to individual banks in a column, then the System Control pins must be allocated in a bank that is central to the rest of banks allocated. This helps reduce data path and clock path skew.

For larger FPGAs (for example, FF1738, FF1760, and similar), it is recommended to place Data Read, Data Write, Address, and System Control pins in the same column to reduce data path and clock path skew.

## User Interface

The user interface has two interfaces: a Read user interface and a Write user interface.

The Read user interface consists of the Read Address interface modules. The Read Address interface consists of the Read Address FIFO. The user has to write the read address bits of the memory into this FIFO.

The Write User interface consists of the Write Data interface and the Write Address interface. The Write Address interface consists of the Write Address FIFO. The user has to write the write address bits of the memory into this FIFO.

The Write Data interface consists of the Write Data FIFO and the Byte Write FIFO. The width of the Write Data FIFO depends upon the data width of the controller design. There are two Write Data FIFOs for every controller: the LSB Write Data FIFO and the MSB Write Data FIFO. The outputs of these FIFOs are SDR and are later converted to DDR at the ODDR primitive before transferring to memory.

The Byte Write enable signals are stored in the Byte Write FIFO by the user.

The controller monitors the status signals of these User FIFOs and issues the READ/WRITE commands to the memory.

The user must wait until the cal\_done signal is asserted by the controller, which indicates completion of calibration prior to writing the user data to the Write Data FIFOs.

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

## QDRII SRAM Initialization and Calibration

QDRII memory is initialized through a specified sequence. Following initialization, the relationship between the data and the FPGA clock is calculated using the TAP logic. The calibration logic is explained briefly as follows.

Calibration is done in three stages:

1. The read strobe CQ is edge-aligned with the read data Q from the memory. The read strobe is a free-running clock from the memory. In the first stage of calibration, the read strobe CQ is passed through the BUFIO, which delays the strobe by the amount of delay in the BUFIO. Now the read strobe CQ is out of synchronization with the read data Q.

A pattern of four bursts of data (with a value of '1' for rise data and '0' for fall data) is written into a particular location in memory. Continuous read commands are issued to the same location of the memory and the read data Q is delayed in the ISERDES, until it is center-aligned with respect to the delayed read strobe CQ.

The `q_init_delay_done` signal in the `phy_read` module indicates the status of the first stage calibration. When `q_init_delay_done` is asserted High, it indicates the completion of first-stage calibration. Now the CQ clocks are centered with respect to the Read Data Q at the input of the ISERDES.

2. In the second stage of calibration, the read data window is center-aligned with respect to the FPGA clock. Here another pattern of four bursts of data is written into a particular memory location. It is read back continuously from the same memory location, and the read data and the delay clock, CQ, are delayed until the registered read data is center-aligned with the FPGA clock.

When the registered read data is center-aligned with the FPGA clock, the alignment of the read data Q with respect to the FPGA clock is complete. The `dly_cal_done` signal in the `phy_read` module indicates the status of second-stage calibration.

3. In the third stage of calibration, the controller issues non-consecutive read commands to the memory. The internal read command signal generated by the controller is then delayed through a shift register until the delayed read command signal is aligned with the ISERDES read data output. Then another level of calibration is done to ensure alignment between the ISERDES data outputs from all the banks used in the interface.

This finishes the calibration of the read data Q, and the `cal_done` signal is asserted High.

XAPP853 [Ref 25] provides more information about the calibration architecture.

The user must strictly follow the pattern data and not modify it. The timing diagrams in “QDRII Controller Interface Signals” explain the user interface commands until the calibration is finished.

## QDRII Controller Interface Signals

Table 10-2 through Table 10-3 describe the QDRII controller system interface signals with and without a DCM, respectively. Table 10-4 describes the QDRII user interface signals. Table 10-5 describes the QDRII memory interface signals. In these tables, all signal directions are with respect to the QDRII memory controller.

Table 10-2: QDRII SRAM System Interface Signals (with a DCM)

Signal Name	Direction	Description
sys_clk_p, sys_clk_n	Input	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.  When the Without DCM option is selected, this clock pair is not present.
dly_clk_200_p, dly_clk_200_n	Input	200 MHz differential clock used in the idelay_ctrl logic.
sys_rst_n	Input	Reset to the QDRII memory controller.
compare_error	Output	This signal represents the status of comparison of read data when compared to the corresponding write data.
cal_done	Output	This signal is asserted when the design initialization and calibration is complete.

Table 10-3: QDRII SRAM System Interface Signals (without a DCM)

Signal Name	Direction	Description
clk0	Input	Input clock
clk180	Input	Input clock with a 180° phase difference
clk270	Input	Input clock with a 270° phase difference
clk200	Input	200 MHz clock for Idelayctrl primitives
dcm_locked	Input	This active-High signal indicates whether the user DCM is locked or not.
sys_rst_n	Input	Reset to the QDRII memory controller
compare_error	Output	This signal represents the status of the comparison between the read data with the corresponding write data.
cal_done	Output	This signal is asserted when the design initialization and calibration is complete.

Table 10-4: QDRII SRAM User Interface Signals (without a Testbench [user\_design])

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 writes to the User Read Address FIFO are ignored.



Table 10-4: QDRII SRAM User Interface Signals (without a Testbench [user\_design]) (Continued)

Signal Name	Direction	Description
user_qr_valid	Output	This status signal indicates that data read from the memory is available to the user.
clk0_tb	Output	All user interface signals are to be synchronized to this clock.
user_rst_0_tb	Output	This reset is active until the DCM is not locked.
user_dwl [(DATA_WIDTH-1):0]	Input	Positive-edge data for memory writes. This data bus is valid when user_d_w_n is asserted.
user_dwh [(DATA_WIDTH-1):0]	Input	Negative-edge data for memory writes. This data bus is valid when user_d_w_n is asserted.
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_bwl_n [(BW_WIDTH-1):0]	Input	Byte enables for QDRII memory positive-edge write data. The byte enables are valid when user_d_w_n is asserted.
user_bwh_n [(BW_WIDTH-1):0]	Input	Byte enables for QDRII memory negative-edge write data. The byte enables are valid when user_d_w_n is asserted.
user_ad_wr [(ADDR_WIDTH-1):0]	Input	QDRII memory address for write data. This address is valid when user_ad_w_n is asserted.
user_ad_rd [(ADDR_WIDTH-1):0]	Input	QDRII memory address for read data. This address is valid when user_r_n is asserted.
user_ad_w_n	Input	This active-Low signal is the write enable for the User Write Address FIFO.
user_d_w_n	Input	This active-Low signal is the write enable for the User Write Data FIFO and Byte Write FIFOs.
user_r_n	Input	This active-Low signal is the write enable for the User Read Address FIFO.

**Notes:**

1. All user interface signal names are prepended with a controller number, for example, cntrl0\_qdr\_q. QDRII SRAM devices currently support only one controller.

Table 10-5: QDRII SRAM Interface Signals

Signal Name	Direction	Description
qdr_d	Output	During WRITE commands, the data is sampled on both edges of K.
qdr_q	Input	During READ commands, the data is sampled on both edges of FPGA clk.
qdr_bw_n	Output	Byte enables for QDRII memory write data. These enable signals are sampled on both edges of the K clock.
qdr_sa	Output	Address for READ and WRITE operations
qdr_w_n	Output	This signal represents the WRITE command.



Table 10-5: QDRII SRAM Interface Signals

Signal Name	Direction	Description
qdr_r_n	Output	This signal represents the READ command.
qdr_cq, qdr_cq_n	Input	These signals are the read clocks transmitted by the QDRII SRAM. Both CQ and CQ_n are used for data capture in this design.
qdr_k, qdr_k_n	Output	Differential write data clocks
qdr_c, qdr_c_n	Output	Input clock to memory for the output data
qdr_dll_off_n	Output	Memory DLL disable when Low

## User Interface Accesses

The user backend logic communicates with the memory controller through a FIFO-based user interface. This interface consists of four related buses:

- A Write Address FIFO bus accepts memory write address from the user
- A Write Data FIFO bus accepts the write data corresponding to the memory write address
- A Read Address FIFO bus accepts the memory read address from the user

The user interface has the following timing and signaling restrictions:

- The Write/Read Address and Write Data FIFOs cannot be written by the user until calibration is complete (as indicated by cal\_done). In addition, the user\_ad\_w\_n, user\_d\_w\_n, and user\_r\_n interface signals need to be held High until calibration is complete.
- For issuing a write command, the memory write address must be written into the Read Address FIFO. The first write data word must be written to the Write Data FIFO on the same clock cycle as the when the write address is written. In addition, the write data burst must be written over consecutive clock cycles; there cannot be a break between bursts of data. These restrictions arise from the fact that the controller assumes write data is available when it receives the write command from the user.

## Write Interface

Figure 10-11 illustrates the user interface block diagram for write operations.

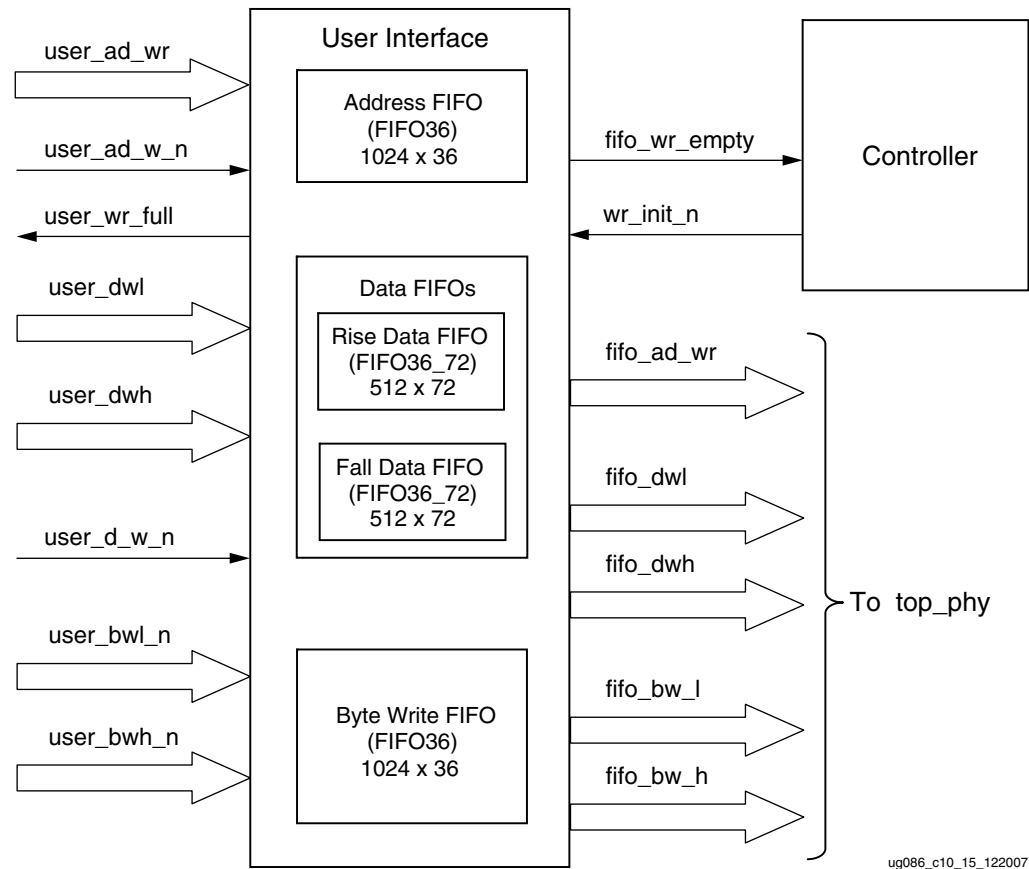


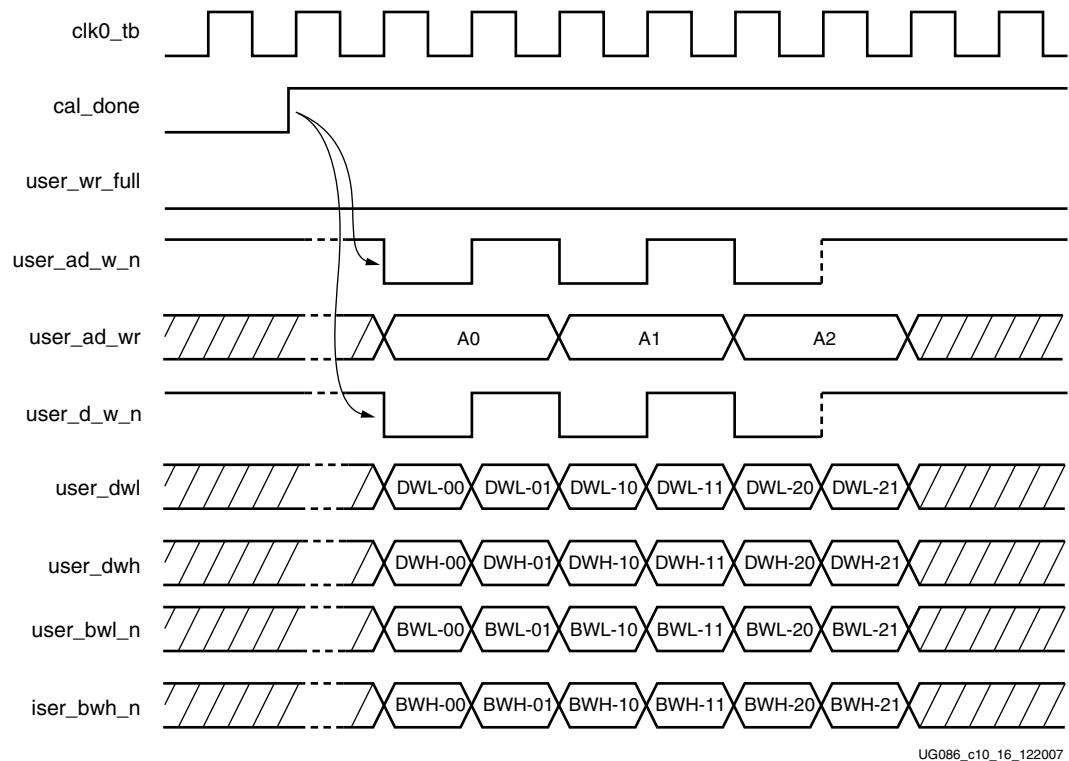
Figure 10-11: Write User Interface Block Diagram

The following steps describe the architecture of Address and Write Data FIFOs and how to perform a write burst operation to QDRII memory from user interface.

1. The user interface consists of an Address FIFO, Data FIFOs, and a Byte Write FIFO. These FIFOs are built out of Virtex-5 FIFO primitives. The Address FIFO is a FIFO36 primitive with 1K x 36 configuration. The Data FIFO is a FIFO36\_72 primitive with 512 x 72 configuration.
2. The Address FIFO is used to store the memory address where the data is to be written from the user interface. A single instantiation of a FIFO36 constitutes the Address FIFO.
3. Two separate sets of Data FIFOs are used for storing the rising-edge and falling-edge data to be written to QDRII memory from the user interface. For 9-bit, 18-bit, and 36-bit configurations, the controller pads the extra bits of the Data FIFO with 0s.
4. The Byte Write FIFO is used to store the Byte Write signals to QDRII memory from the user interface. Extra bits are padded with zeros.
5. The user can initiate a write command to memory by writing to the Write Address FIFO, Write Data FIFO, and Byte Write FIFOs when the FIFO full flags are deasserted and after the calibration done signal `cal_done` is asserted. The user should not access any of these FIFOs until `cal_done` is asserted. During the calibration process, the

controller writes pattern data into the Data FIFOs. The `cal_done` signal assures that the clocks are stable, the reset process is completed, and the controller is ready to accept commands. Status signal `user_wr_full` is asserted when the Address FIFO, Data FIFOs, or Byte Write FIFOs are full.

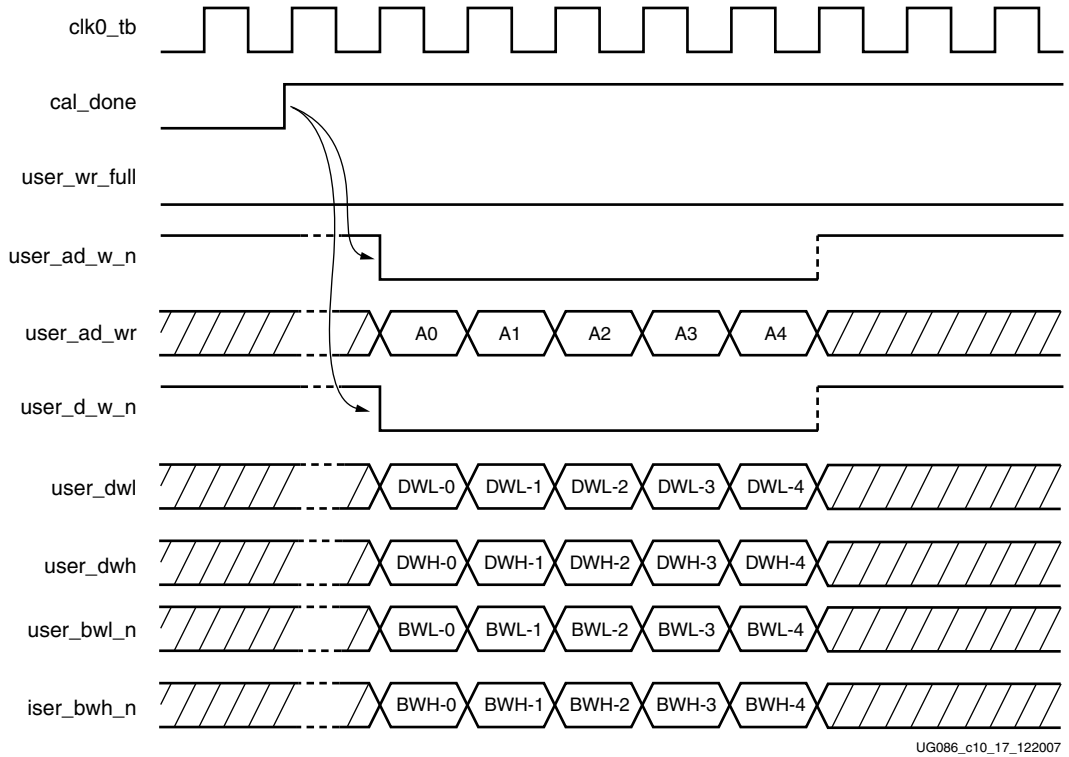
6. When signal `user_ad_w_n` is asserted, `user_ad_wr` is stored in the Address FIFO. When signal `user_d_w_n` signal is asserted, `user_dwl` and `user_dwh` are stored into the Data FIFO, and `user_bwl` and `user_bwh` are stored into the Byte Write FIFOs. For proper controller functionality, `user_ad_w_n` and `user_d_w_n` must be asserted and deasserted simultaneously.
7. The controller reads the Address, Data, and Byte Write FIFOs when they are not empty by issuing the `wr_init_n` signal. The QDRII memory write command is generated from the `wr_init_n` signal by properly timing it.



UG086\_c10\_16\_122007

Figure 10-12: Write User Interface Timing Diagram for BL = 4

8. Figure 10-12 shows the timing diagram for a write command with a burst length of four. The address should be asserted for one clock cycle as shown. For BL = 4, each write to the Address FIFO has two writes to the Data FIFO consisting of two rising-edge and two falling-edge data.
9. Figure 10-13 shows the timing diagram for a write command with a burst length of two. For BL = 2, each write to the Address FIFO has one write to Data FIFO, consisting of one rising-edge and one falling-edge data. Commands can be given in every clock when BL = 2.

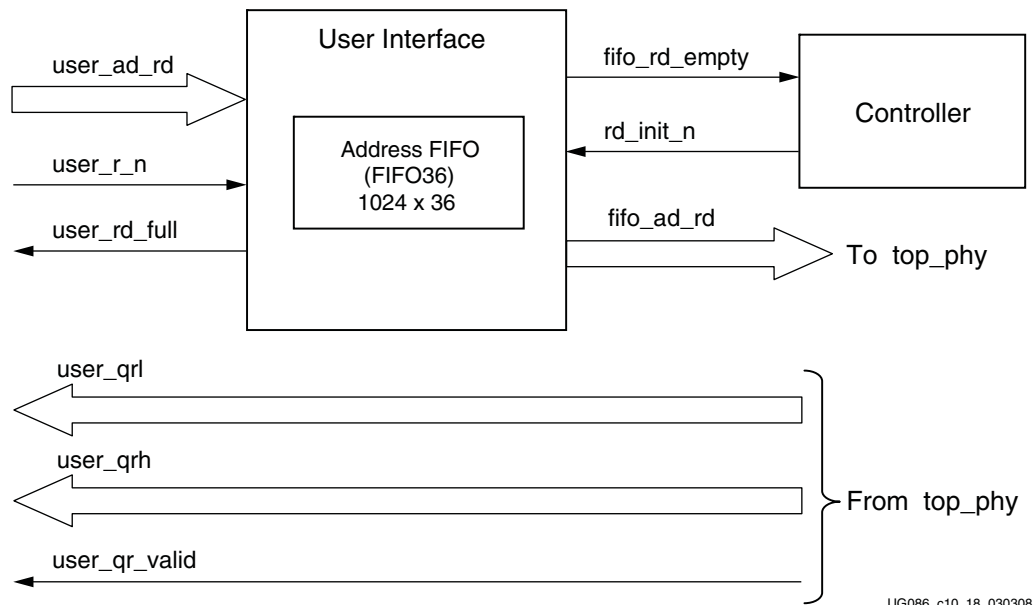


UG086\_c10\_17\_122007

Figure 10-13: Write User Interface Timing Diagram for BL = 2

### Read Interface

Figure 10-14 shows a block diagram for the read interface.



UG086\_c10\_18\_030308

Figure 10-14: Read User Interface Block Diagram

The following steps describe the architecture of the read user interface and how to perform a QDRII SRAM burst read operation.

1. The read user interface consists of an Address FIFO built out of a Virtex-5 FIFO36 of configuration 1K x 16.
2. To initiate a QDRII read command, the user writes the Address FIFO when the FIFO full flag `user_rd_full` is deasserted and the calibration done signal `cal_done` is asserted. Writing to the Address FIFO is an indication to the controller that it is a Read command. The `cal_done` signal assures that the controller clocks are stable, the internal reset process is completed, and the controller is ready to accept commands.
3. The user should issue the Address FIFO write-enable signal `user_r_n` along with read address `user_ad_rd` to write the read address to the Address FIFO.
4. The controller reads the Address FIFO when status signal `fifo_rd_empty` is deasserted and generates the appropriate control signals to QDRII memory required for a read command.
5. Prior to the actual read and write commands, the design calibrates the latency in number of clock cycles from the time the read command is issued to the time the data is received. Using this precalibrated delay information, the controller generates the user valid signal `user_qr_valid`.
6. The High state of the `user_qr_valid` signal indicates that read data is available.
7. The user must access the read data as soon as `user_qr_valid` is asserted High.
8. [Figure 10-15](#) and [Figure 10-16](#) show the user interface timing diagrams for BL = 4 and BL = 2.
9. After the read address is loaded into the Read Address FIFO, it can take a minimum of 14 clock cycles, worst case, for the controller to assert `user_qr_valid` High.

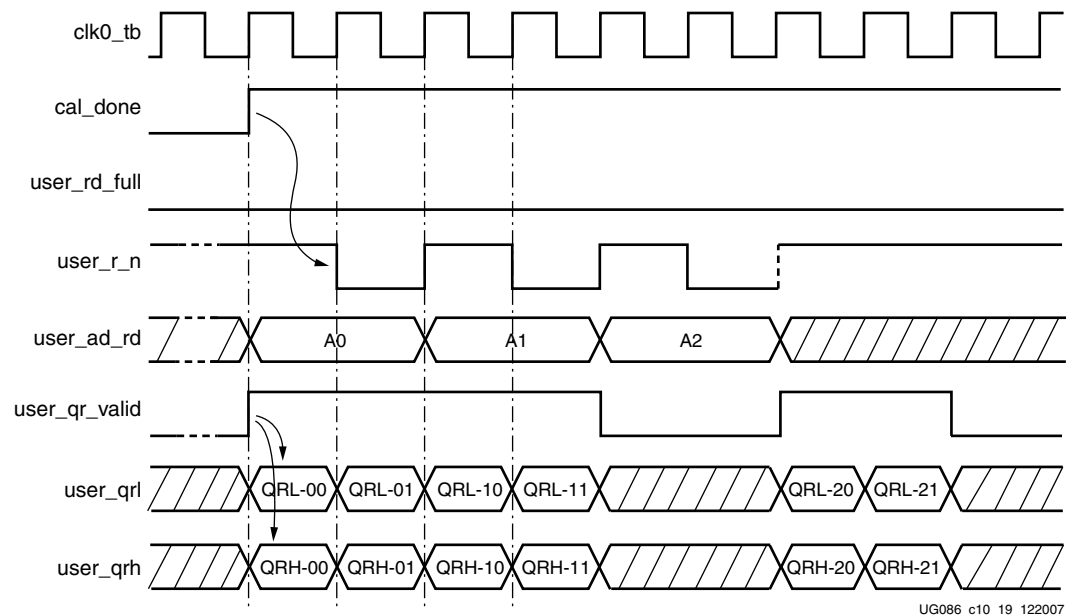


Figure 10-15: Read User Interface Timing diagram for BL = 4

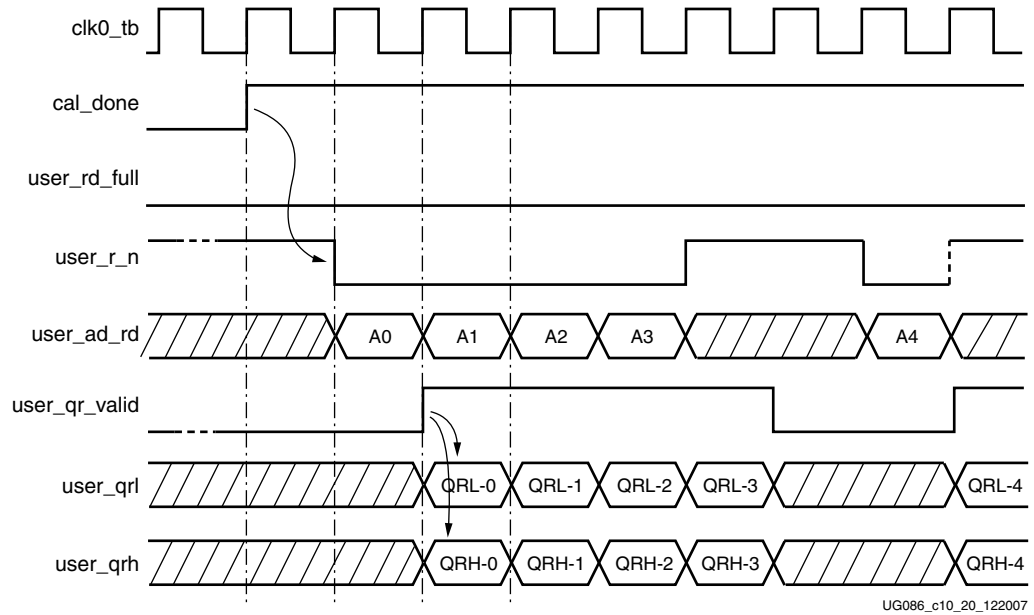


Figure 10-16: Read User Interface Timing diagram for BL = 2

Table 10-6 shows the read latency of the controller.

Table 10-6: Maximum Read Latency

Parameter	Number of Clock Cycles	Description
User read command to Read Address FIFO empty flag	6	<ul style="list-style-type: none"> <li>• 2 clock cycles for register stages</li> <li>• 4 clock cycles for empty flag deassertion in the FWFT mode</li> </ul>
Read empty flag to command to the memory	2.5	<ul style="list-style-type: none"> <li>• 1 clock cycle to generate the read command in the controller state machine</li> <li>• 1.5 cycles to transfer the command to the memory</li> </ul>
Memory read command to valid data available	5.5	<ul style="list-style-type: none"> <li>• 1.5 clock cycles of memory read latency</li> <li>• 3 clock cycles to capture and transfer read data to the FPGA clock domain</li> <li>• 1 clock cycle for aligning all the read data captured</li> </ul>
<b>Total Latency</b>	<b>14</b>	

Table 10-7 shows the list of signals for a QDRII SRAM design allocated in a group from bank selection check boxes in MIG.

Table 10-7: QDRII Signal Allocations

Bank Selected	Signals Allocated in the Group
Address	Memory address and memory control
Data_Write	Memory write data, memory byte write, and K and C clocks
Data_Read	Memory read data and memory CQ
System Control	System reset from the user interface and status signals
System_Clock	System clocks from the user interface

MIG shows check boxes for Address, Data\_Write, Data\_Read, System Control, and System\_Clock when a bank is selected for a QDRII SRAM design.

When the Address box is checked in a bank, the address, qdr\_w\_n, qdr\_r\_n, and qdr\_dll\_off\_n bits are assigned to that particular bank.

When the Data\_Write box is checked in a bank, the memory data write, memory byte write bits, the memory write clocks, and the memory input clock for the output data are assigned to that particular bank.

When the Data\_Read box is checked in a bank, the memory data read and memory read clocks are assigned to that particular bank.

When the System Control box is checked in a bank, the sys\_rst\_n, compare\_error, and cal\_done bits are assigned to that particular bank.

When the System\_Clock box is checked in a bank, the sys\_clk\_p, sys\_clk\_n, dly\_clk\_200\_p, and dly\_clk\_200\_n bits are assigned to that particular bank.

For special cases, such as without a testbench and without a DCM, the corresponding input and output ports are not assigned to any FPGA pins in the design UCF because the user can connect these ports to the FPGA pins or can connect to some logic internal to the same FPGA.

## Supported Devices

The design generated out of MIG is independent of the memory package, hence the package part of the memory component is replaced with X, where X indicates any package. Table 10-8 shows the list of components supported by MIG.

Table 10-8: Supported Devices for QDRII SRAM

Virtex-5 FPGA (Verilog and VHDL)		
Components	Make	Configuration
CY7C1314BV18-167BZXC	Cypress	x36
CY7C1315BV18-250BZC	Cypress	x36
CY7C1515V18-250BZC	Cypress	x36
K7R161882B-FC25	Samsung	x18
K7R161884B-FC25	Samsung	x18
K7R161884B-FC30	Samsung	x18
K7R163682B-FC25	Samsung	x36

Table 10-8: Supported Devices for QDRII SRAM (Continued)

Virtex-5 FPGA (Verilog and VHDL)		
Components	Make	Configuration
K7R163684B-FC25	Samsung	x36
K7R321884M-FC25	Samsung	x18
K7R321884C-FC25	Samsung	x18
K7R323682C-FC30	Samsung	x36
K7R323684M-FC25	Samsung	x36
K7R323684C-FC25	Samsung	x36
K7R641882M-FC25	Samsung	x18
K7R641884M-FC25	Samsung	x18
K7R641884M-FC30	Samsung	x18
K7R643682M-FC25	Samsung	x36
K7R643684M-FC30	Samsung	x36

## Simulating the QDRII SRAM Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains the external testbench, memory model, `.do` file, and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more details about the files in the `sim` folder and to simulate the design, see `simulation_help.chm` in the `sim` folder.

## Hardware Tested Configurations

The frequencies shown in Table 10-9 were achieved on the Virtex-5 FPGA ML561 Memory Interface Development Board under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in MIG wizard is based on combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameter for a 72-bit wide interface.

Table 10-9: Hardware Tested Configurations

FPGA Device	XC5VLX50TFF1136-2
Memory Component	K7R643684M-FC30
Data width	72
Burst Length	4
Frequency	100 MHz to 360 MHz
Flow Vendors	Synplicity and XST
Design Entry	VHDL and Verilog



## Implementing DDR SDRAM Controllers

This chapter describes how to implement DDR SDRAM interfaces for Virtex™-5 FPGAs generated by MIG. This design is based on XAPP851 [Ref 24].

### 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 11-1. 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.

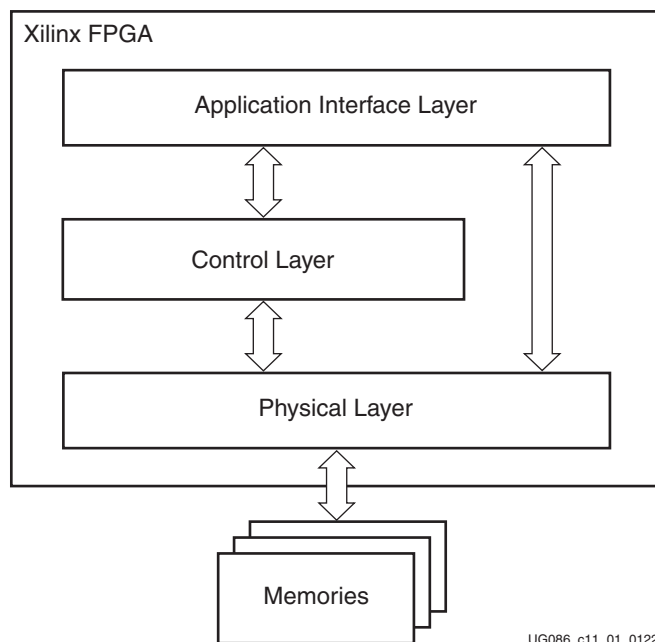


Figure 11-1: **Modular Memory Interface Representation**

## Feature Summary

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

### Supported Features

The DDR SDRAM controller design supports the following:

- Burst lengths of two, four, and eight
- Sequential and interleaved burst types
- DDR SDRAM components and DIMMs
- CAS latencies of 2, 2.5, and 3
- Verilog and VHDL
- With and without a testbench
- Bank management
- Byte-wise data masking
- Linear addressing
- With and without a DCM
- Registered DIMMs, Unbuffered DIMMs and SO-DIMMs.

The supported features are described in more detail in [“Architecture.”](#)

### Design Frequency Ranges

Table 11-1: Design Frequency Range in MHz

Memory	FPGA Speed Grade					
	-1		-2		-3	
	Min	Max	Min	Max	Min	Max
Component	100	200	100	200	100	200
DIMM	100	200	100	200	100	200

### Unsupported Features

The DDR SDRAM controller design does not support:

- Deep memories/dual rank DIMMs
- Multicontrollers

## Architecture

### Implemented Features

This section provides details on the supported features of the DDR SDRAM controller. The Virtex-5 FPGA DDR SDRAM design is a generic design that works for most of the features mentioned above. User input parameters are defined as parameters for Verilog and generics in VHDL in the design modules and are passed down the hierarchy. For example,

if the user selects a burst length of 4, then it is defined as follows in the <top\_module> module:

```
parameter BURST_LEN = 4, // burst length (in doublewords)
```

The user can change this parameter for various burst lengths to get the desired output. The same concept holds for all the other parameters listed in the <top\_module> module.

Table 11-2 lists the details of all parameters.

Table 11-2: Parameterization of DDR SDRAM Virtex-5 FPGA Design

Category	Parameter Name	Description	Other Notes	Value Restrictions
Memory Width	BANK_WIDTH	Number of memory bank address bits		
	CKE_WIDTH	Number of memory clock enable outputs		
	CLK_WIDTH	Number of differential clock outputs	Determined by the number of components/modules (one pair per component)	
	COL_WIDTH	Number of memory column bits		
	CS_BITS	$\log_2(\text{CS\_NUM})$	Used for chip-select related address decode. See notes for CS_NUM and CS_WIDTH.	
	CS_NUM	Number of separate chip selects	Different from CS_WIDTH. For example, for a 32-bit data bus with 2 x16 parts, CS_NUM = 1, but CS_WIDTH = 2 (that is, a single chip select drives two separate outputs, one for each component)	$\text{CS\_WIDTH} / \text{CS\_NUM} = \text{integer}$
	CS_WIDTH	Number of memory chip selects	Determined by the number of components/modules (one per component)	$\text{CS\_WIDTH} / \text{CS\_NUM} = \text{integer}$
	DM_WIDTH	Number of data mask bits	Can be a different value from DQS_WIDTH if x4 components are used	$(\text{DQ\_WIDTH})/8$
	DQ_BITS	$\log_2(\text{DQS\_WIDTH} * \text{DQ\_PER\_DQS})$	Used for data bus calibration decode	$(\text{DQ\_WIDTH}) / \text{Number of data bits}$
	DQ_WIDTH	Number of data bits		
	DQ_PER_DQS	Number of memory DQ data bits per strobe		
	DQS_BITS	$\log_2(\text{DQS\_WIDTH})$		
	DQS_WIDTH	Number of memory DQS strobes		
ROW_WIDTH	Number of memory address bits			
Memory Options	BURST_LEN	Burst length		(2,4,8)
	BURST_TYPE	Burst type (0: sequential, 1: interleaved)		(0,1)
	CAS_LAT	CAS latency (equal to 25 for CL = 2.5)		(2,25,3)
	MULTI_BANK_EN	Bank management enable	If enabled, up to four banks are kept open; otherwise, one bank is kept open	(0,1)
	REDUCE_DRV	Reduced strength memory I/O enable. Set (1) for reduced I/O drive strength.	Not supported for all DDR/DDR2 widths	(0,1)
	REG_ENABLE	Set (1) for registered memory module	Accounts for an extra clock cycle delay on address/control for a registered module	(0,1)

Table 11-2: Parameterization of DDR SDRAM Virtex-5 FPGA Design (Continued)

Category	Parameter Name	Description	Other Notes	Value Restrictions
Memory Timing	$T_{REFL\_NS}$	Auto refresh interval (in ns)	Take directly from memory data sheet	
	$T_{RAS}$	Active to precharge delay (in ps)	Take directly from memory data sheet	
	$T_{RCD}$	Active to read/write delay (in ps)	Take directly from memory data sheet	
	$T_{RFC}$	Refresh to refresh, refresh to active delay (in ps)	Take directly from memory data sheet	
	$T_{RP}$	Precharge to command delay (in ps)	Take directly from memory data sheet	
	$T_{WR}$	Used to determine write to precharge (in ps)	Take directly from memory data sheet	
	$T_{WTR}$	Write to read (in ps)	Take directly from memory data sheet	
Miscellaneous	CLK_PERIOD	Memory clock period (in ps)	Used for PHY calibration and DCM (if applicable) setting	
	DLL_FREQ_MODE	DCM Frequency Mode	Determined by CLK_PERIOD. Needed only if the DCM option is selected.	("LOW", "HIGH")
	DDR2_ENABLE	Select either DDR or DDR2 interface (equal to 1 for DDR2)	Provided from the mem_if_top level and below	(0,1)
	SIM_ONLY	Enable bypass of 200 $\mu$ s power-on delay		(0,1)
	RST_ACT_LOW	Indicates the polarity of the input reset signal (sys_rst_n)	1: Reset is active Low. 0: Reset is active High.	(0,1)

## 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. The burst length determines the maximum number of column locations accessed for a given READ or WRITE command. The DDR SDRAM ctrl module implements a burst length that is programmed.

## 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. CAS latency is implemented in the ctrl module. For CL = 2.5, the input value is read as "25" in the design. During read data operations, the generation of the read\_en signal varies according to the CL in the ctrl module.

## Precharge

The PRECHARGE command is used to close the open row in a bank if there is a command to be issued in the same bank. The Virtex-5 DDR controller issues a PRECHARGE command only if there is already an open row in the particular bank where a read or write command is to be issued, thus increasing the efficiency of the design. The auto-precharge function is not supported in this design. This design ties the A10 bit Low during normal reads and writes.

## Data Masking

Virtex-5 DDR SDRAM controllers support bitwise data masking of the data bits during a write operation. For x4 components, data masking cannot be done on a per nibble basis due to an internal block RAM based FIFO limitation. The mask data is stored into the FIFOs along with the write data.

## Auto Refresh

An AUTO REFRESH command is issued to the DDR memory at specified intervals of time to refresh the charge to retain the data.

## Bank Management

Bank management is done by the Virtex-5 DDR SDRAM controller design to increase the efficiency of the design. The controller keeps track of whether the bank being accessed already has an open row or not, and also decides whether a PRECHARGE command should be issued or not to that bank. When bank management is enabled via the MULTI\_BANK\_EN parameter, a maximum of four banks/rows can open at any one time. A least-recently-used (LRU) algorithm is employed to keep the three banks most recently used. It closes the bank least recently used when a new bank/row location needs to be accessed. The bank management feature can also be disabled by clearing MULTI\_BANK\_EN. In this case, only one bank is kept open at any one time.

## Linear Addressing

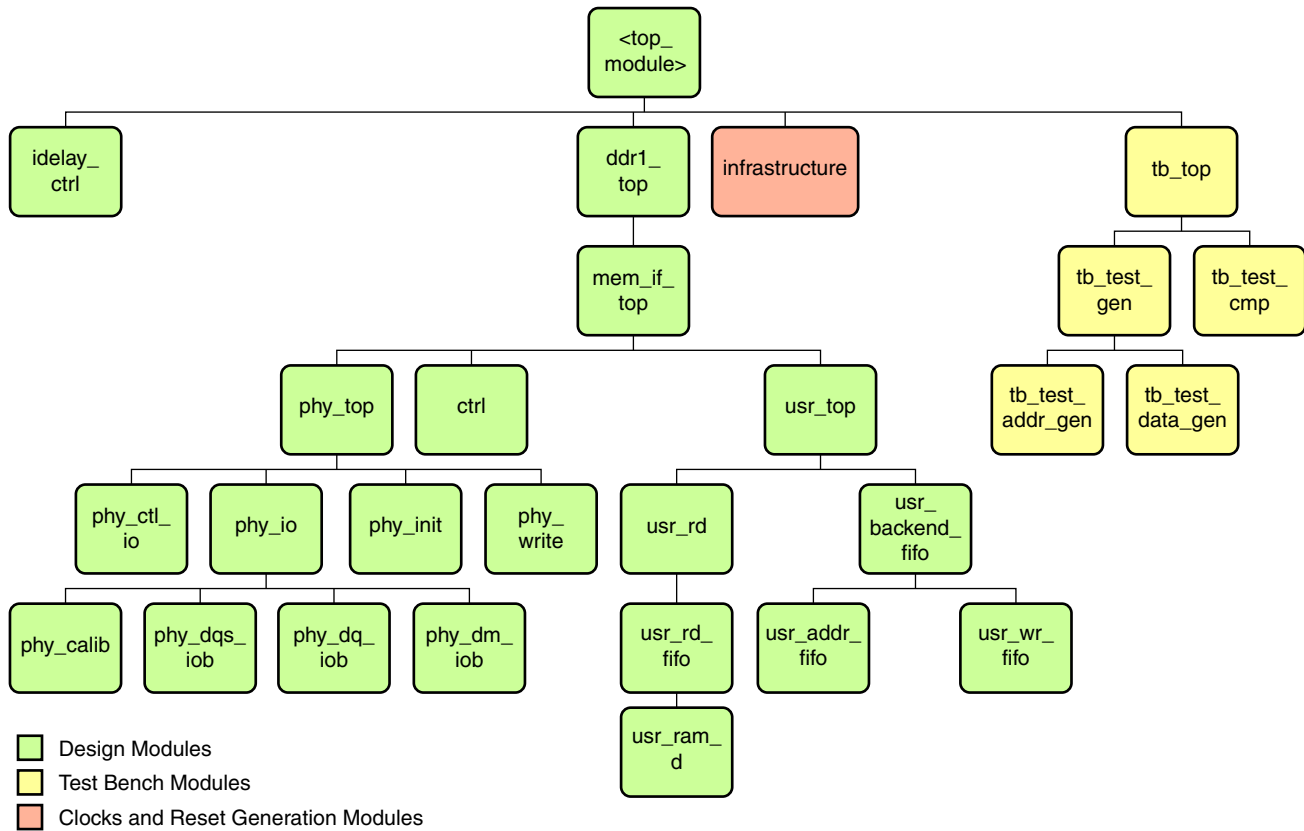
Linear addressing refers to the way the user provides the address of the memory to be accessed. For Virtex-5 DDR SDRAM controllers, the user provides the address information through the app\_af\_addr signal. As the densities of the memory devices vary, the number of column address bits and row address bits also change. In any case, the row address bits in the app\_af\_addr signal always start from the next higher bit where the column address ends. This feature increases the coverage of more devices that can be supported with the design.

## Different Memories (Density/Speed)

The DDR SDRAM controller supports different densities. For DDR components shown in MIG, densities can vary from 128 Mb to 1 Gb. The user can select the various configurations from the "Create custom part" option; the supported maximum column address is 13, the maximum row address is 15, and the maximum bank address is 2. The design can decode write and read addresses from the user in the DDR SDRAM ctrl module. The user address consists of column, row, and bank addresses.

## Hierarchy

Figure 11-2 shows the hierarchical structure of the design generated by MIG with a DCM and a testbench.



UG086\_c11\_02\_091707

Figure 11-2: Hierarchical Structure of Virtex-5 DDR SDRAM Design

The modules are classified in three types:

- Design modules
- Testbench modules
- Clock and reset generation modules

For designs without a testbench, the correspondingly shaded modules are not present. In this case, the user interface signals appear in the <top\_module> module. [Table 11-3, page 361](#) provides a list of these signals.

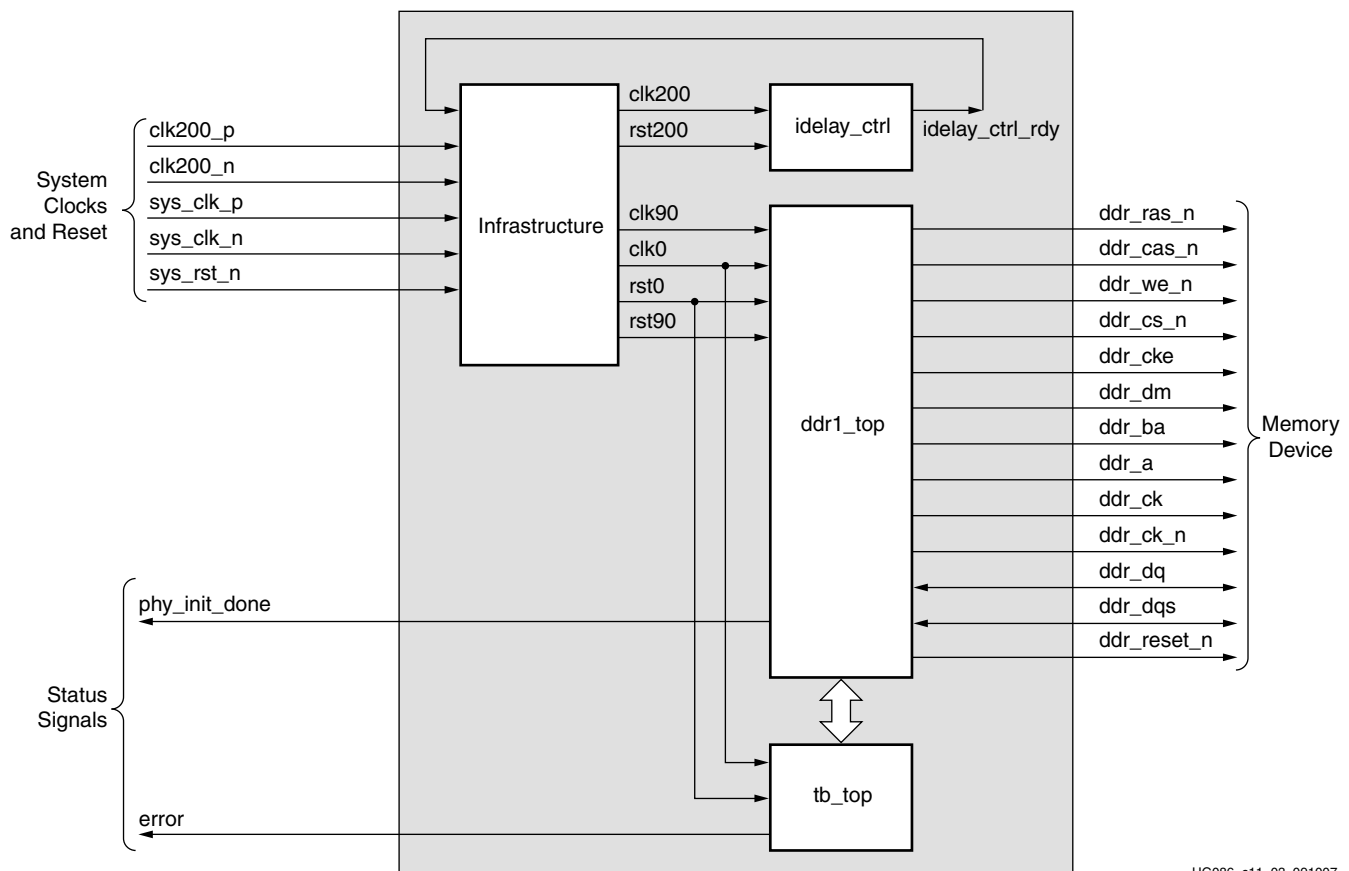
The infrastructure module generates the clock and reset signals for the design. It instantiates a DCM when MIG generates a design with a DCM. The inputs to this module are the differential design clock and a 200 MHz differential clock for the IDELAYCTRL module. A user reset is also input to this module. Using the input clocks and reset signals, the system clocks and the system reset are generated in this module, which is used in the design.

If the design has no DCM, the DCM primitive is not instantiated in the module. Instead, the system operates on the user-provided clocks. A system reset is also generated in the infrastructure module using the input DCM\_LOCK signal.

## MIG Design Options

MIG provides various options to generate the design with or without a testbench or with or without a DCM. This section provides detailed descriptions of the different types of designs the user can generate using the MIG options.

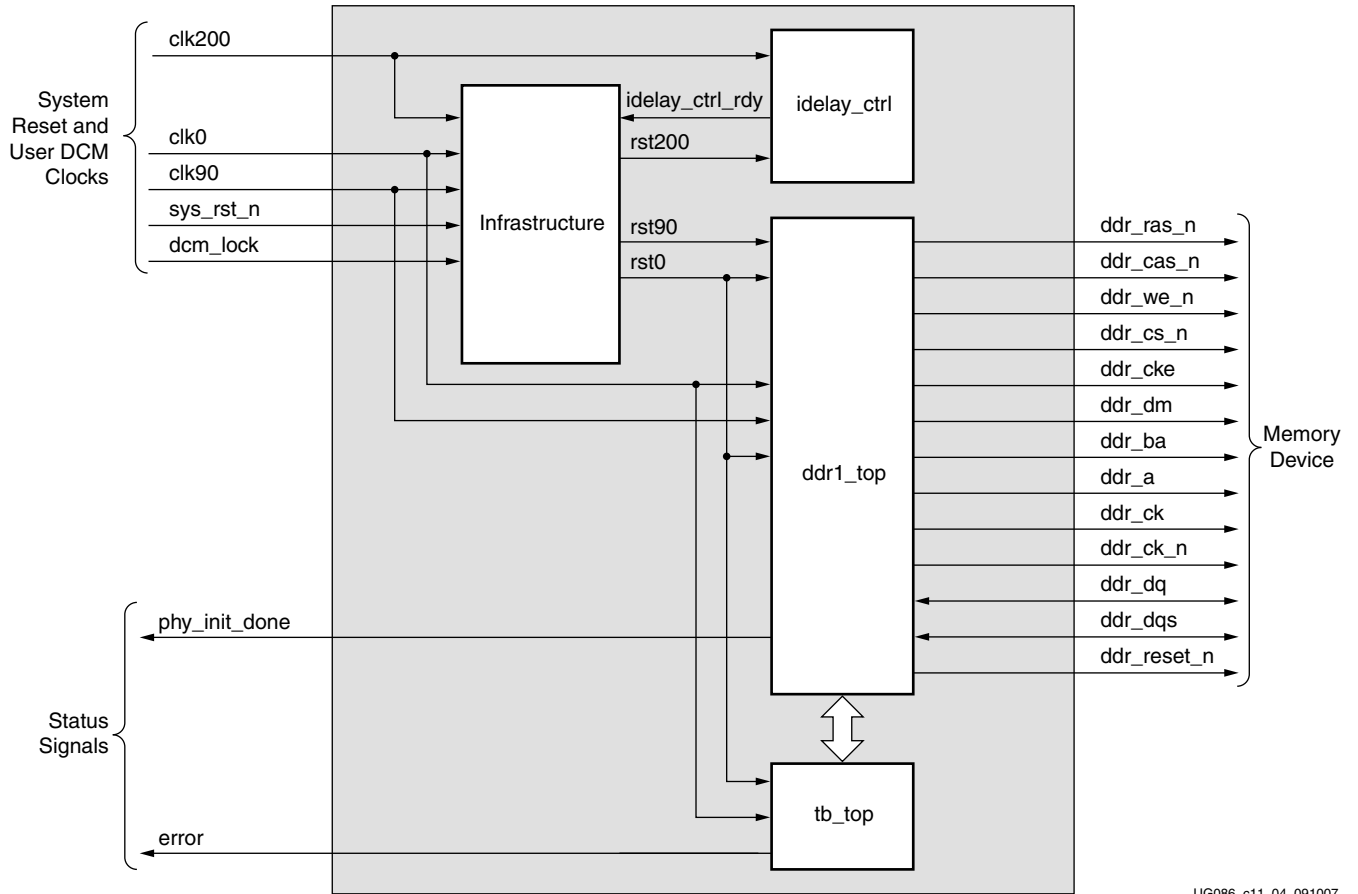
Figure 11-3 shows a block diagram representation of the top-level module for a design with a DCM and a testbench. The inputs consist of differential clocks for the design and Idelayctrl modules and the user reset. The error output signal indicates whether the case passes or fails. The phy\_init\_done signal indicates the completion of initialization and calibration of the design. Because the DCM is instantiated in the infrastructure module, it generates the required clocks and reset signals for the design.



UG086\_c11\_03\_091007

Figure 11-3: Top-Level Block Diagram of the DDR SDRAM Design with a DCM and a Testbench

Figure 11-4 shows a block diagram representation of the top-level module for a design with a testbench but without a DCM. The inputs consist of user clocks for the design and Idelayctrl modules and the user reset. The design uses the user input clocks. These clocks should be single-ended. The infrastructure module uses the input reset and `dcm_lock` signals to reset the design. The user application must have a DCM primitive instantiated in the design. The error output signal indicates whether the case passes or fails. The `phy_init_done` signal indicates the completion of initialization and calibration of the design.

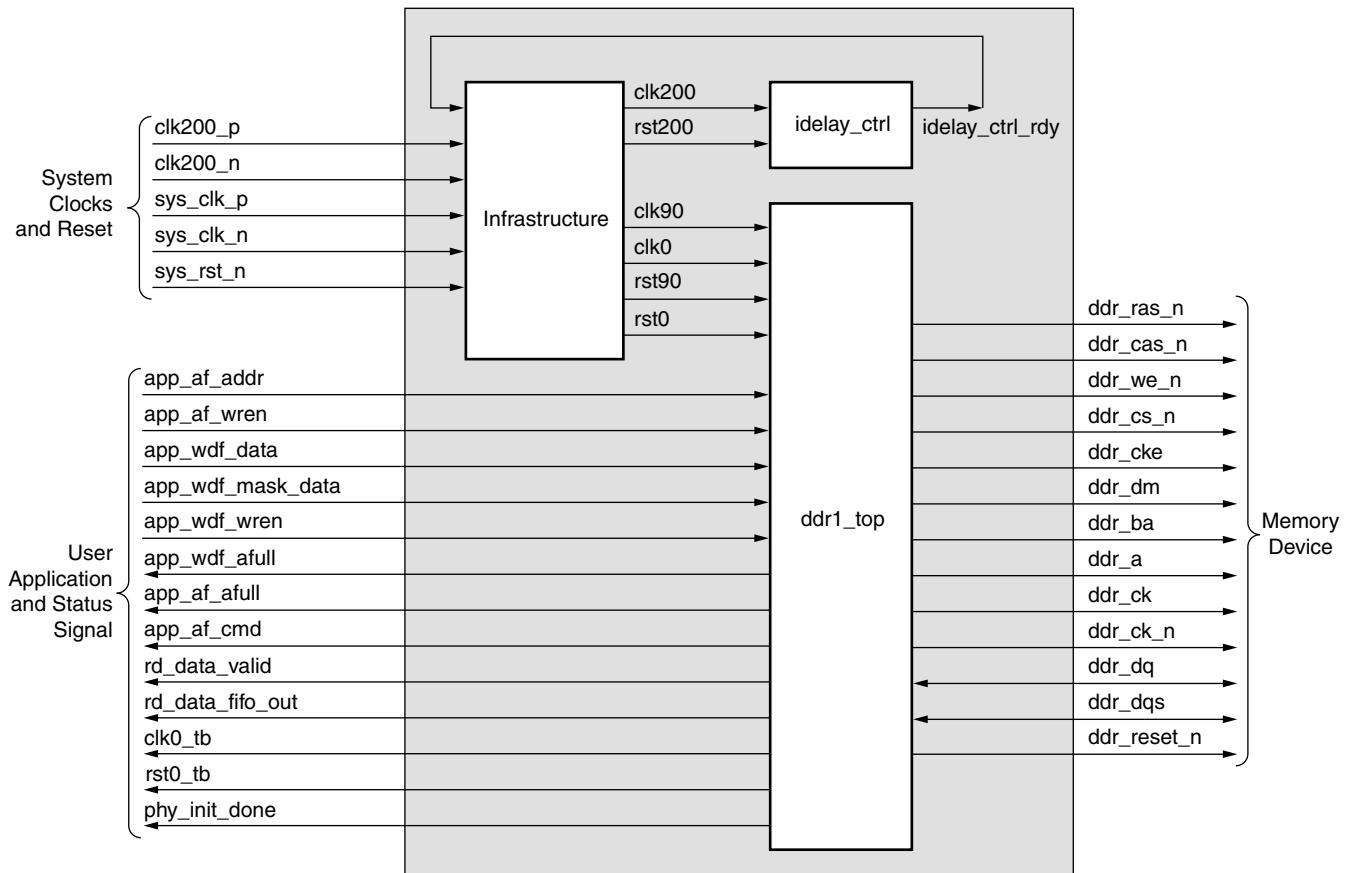


UG086\_c11\_04\_091007

Figure 11-4: Top-Level Block Diagram of the DDR SDRAM Design with a Testbench but without a DCM



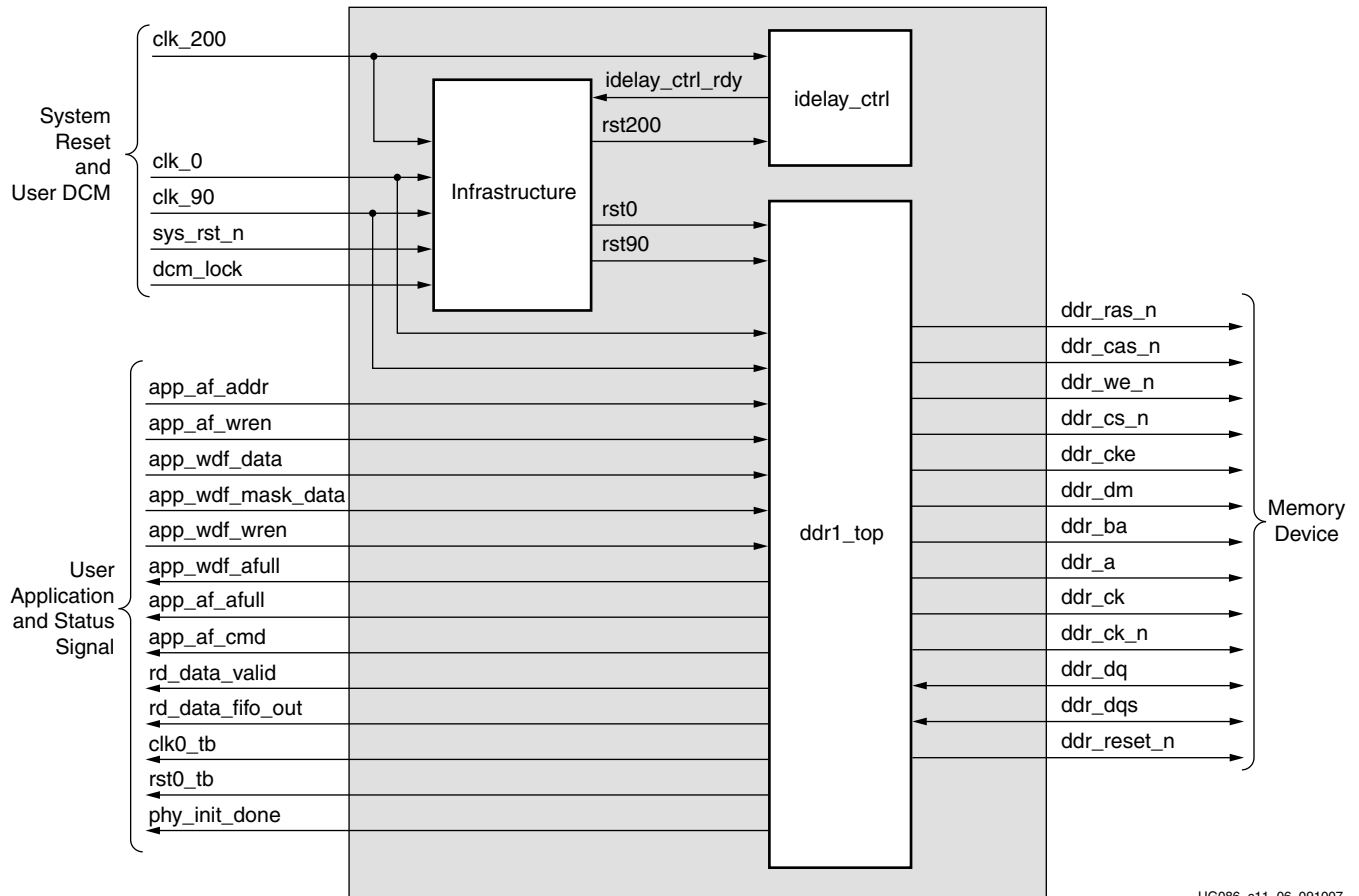
Figure 11-5 shows a block diagram representation of the top-level module for a design with a DCM but without a testbench. The `phy_init_done` signal indicates the completion of initialization and calibration of the design. The user interface signals are also listed in the `<top_module>` module. The design provides the `clk_tb` and `reset_tb` signals to the user to synchronize with the design. Because the DCM is instantiated in the infrastructure module, it generates the required clock and reset signals for the design.



UG086\_c11\_05\_091007

Figure 11-5: Top-Level Block Diagram of the DDR SDRAM Design with a DCM but without a Testbench

Figure 11-6 shows a block diagram representation of the top-level module for designs without a DCM or a testbench. The inputs consist of user clocks for the design and Idelayctrl modules and the user reset. The design uses the user input clocks. These clocks should be single-ended. To reset the design, the signals are generated using the input reset and the dcm\_lock signals in the infrastructure module. The user application must have a DCM primitive instantiated in the design. The phy\_init\_done signal indicates the completion of initialization and calibration of the design. The user interface signals are also listed in the <top\_module> module. The design provides the clk\_tb and reset\_tb signals to the user to synchronize with the design.



UG086\_c11\_06\_091007

Figure 11-6: Top-Level Block Diagram of the DDR SDRAM Design without a DCM or a Testbench

Figure 11-7 shows an expanded block diagram of the design. The design's top module is expanded to show various internal blocks. The functions of these blocks are explained in following subsections.

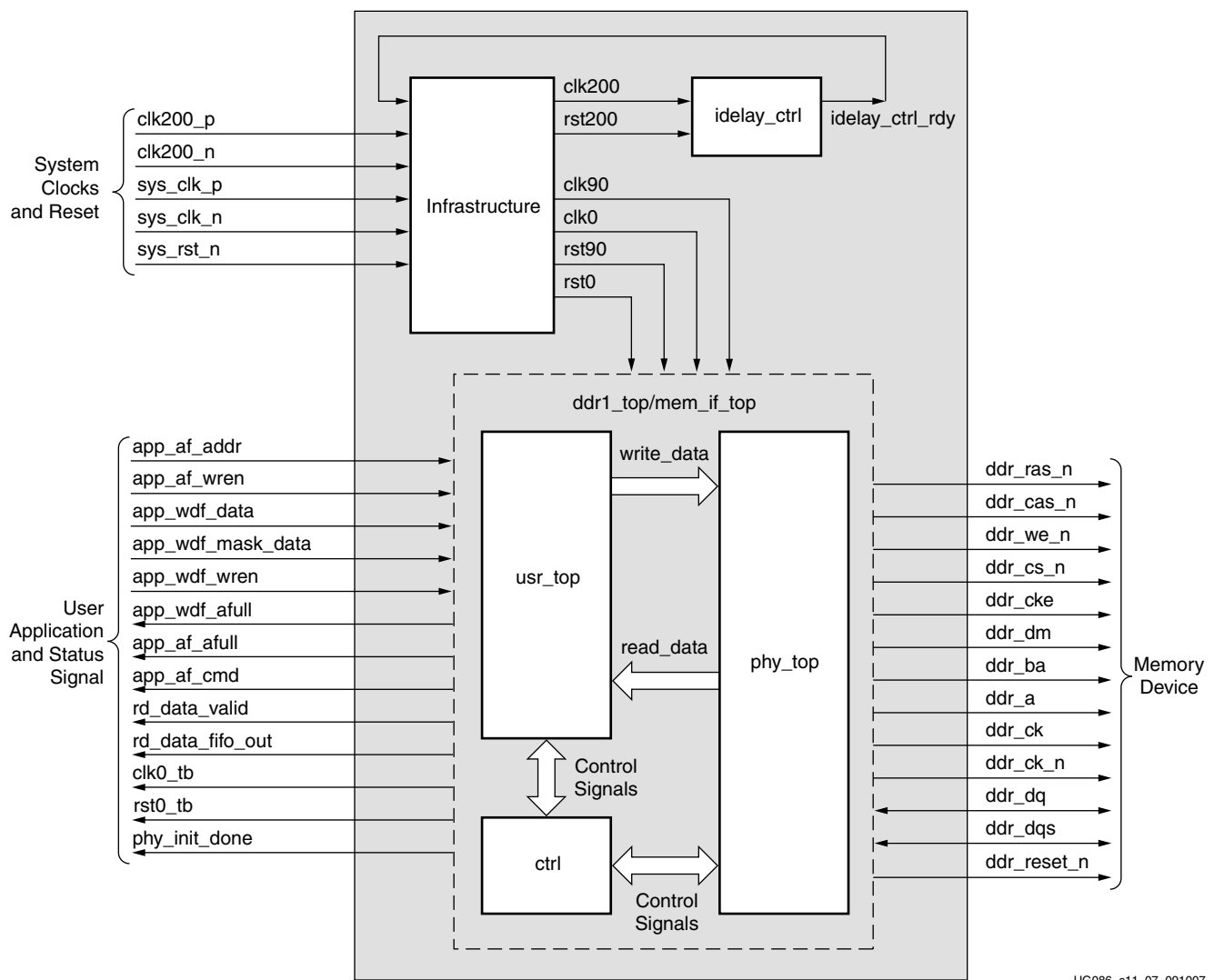


Figure 11-7: Detailed Block Diagram of the DDR SDRAM Design with a DCM but without a Testbench

## Infrastructure

The infrastructure module generates the clock and reset signals for the design. The user clocks and user reset are input to this module. In designs generated with a DCM, the input clocks are differential. There are clocks for design use and also a 200 MHz clock for the idelayctrl primitive. These differential clocks are first passed through the buffers, and the single-ended output of the buffers is used. The single-ended output of sys\_clk\_p and sys\_clk\_n is then given to the DCM input. The clock outputs of the DCM are clk0 and clk90. The dcm\_lock signal and user reset input are used to generate the synchronous system resets for the design. After the DCM is locked, the design is in the reset state for at least 25 clocks.

When the user chooses the no DCM option in the GUI, the design does not use any DCM primitives. Instead it works on the clocks provided by the user. The input clocks in this

case have to be single-ended. The `dcm_lock` status and user input reset signals are the inputs to the module when there is no DCM. These signals are used to generate the synchronous system resets for the design.

## idelay\_ctrl

This module instantiates the `IDELAYCTRL` primitive of the Virtex-5 FPGA. The `IDELAYCTRL` primitive is used to continuously calibrate the individual delay elements in its region to reduce the effect of process, temperature, and voltage variations. A 200 MHz clock has to be fed to this primitive.

## ctrl

The `ctrl` module is the main controller of the Virtex-5 DDR SDRAM controller design. It generates all the control signals required for the DDR memory interface and the user interface. This module signals the FIFOs instantiated in the user interface to output the fed data in it and also signals the physical layer to output the data on the IOBs during a write operation. During a read operation, the data read from the memory is taken from the physical layer and written into the user interface FIFOs using the control signals generated by the `ctrl` module.

The `ctrl` module decodes the user command and issues the specified command to the memory. The `app_af_cmd` signal is decoded as a write command when it equals `3'b000`, and `app_af_cmd` is decoded as a read command when it equals `3'b001`. The commands and control signals are generated based on the input burst length and CAS latency. If the multi-bank option is enabled, the `ctrl` module also takes care of bank management, so as to increase the efficiency of the design. At a given point of time, a maximum of four banks can be open. The controller issues a `PRECHARGE` command to the bank only if there is already an open row in that bank and the next command is to be issued to a different row. An `ACTIVE` command is generated to open the row in that particular bank. Thus the efficiency is increased.

## phy\_top

The `phy_top` module is the top level of the physical interface of the design. The physical layer includes the input/output blocks (IOBs) and other primitives used to read and write the double data rate signals to and from the memory, such as `IDDR` and `ODDR`. This module also includes the `IODELAY` elements of the Virtex-5 FPGA. These `IODELAY` elements are used to delay the input strobe and data signals to capture the valid data into the Read Data FIFO.

The memory control signals, such as `RAS_N`, `CAS_N`, and `WE_N`, are driven from the buffers in the IOBs. All the input and output signals to and from the memory are referenced from the IOB to compensate for the routing delays inside the FPGA.

The `phy_init` module, which is instantiated in the `phy_top` module, is used to initialize the DDR memory in a predefined sequence according to the JEDEC standard for DDR SDRAM.

The `phy_calib` module calibrates the design to align the strobe signal such that it always captures the valid data in the FIFO. This calibration is needed to compensate for the trace delays between the memory and the FPGA devices.

The `phy_write` module splits the user data into rise data and fall data to be sent to the memory as a double data rate signal using `ODDR`. Similarly, while reading the data from memory, the data from `IDDR` is combined to get a single vector that is written into the read FIFO.

## usr\_top

The `usr_top` module is the user interface block of the design. It receives and stores the user data, command, and address information in respective FIFOs. The `ctrl` module generates the required control signals for this module. During a write operation, the data stored in the `usr_wr_fifo` is read and given to the physical layer to output to the memory. Similarly, during a read operation, the data from the memory is read via IDDR and written into the FIFOs. This data is given to the user with a valid signal (`rd_data_valid`), which indicates valid data on the `rd_data_fifo_out` signal. See “[User Interface Accesses](#),” page 365 for required timing requirements and restrictions for user interface signals.

Table 11-3 lists the user interface signals.

Table 11-3: User Interface Signals

Signal	Direction <sup>(1)</sup>	Description
<code>app_af_cmd[2:0]</code> <sup>(2)</sup>	Input	3-bit command to the Virtex-5 DDR SDRAM design. <code>app_af_cmd = 3'b000</code> for write commands <code>app_af_cmd = 3'b001</code> for read commands Operation is not guaranteed if the user gives values other than the specified ones.
<code>app_af_addr[30:0]</code> <sup>(2, 3)</sup>	Input	Provides the address, row address, and column address of the memory location to be accessed. Column address = <code>app_af_addr[COL_WIDTH-1: 0]</code> Row address = <code>app_af_addr[ROW_WIDTH+COL_WIDTH -1: COL_WIDTH]</code> Bank address = <code>app_af_addr[BANK_WIDTH+ROW_WIDTH+COL_WIDTH-1: ROW_WIDTH+COL_WIDTH]</code>
<code>app_af_wren</code> <sup>(2)</sup>	Input	Write enable to the user address FIFO. This signal should be synchronized with the <code>app_af_addr</code> and <code>app_af_cmd</code> signals.
<code>app_wdf_data[2*DQ_WIDTH-1:0]</code> <sup>(2)</sup>	Input	User input data. It should have the fall data and the rise data. Rise data = <code>app_wdf_data[DQ_WIDTH-1: 0]</code> Fall data = <code>app_wdf_data[2*DQ_WIDTH-1: DQ_WIDTH]</code>
<code>app_wdf_wren</code> <sup>(2)</sup>	Input	Write enable for the user write FIFO. This signal should be synchronized with the <code>app_wdf_data</code> and <code>app_wdf_mask_data</code> signals.
<code>app_wdf_mask_data[2*DM_WIDTH-1: 0]</code> <sup>(2)</sup>	Input	User mask data. It should contain the masking information for both rise and fall data. Rise mask data = <code>app_wdf_mask_data[DM_WIDTH-1: 0]</code> Fall mask data = <code>app_wdf_mask_data[2*DM_WIDTH-1: DM_WIDTH]</code>
<code>app_af_afull</code> <sup>(2)</sup>	Output	Almost Full status of the address FIFO. The user can write 12 more locations into the FIFO after <code>app_af_afull</code> is asserted.
<code>app_wdf_afull</code> <sup>(2)</sup>	Output	Almost Full status of the user write FIFO. The user can write 12 more locations into the FIFO after <code>app_wdf_afull</code> is asserted.
<code>rd_data_fifo_out[2*DQ_WIDTH-1: 0]</code> <sup>(2)</sup>	Output	Read data from the memory. Read data is stored in the user write FIFO.
<code>rd_data_valid</code> <sup>(2)</sup>	Output	Status signal indicating that data read from the memory is available to the user.
<code>clk0_tb</code>	Output	Clock output to the user. All the user input data and commands must be synchronized with this clock.
<code>rst0_tb</code>	Output	Active-High reset for the user interface.

**Notes:**

1. The direction indicated in this table is referenced from the design perspective. For example, input indicates that the signal is input to the design and output for the user.
2. See “[User Interface Accesses](#),” page 365 for required timing requirements and restrictions for the user interface signals.
3. Addressing in the Virtex-5 FPGA is linear. That is, the row address bits immediately follow the column address bits, and the bank address bits follow the row address bits, thus supporting more devices.

Table 11-4: Design Status Signals

Signal	Direction	Description
phy_init_done	Output	Indicates the completion of initialization and calibration of the design.

## System Interface Signals

Table 11-5 and Table 11-6 shows the system interface signals for designs with and without a DCM, respectively.

Table 11-5: System Interface Signals with a DCM

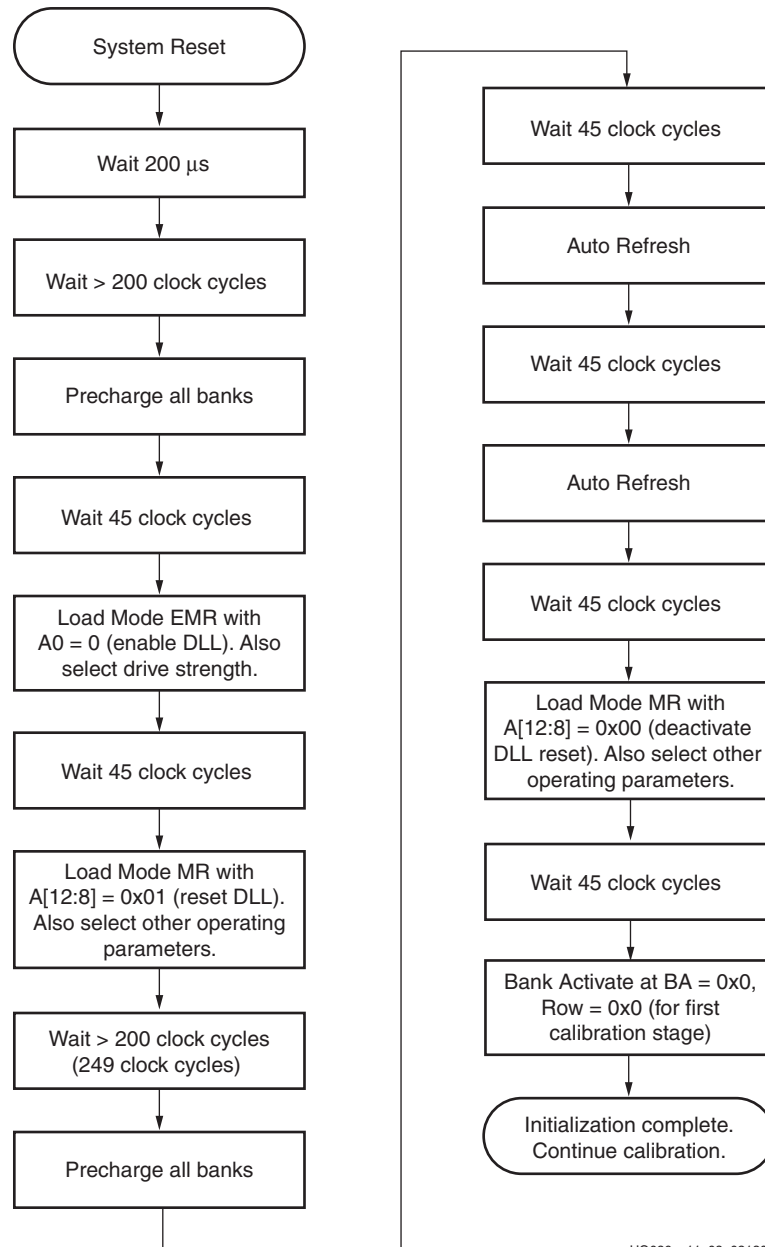
Signal	Direction	Description
sys_clk_p, sys_clk_n	Input	Differential input clocks to the DCM. The DDR SDRAM controller and memory operate on this clock.
sys_rst_n	Input	Active-Low reset to the DDR SDRAM controller.
clk200_p, clk200_n	Input	200 MHz input differential clock for the IDELAYCTRL primitive of Virtex-5 FPGA.

Table 11-6: System Interface Signals without a DCM

Signal	Direction	Description
clk0	Input	The DDR SDRAM controller and memory operate on this clock.
sys_rst_n	Input	Active-Low reset to the DDR SDRAM controller. This signal is used to generate a synchronous system reset.
clk90	Input	90° phase-shifted clock with the same frequency as clk0.
clk200	Input	200 MHz input differential clock for the IDELAYCTRL primitive of the Virtex-5 FPGA.
dcm_lock	Input	The status signal indicating whether the DCM is locked or not. This signal is used to generate a synchronous system reset.

## DDR SDRAM Initialization

DDR memory is initialized through a specified sequence as shown in [Figure 11-8](#). This initialization sequence is in accordance with JEDEC specifications for DDR SDRAMs. The initialization logic is implemented in the physical layer.

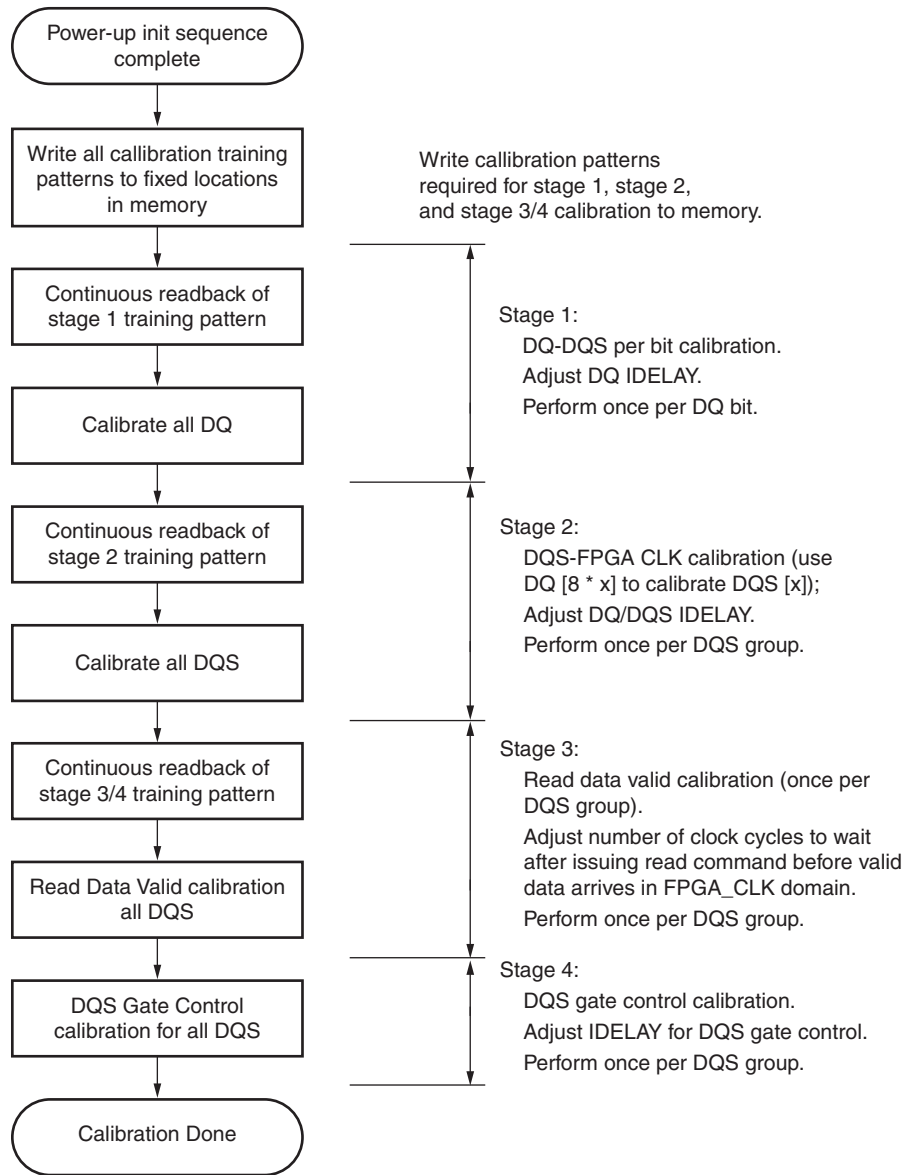


UG086\_c11\_08\_021307

Figure 11-8: DDR SDRAM Initialization

## DDR SDRAM Design Calibration

Before issuing user read and write commands, the design is calibrated to ensure that correct data is captured in the ISERDES primitives of Virtex-5 FPGAs. Calibration logic is implemented in the physical layer of the design. Figure 11-9 shows the overall calibration sequence. For more details on the calibration algorithm for the Virtex-5 DDR interface, see XAPP851 [Ref 24].



UG086\_c9\_08\_020507

Figure 11-9: Overall Design Calibration Sequence

The first calibration stage sets the IDELAY value for each DQ (IDELAY for DQS remains at 0 during this time), and is performed even before a phase relationship between DQS and FPGA\_CLK has been established. A training pattern of “10” (1 = rising, 0 = falling) is used to calibrate DQ.

The second calibration stage includes calibration between the DQS and the FPGA clock.



The third calibration stage is read-enable calibration, which compensates for the round-trip delay between when the read command is issued by the controller, and the captured read data is valid at the outputs of the ISERDES.

The fourth stage includes calibration of a squelch circuit that gates the input DQS to avoid the glitch that propagates to the second rank of flops in the ISERDES. The glitch occurs when DQS goes from the Low state to the 3-state level after the last edge of the DQS, which might cause a “false” rising and/or falling edge on the DQS input to the FPGA. Unless the DQS glitch is gated after the last DQS falling edge of a read burst, the data registered in the ISERDES might change prematurely. During calibration, an auto-refresh command is issued to memory at intervals depending on the stage of calibration.

After initialization and calibration is done, the controller is signaled to start normal operation of the design. Now, the controller can start issuing user write and read commands to the memory.

## User Interface Accesses

The user backend logic communicates with the memory controller through a synchronous FIFO-based user interface. This interface consists of three related buses:

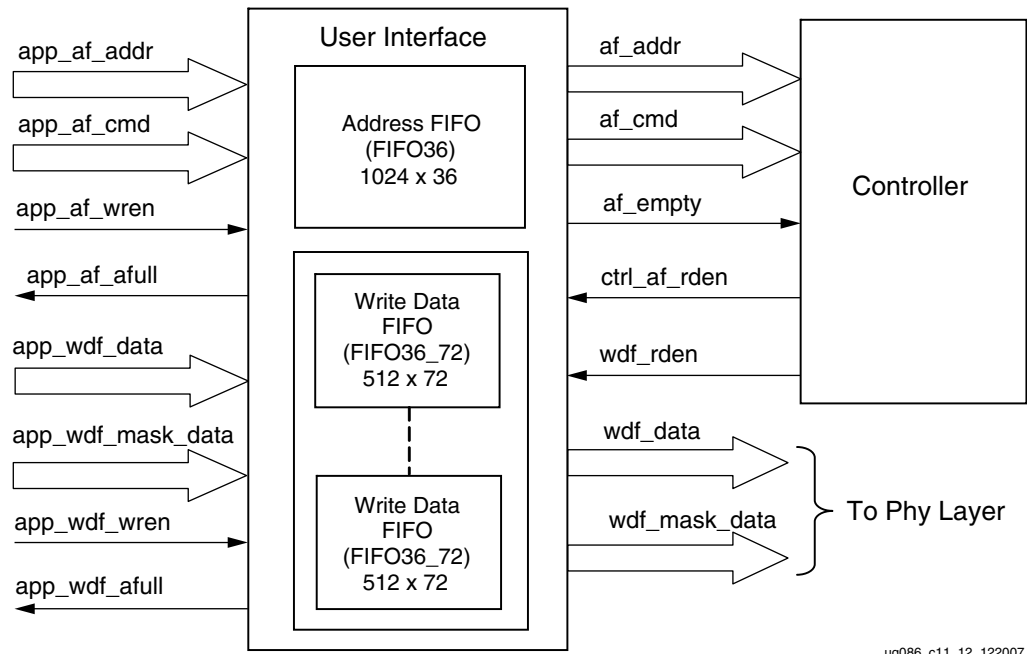
- a command/address FIFO bus accepts write/read commands as well as the corresponding memory address from the user
- a Write Data FIFO bus that accepts the corresponding write data when the user issues a write command on the command/address bus
- a read bus on which the corresponding read data for an issued read command is returned

The user interface has the following timing and signaling restrictions:

1. Commands and write data cannot be written by the user until calibration is complete (as indicated by `phy_init_done`). In addition, the following interface signals need to be held Low until calibration is complete: `app_af_wren`, `app_wdf_wren`, `app_wdf_data[]`, `app_wdf_mask_data[]`. Failure to hold these signals Low causes errors during calibration. This restriction arises from the fact that the Write Data FIFO is also used during calibration to hold the training patterns for the various stages of calibration.
2. When issuing a write command, the first write data word must be written to the Write Data FIFO either prior to, or on the same clock cycle as the when the write command is issued. In addition, the write data must be written by the user over consecutively clock cycles, there cannot be a break in between words. These restrictions arise from the fact that the controller assumes write data is available when it receives the write command from the user.
3. The output of the Read Data FIFO (specifically, the `rd_data_fifo_out` and `rd_data_valid` signals) are synchronous to `clk90`, and not to `clk0`. The user might need to insert an extra pipeline stage to resynchronize the data to `clk0` if place-and-route timing cannot be met on these 3/4 cycle paths.

## Write Interface

Figure 11-10 shows the user interface block diagram for write operations.



ug086\_c11\_12\_122007

Figure 11-10: User Interface Block Diagram for Write Operations

The following steps describe the architecture of the Address and Write Data FIFOs and show how to perform a write burst operation to DDR SDRAM from the user interface.

1. The user interface consists of an Address FIFO and a Write Data FIFO. The Write Data FIFO is constructed using Virtex-5 FIFO36\_72 primitive with a 512 x 72 configuration. The 72-bit architecture comprises one 64-bit port and one 8-bit port. For Write Data FIFOs, the 64-bit port is used for data bits and the 8-bit port is used for mask bits. Mask bits are available only when supported by the memory part and when the Data Mask is enabled in the MIG GUI. Some memory parts, such as Registered DIMMs of x4 parts, do not support mask bits.
2. The Address FIFO is constructed using Virtex-5 FIFO36 primitive with a 1024 x 36 configuration. The 36-bit architecture comprises one 32-bit port and one 4-bit port. The 32-bit port is used for addresses (app\_af\_addr), and the 4-bit port is used for commands (app\_af\_cmd).
3. The Address FIFO is common for both Write and Read commands. It comprises an address part and the command part. Command bits discriminate between write and read commands.
4. The user interface data width app\_wdf\_data is twice that of the memory data width. For an 8-bit memory width, the user interface is 16 bits consisting of rising edge data and falling edge data. For every 8 bits of data, there is a mask bit. For 72-bit memory data, the user interface data width app\_wdf\_data is 144 bits, and the mask data app\_wdf\_mask\_data is 18 bits.
5. The minimum configuration of the Write Data FIFO is 512 x 72 for a memory data width of 8 bits. For an 8-bit memory data width, the least-significant 16 bits of the data port are used for write data and the least-significant two bits of the 8-bit port are used

for mask bits. The controller internally pads all zeros for the most-significant 48 bits of the 64-bit port and the most-significant six bits of the 8-bit port.

6. Depending on the memory data width, MIG instantiates multiple FIFO36\_72s to gain the required width. For designs using 8-bit to 32-bit data width, one FIFO36\_72 is instantiated; for 72-bit data width, a total of three FIFO36\_72s are instantiated. The bit architecture comprises 32 bits of rising-edge data, 4 bits of rising-edge mask, 32 bits of falling-edge data, and 4 bits of falling-edge mask, which are all stored in a FIFO36\_72. MIG routes the `app_wdf_data` and `app_wdf_mask_data` to FIFO36\_72s accordingly.
7. The user can initiate a write to memory by writing to the Address FIFO and the Write Data FIFO when FIFO full flags are deasserted. Status signal `app_af_full` is asserted when the Address FIFO is full; similarly, `app_wdf_full` is asserted when Write Data FIFO is full.
8. At power-on, both Address FIFO and Write Data FIFO full flags are deasserted.
9. The user should assert Address FIFO write enable signal `app_af_wren` along with address `app_af_addr` and command `app_af_cmd` to store the address and command into Address FIFO.
10. The user data should be synchronized to the `clk_tb` clock. Data FIFO write-enable signal `app_wdf_wren` should be asserted to store write data `app_wdf_data` and mask data `app_wdf_mask_data` into the Write Data FIFOs. Rising-edge and falling-edge data should be provided together for each write to the Data FIFO. The Virtex-5 DDR SDRAM controller design supports byte-wise masking of data only.
11. The write command should be given by keeping `app_af_cmd = 3'b000` and asserting `app_af_wren`. Address information is given on the `app_af_addr` signal. Address and command information is written into the User Address FIFO.
12. After the completion of the initialization and calibration process and when the User Address FIFO empty signal is deasserted, the controller reads the command and address FIFO and issues a write command to the DDR SDRAM.

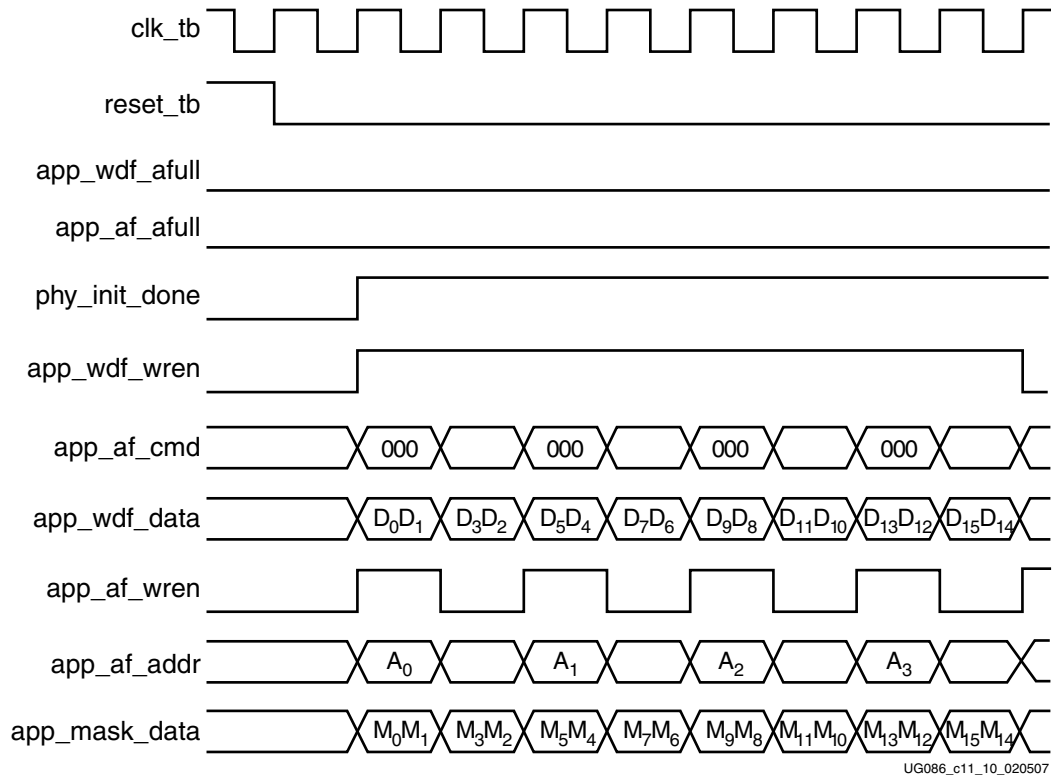
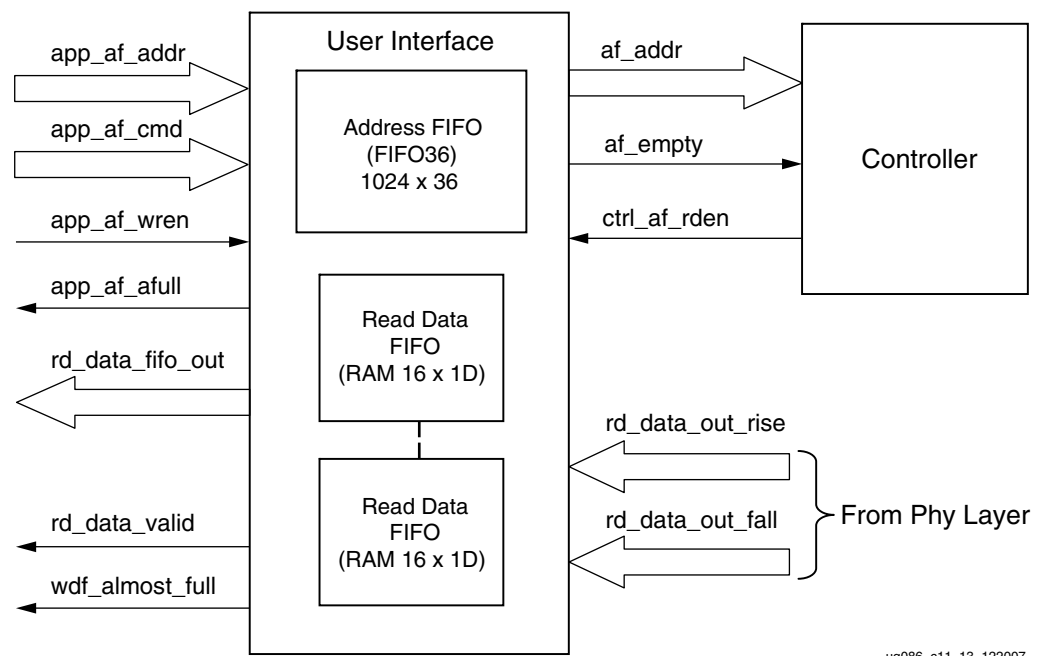


Figure 11-11: DDR SDRAM Write Burst for Four Bursts (BL = 4)

- The write timing diagram in Figure 11-11 is derived from the MIG-generated test bench for a burst length of four (BL = 4). As shown, each write to Address FIFO should have two writes to the Data FIFO. The `phy_init_done` signal indicates memory initialization and calibration completion.

## Read Interface

Figure 11-12 shows the block diagram of the read interface.



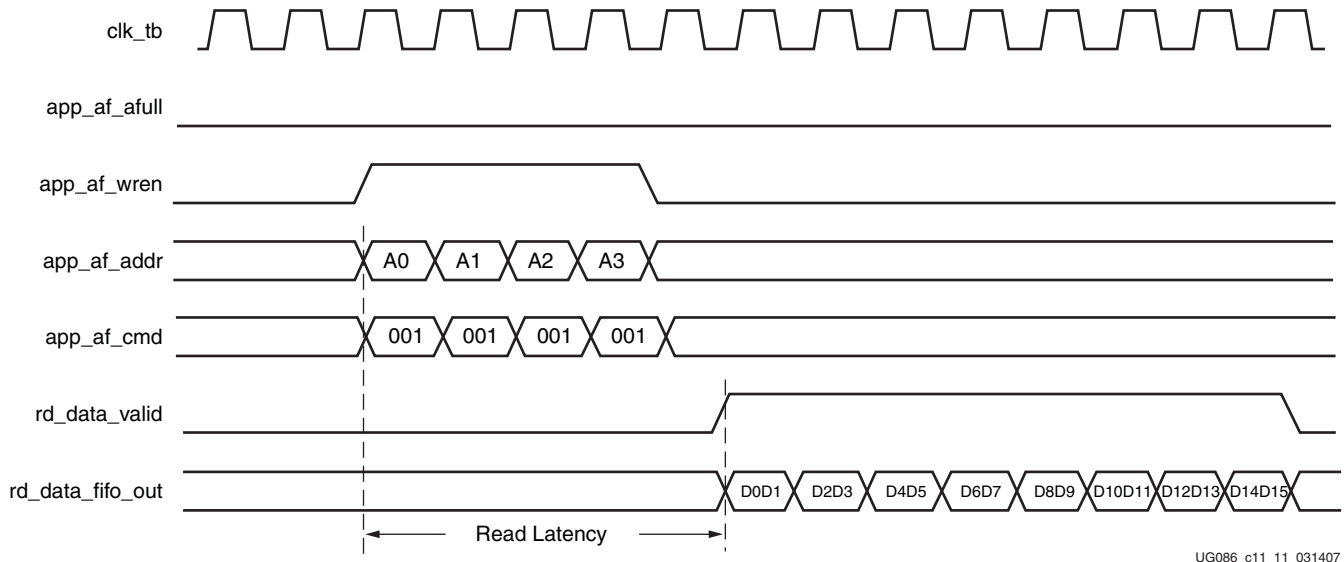
ug086\_c11\_13\_122007

Figure 11-12: User Interface Block Diagram for Read Operation

The following steps describe the architecture of the Read Data FIFO and show how to perform a read burst operation from DDR SDRAM from the user interface.

1. The read user interface consists of an Address FIFO and a Read Data FIFO. The Address FIFO is common between reads and writes. The Read Data FIFO is built out of Distributed RAMs of 16 x 1 configuration. MIG instantiates the number of RAM16Ds depending on the data width. For example, for 8-bit data width, MIG instantiates a total of 16 RAM16X1Ds, 8 for rising-edge data and 8 for falling-edge data. Similarly, for 72-bit data width, MIG instantiates a total of 144 RAM16Ds, 72 for rising-edge data and 72 for falling-edge data.
2. The user can initiate a read to memory by writing to the Address FIFO when the FIFO full flag `app_af_afull` is deasserted.
3. To write the read address and read command into the Address FIFO, the Address FIFO write enable signal `app_af_wren` should be issued, along with the memory read address `app_af_addr` and `app_af_cmd` commands (set to 001 for a read command).
4. The controller reads the Address FIFO and generates the appropriate control signals to memory. After decoding `app_af_cmd`, the controller issues a read command to the memory at the specified address.
5. Prior to the actual read and write commands, the design calibrates the latency in number of clock cycles from the time the read command is issued to the time the data is received. Using this precalibrated delay information, the controller stores the read data in Read Data FIFOs.
6. The `rd_data_valid` signal is asserted when data is available in the Read Data FIFOs.
7. When calibration is completed, the controller generates the control signals to capture the read data from the FIFO according to the CAS latency selected by the user. The

rd\_data\_valid signal is asserted when the read data is available to the user, and rd\_data\_fifo\_out is the read data from the memory to the user.



UG086\_c11\_11\_031407

Figure 11-13: DDR SDRAM Read Burst for Four Bursts (BL = 4)

- Figure 11-13 shows the user interface timing diagram for a read command, burst length of four.

Read latency is defined as the time between when the read command is written to the user interface bus until when the corresponding first piece of data is available on the user interface bus (see Figure 11-13).

When benchmarking read latencies, it is important to specify the exact conditions under which the measurement occurs.

Read latency varies based on the following parameters:

- Number of commands already in the FIFO pipeline before the read command is issued
- Whether an ACTIVATE command needs to be issued to open the new bank/row
- Whether a PRECHARGE command needs to be issued to close a previously opened bank
- Specific timing parameters for the memory, such as  $T_{RAS}$  and  $T_{RCD}$  in conjunction with the bus clock frequency
- Commands can be interrupted, and banks/rows can forcibly be closed when the periodic AUTO REFRESH command is issued
- CAS latency
- Board-level and chip-level (for both memory and FPGA) propagation delays

Table 11-7 and Table 11-8 show read latencies for the Virtex-5 FPGA DDR interface for two different conditions. Table 11-7 shows the case where a row activate is not required prior to issuing a read command on the DDR bus. This situation is possible, for example, when bank management is enabled, and the read targets an already opened bank. Table 11-8 shows the case when a read results in a bank/row conflict. In this case, a precharge of the previous row must be followed by an activation of the new row, which increases read latency. Other specific conditions are noted in the footnotes for each table.

Table 11-7: Read Latency without Precharge and Activate

Parameter	Number of Clock Cycles
User READ command to empty signal deassertion (using FIFO36)	5 clocks
Empty signal to READ command on DDR bus	4.5 clocks
READ command to read valid assertion	11.5 clocks
<b>Total</b>	<b>21 clocks</b>

**Notes:**

1. Test conditions: Clock frequency = 200 MHz, CAS latency = 3, DDR -5 speed grade device.
2. Access conditions: Read to an already open bank/row is issued to an empty control/address FIFO.
3. Some entries have fractional clock cycles because the inverted version of CLK0 is used to drive the DDR memory.
4. The Virtex-5 DDR interface uses a FIFO36 for the address/control FIFO. It is possible to shorten the READ command to empty signal deassertion latency by implementing the FIFO as a distributed RAM FIFO or removing the FIFO altogether, as the application requires.

Table 11-8: Read Latency with Precharge and Activate

Parameter	Number of Clock Cycles
User READ command to empty signal deassertion (using FIFO36)	5 clocks
Empty signal to PRECHARGE command on DDR bus	4.5 clocks
PRECHARGE to ACTIVE command to DDR memory	3 clocks
ACTIVE to READ command to DDR memory	4 clocks
READ command to read valid assertion	11.5 clocks
<b>Total</b>	<b>28 clocks</b>

**Notes:**

1. Test conditions: Clock frequency = 200 MHz, CAS latency = 3, DDR -5 speed grade device.
2. Access conditions: Read that results in a bank/row conflict is issued to an empty control/address FIFO. This requires that the previous bank/row be closed first.
3. Some entries have fractional clock cycles because the inverted version of CLK0 is used to drive the DDR memory.
4. The Virtex-5 DDR interface uses a FIFO36 for the address/control FIFO. It is possible to shorten the READ command to empty signal deassertion latency by implementing the FIFO as a distributed RAM FIFO or removing the FIFO altogether, as the application requires.

## Supported Devices

The design generated by MIG is independent of the memory package; therefore, the package part of the memory component is replaced with XX, where XX indicates a “don't care” condition. The tables below list the components (Table 11-9) and DIMMs (Table 11-10 through Table 11-12) supported by MIG for DDR SDRAM.

Table 11-9: Supported Components for DDR SDRAM (Virtex-5 FPGAs)

Components	Packages (XX)	Components	Packages (XX)
MT46V32M4XX-75	P,TG	MT46V32M4XX-5B	-
MT46V64M4XX-75	FG,P,TG	MT46V64M4XX-5B	BG,FG,P,TG
MT46V128M4XX-75	BN,FN,P,TG	MT46V128M4XX-5B	BN,FN,P,TG
MT46V256M4XX-75	P,TG	MT46V256M4XX-5B	P,TG
MT46V16M8XX-75	P,TG	MT46V16M8XX-5B	TG,P
MT46V32M8XX-75	FG,P,TG	MT46V32M8XX-5B	BG,FG,P,TG
MT46V64M8XX-75	BN,FN,P,TG	MT46V64M8XX-5B	BN,FN,P,TG
MT46V128M8XX-75	P,TG	MT46V128M8XX-5B	-
MT46V8M16XX-75	P,TG	MT46V8M16XX-5B	TG,P
MT46V16M16XX-75	BG,FG,P,TG	MT46V16M16XX-5B	BG,FG,P,TG
MT46V32M16XX-75	-	MT46V32M16XX-5B	BN,FN,P,TG
MT46V64M16XX-75	P,TG	MT46V64M16XX-5B	-

Table 11-10: Supported Unbuffered DIMMs for DDR SDRAM (Virtex-5 FPGAs)

Unbuffered DIMMs	Packages (X)	Unbuffered DIMMs	Packages (X)
MT4VDDT1664AX-40B	G,Y	MT8VDDT3264AX-40B	G,Y
MT4VDDT3264AX-40B	G,Y	MT9VDDT3272AX-40B	Y

Table 11-11: Supported Registered DIMMs for DDR SDRAM (Virtex-5 FPGAs)

Registered DIMMs	Packages (X)	Registered DIMMs	Packages (X)
MT9VDDF3272X-40B	G,Y	MT18VDDF6472X-40B	D,G,Y
MT9VDDF6472X-40B	G,Y	MT18VDDF12872X-40B	DY,G,Y

Table 11-12: Supported SODIMMs for DDR SDRAM (Virtex-5 FPGAs)

SODIMMs	Packages (X)	SODIMMs	Packages (X)
MT4VDDT3264HX-40B	G,Y	MT9VDDT3272HX-40B	-
MT4VDDT1664HX-40B	Y	MT9VDDT6472HX-40B	G,Y
MT8VDDT3264HX-40B	-	MT9VDDT12872HX-40B	-
MT8VDDT6464HX-40B	DG,DY,G,Y		



## Simulating a DDR SDRAM Design

After generating the design, MIG creates a `sim` folder in the specified path. This folder contains simulation files for a particular design. The `sim` folder contains the external testbench, memory model, `.do` file, and the executable file to simulate the generated design. The memory model files are currently generated in Verilog only. To learn more details about the files in `sim` folder and to simulate the design, refer to `simulation_help.chm` in the `sim` folder.

## Hardware Tested Configurations

The frequencies shown in [Table 11-13](#) were achieved on the Virtex-5 FPGA ML561 Memory Interfaces Development Board under nominal conditions. This frequency should not be used to determine the maximum design frequency. The maximum design frequency supported in the MIG wizard is based a combination of the TRCE results for fabric timing on multiple device/package combinations and I/O timing analysis using FPGA and memory timing parameters for a 64-bit wide interface.

*Table 11-13: Hardware Tested Configurations*

Synthesis Tools	XST and Synplicity
HDL	Verilog and VHDL
FPGA Device	XC5VLX50T-FF1136-2
Burst Lengths	2, 4, 8
CAS Latency (CL)	2, 2.5, 3
32-bit Design	Tested on 16-bit Component "MT46V32M16XX-5B"
Component, CL=2	110 MHz to 170 MHz
Component, CL=2.5	110 MHz to 210 MHz
Component, CL=3	110 MHz to 250 MHz





## *Section V: DDR2 Debug Guide*

### *Chapter 12, "Debugging MIG DDR2 Designs"*

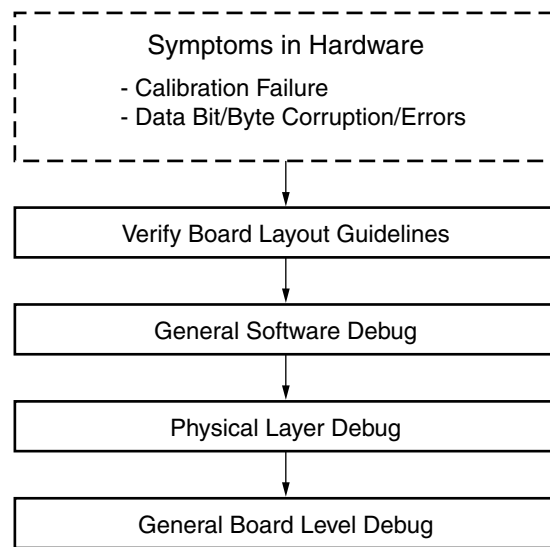


## Debugging MIG DDR2 Designs

### Introduction

Debugging problems encountered during hardware testing of MIG-generated memory interfaces can be challenging. Because of the complexity involved in designing with memory interfaces, it is necessary to have a debugging process to narrow down to the root cause of the problem to then be able to focus on the required resolution.

This chapter provides a step-by-step process for debugging designs that use MIG-generated memory interfaces. It provides details on board layout verification, design implementation verification, usage of the physical layer of MIG controllers to debug board-level issues, and general board-level debug techniques. The information in this chapter is specific to DDR2 SDRAM designs. However, the techniques covered can be applied to other memory interfaces. The overall flow for debugging problems encountered in hardware for MIG-based memory interface designs is shown in [Figure 12-1](#):



UG086\_01\_122107

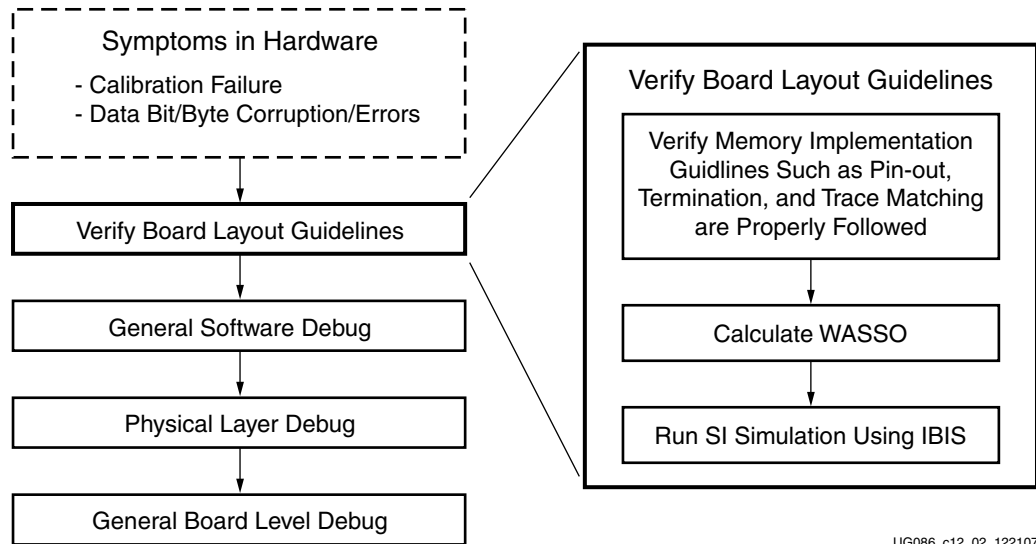
**Figure 12-1: MIG Debug Flowchart**

The following sections go into detail on each of these important debugging steps to aid in providing resolution to calibration failures and data corruptions or errors.

## Verifying Board Layout

### Introduction

There are three main steps in verifying the board layout for a memory interface, as shown in [Figure 12-2](#).



UG086\_c12\_02\_122107

Figure 12-2: Verify Board Layout Guidelines

### Memory Implementation Guidelines

See [Appendix A, “Memory Implementation Guidelines”](#) for specifications on pinout guidelines, termination, I/O standards, trace matching, and loading. The guidelines provided are specific to both memory technologies as well as MIG output designs. It is very important to verify that these guidelines have been read and considered during board-layout. Failure to follow these guidelines can result in problematic behavior in hardware, which is detailed throughout this chapter.

### Calculate WASSO

It is important to take into consideration *Weighted Average Simultaneously Switching Output* (WASSO) limits when generating a MIG pinout. The FPGA data sheets define the SSO limits for each bank. WASSO calculations take this into account along with design-specific parameters, such as board-level inductance, input logic-low threshold, input undershoot voltage, and output loading capacitance. WASSO ensures even distribution of fast/strong drivers across the package, that the number of simultaneously switching outputs does not exceed the per-bank limit and that the chip does not generate excessive ground bounce.

WASSO Calculators for Virtex™-4 devices [\[Ref 30\]](#) or Virtex-5 devices [\[Ref 31\]](#) should be used to find WASSO limits based on board-specific parameters.

These calculations should be run during both pre-board layout and post-board layout. The results found can then be entered in the Bank Selection page of the MIG GUI. (Refer to [“Bank Selection,” page 47.](#)) MIG follows these WASSO Limits when generating the pinout. Please see [Appendix C, “WASSO Limit Implementation Guidelines”](#) for further information.

## Run SI Simulation Using IBIS

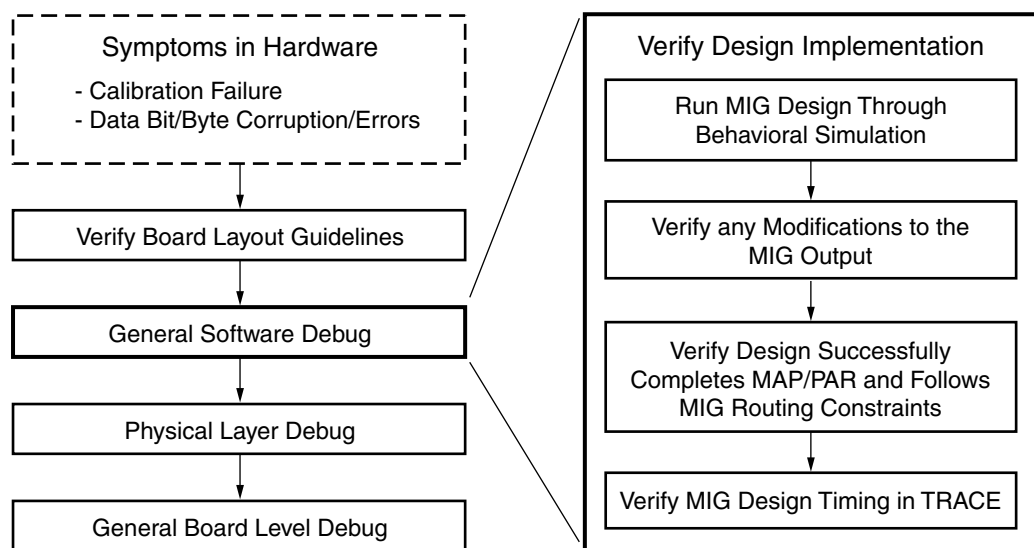
The final critical step in verifying board layout guidelines have been followed is to run signal integrity simulations using IBIS. These simulations should always be run both pre-board layout and post-board layout. The purpose of running these simulations is to confirm the signal integrity on the board.

The ML561 Hardware-Simulation Correlation chapter of the ML561 User Guide [Ref 13] can be used as a guideline. This chapter provides a detailed look at signal integrity correlation results for the ML561 board and can be used as an example for what to look at and what is good to see. It also provides steps to create a design specific IBIS model to aid in setting up the simulations.

## Verifying Design Implementation

### Introduction

There are four main steps in verifying the design implementation of a MIG output as shown in Figure 12-3:



UG086\_03\_122107

Figure 12-3: Verify Design Implementation

### Behavioral Simulation

Running behavioral simulation verifies the functionality of the design. Both the example\_design and user\_design provided with the MIG DDR2 controllers include a complete environment which allows the user to simulate the reference design and view the outputs. Scripts are provided to run behavioral simulation.

- For Virtex-4 family designs, see “[Simulating the DDR2 SDRAM Design](#)” in Chapter 3.
- For Spartan™-3/3E/3A/3AN/3A DSP family designs, see “[Tool Output](#)” in Chapter 8.
- For Virtex-5 family designs, see “[Simulating the DDR2 SDRAM Design](#)” in Chapter 9.

The Xilinx UNISIM libraries must be mapped into the simulator. If the UNISIM libraries are not set up for your environment, go to the COMPXLIB chapter of the Development Systems Reference Guide section for assistance compiling Xilinx simulation models and setting up the simulator environment. This guide can be found in the ISE™ Software Manuals.

## Verify Modifications to MIG Output

There are three modifications to the MIG output that are commonly made:

1. Changing the pinout in the provided output UCF
2. Changing design parameters defined in the output source code
3. Migrating the MIG output design into an ISE project

Each of these changes can cause problems with the implemented design that are not always visible to the user.

### Changing the Pinout Provided in the Output UCF

MIG allows users to select the desired banks rather than the exact pin locations for the memory interface. This is because specific pin assignment guidelines must be followed. See [Appendix A, “Memory Implementation Guidelines”](#) for detailed pin assignment guidelines.

Following these pin assignment guidelines when making changes to the output pinout ensures proper pin placement. However, design implementation problems can still occur. The Virtex-5 and Spartan-3 DDR2 designs require specific placement constraints outside of the pin locations. These constraint values are dependent on the pinout and so the constraints output with the MIG UCF are not correct if the pin locations are changed. The Spartan-3 and Virtex-5 architecture specific sections of this debug guide provide detailed information on these constraints and how changes cause problems.

It is always recommended to use the MIG pinout. If specific pins in the selected banks cannot be used for the memory interface, use the Reserve Pins feature of the MIG tool. (Refer to section [“Reserve Pins” in Chapter 1.](#)) If changes are made to the Virtex-4 or Virtex-5 pinout, the Verify UCF feature should always be used to test the changes against the pin assignment guidelines. (Refer to section [“Verify UCF File” in Chapter 1.](#))

### Changing Design Parameters

Often users need to change specific design parameters such as address/data widths, DDR2 memory parameters, and clock period after generating the DDR2 design. These modifications often require multiple changes to the MIG source code that are not always visible to the user. It is always recommended to re-run MIG when making any design parameter change.

For Spartan-3 and Virtex-4 family MIG designs, design parameters are defined through ``defines`. In some cases, changing one design parameter requires changing multiple ``defines` and/or portions of the source code. As an example, when changing the address or data bus widths, the source code replicates multiple instances that depend on the bus width. In this case, it is necessary to instantiate additional elements for new bits manually. Because of required modifications such as this, MIG should always be re-run when a design parameter change is required.



For Virtex-5 FPGA MIG designs, design parameters are defined using top-level parameters and generate statements. Changes to the code are no longer necessary. However, it is still recommended to re-run MIG when making design parameter modifications.

## Migrating MIG Output into ISE Project

Currently, MIG can only be generated through a stand-alone CORE Generator™ project. A batch file (`ise_flow`) is provided in the `par` directory of the output reference design to implement the MIG output design through the backend ISE tools. If the user needs to migrate the MIG reference design into an ISE project, there are two options:

1. Create a new ISE project with the MIG reference design
2. Add the MIG reference design to an existing ISE project

In order to create a new ISE project with the MIG reference design, the `create_ise` script file has been provided in the `par` directory of the output reference design. This script creates an `.ise` project file which includes the MIG source code, UCF, and appropriate synthesis and implementation project options. This script file is only available when XST is set as the synthesis option.

If the MIG reference design needs to be added to an existing ISE project, the source files must be manually added and the implementation options manually copied to the project. The `ise_flow` script file should be opened to view the necessary synthesis and implementation project options to copy into the ISE project.

When migrating a Spartan-3 Generation FPGA MIG design into an ISE project, the environment variable `XIL_ROUTE_ENABLE_DATA_CAPTURE` must be set. This variable is required to enable a needed template router and is discussed in detail in the Spartan-3 debug section below.

## Verify Successful Placement and Routing

In order to ensure proper timing of address/control or writes to the memory, specific flip-flops must be pushed into IOBs. These flip-flops include address, control and data 3-state output. To ensure proper timing, the flip-flops must be located within the IOBs. The MIG source code provides attributes to push these flip-flops into their respective IOBs. The attributes however, are specific to the synthesis tool selected in the CORE Generator project options. If XST is selected, the attributes are specific only to XST. Ensure the synthesis tool selected in the CORE Generator project options is used.

Once the design has successfully complete Place and Route, FPGA Editor can then be used to verify the correct placement of these flip-flops in the IOBs. Search for the address, control, and data IOBs in the 'List1' window under 'All Components.' Individually open each of these IOB components to verify the flip-flop is properly packed in the IOB. If the flops are not properly packed, ensure the synthesis attributes were picked up when running XST or Synplify Pro.

## Verify IDELAYCTRL Instantiation for Virtex-4 and Virtex-5 FPGA Designs

Virtex-4 and Virtex-5 FPGA designs require instantiation of the IDELAYCTRL module in the HDL in order to support the use of the IDELAY ChipSync elements for read data capture.

MIG uses the "Automatic" method for IDELAYCTRL instantiation: specifically, the MIG HDL only instantiates a single IDELAYCTRL for the entire design. No location (LOC) constraints are included in the MIG-generated UCF. This method relies on the ISE tools to

replicate and place as many IDELAYCTRLs (for example, one per clock region that use IDELAYs) as needed. In addition, the tool logically ANDs the RDY signal of each of the replicated IDELAYCTRL blocks.

The alternate method to instantiating IDELAYCTRLs is to manually instantiate as many as are needed in the design, and use LOC constraints in the UCF to fix their location. Each IDELAYCTRL must be individually location constrained. This method becomes necessary to use with a MIG design in the following cases:

- Multiple memory interfaces are used on the same device.
- The MIG design is used with other IP cores or user designs that also require the use of IDELAYCTRL and IDELAYs.
- Previous ISE tool releases 8.2.03i and 9.1i had an issue with IDELAYCTRL block replication and trimming. When using these revisions of the tool, the user must instantiate and constrain the location of each IDELAYCTRL individually.

See the Virtex-4 User Guide [Ref 7] and Virtex-5 FPGA User Guide [Ref 10] for more information on the requirements of IDELAYCTRL placement.

## Verify TRACE Timing

As a final check of proper software implementation of the MIG design, verify that all MIG provided timing constraints have completed successfully. There should be no failed timing paths in the provided MIG constraints. If the design was run in batch mode using the `ise_flow` script file, the TRACE output `<design_name>.twr` file can be opened. If the design was ran using the ISE tools, select the Analyze Post-Place and Route Static Timing option located under the Processes tab.

# Debugging the Spartan-3 FPGA Design

## Introduction

For a detailed discussion of the Spartan-3 FPGA DDR2 interface design, see application notes XAPP454 [Ref 14] and XAPP768c [Ref 23].

## Read Data Capture

Read data capture is executed using LUT based delay circuits to delay the DQS and loopback signals. The delayed DQS is then used to capture data into LUT RAM based FIFOs with the delayed loopback used as the write enable.

There are four main steps in debugging this data capture implementation as shown in [Figure 12-4](#).

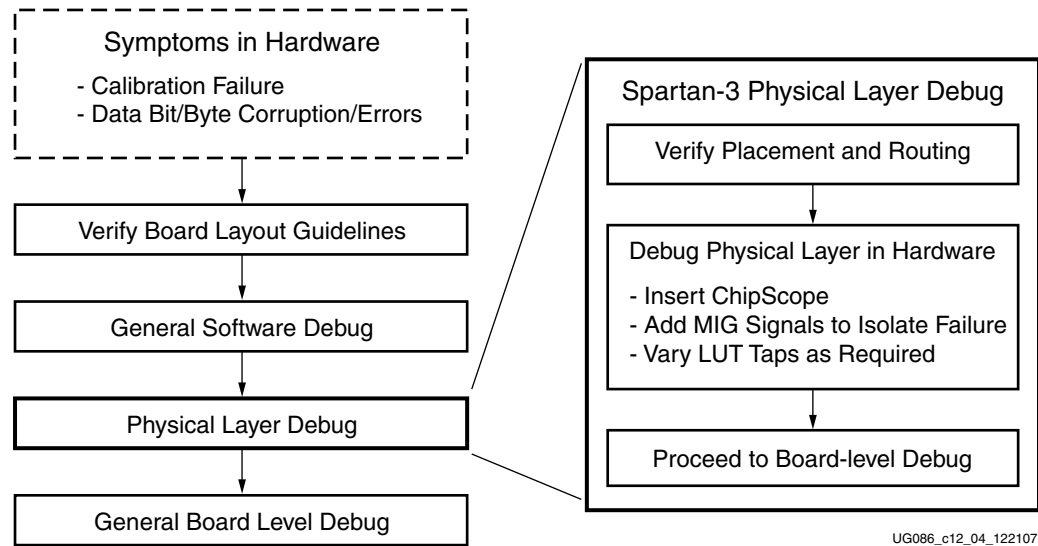


Figure 12-4: Spartan-3 FPGA Physical Layer Debug

## Verify Placement and Routing

The proper implementation of the data capture algorithm requires specific pinout and placement constraints which include PIN, LUT, BEL, and MAXDELAY, as well as usage of template routes during Place and Route.

MIG creates the appropriate UCF for the banks selected and should always be used. If changes are made to the pinout, the remaining placement constraints are no longer correct because these are based on the pin locations. Information on the specific guidelines used in creating Spartan-3 Generation FPGA UCFs are provided in [Appendix A, “Memory Implementation Guidelines.”](#) If these constraints are not followed, the data capture algorithm is not implemented properly and the results in hardware might not be as expected.

When the appropriate UCF is implemented, all related components are placed properly. This correct placement and usage of the XIL\_ROUTE\_ENABLE\_DATA\_CAPTURE environment variable forces specific routing algorithms (template routes) to be used during implementation of the PAR tools. There are two specific routing algorithms that are used:

- Routing DQ bits from a PAD to a Distributed Memory component
  - ◆ Requires the environment variable XIL\_ROUTE\_ENABLE\_DATA\_CAPTURE to be enabled during PAR implementation. This environment variable is set in the implementation script file (`ise_flow.bat`) provided in the /par MIG output directory.
- Routing delayed DQS strobe signals using Local Clocking resources
  - ◆ The PAR tools automatically treats Local Clocks as template routes and locks down the routes correctly without using the environment variable.

## DQ Routing

The template router set through the environment variable ensures the data bits are routed from a PAD to a Distributed Memory to capture the data in an Asynchronous FIFO using the Local Clock to write the data, and a Global Clock to read the data. These routes require a template to guarantee that the delay remains constant between all data bits.

Once the design is implemented, load the resultant `.ncd` and `.pcf` files into FPGA Editor to visually verify the template routes for the data bits, as follows:

1. Open the design in FPGA Editor by selecting **Start** → **Programs** → **Xilinx ISE 10.1i** → **Accessories** → **FPGA Editor**, or load through the View/Edit Routed Design (FPGA Editor) option in the Processes tab of an ISE project.
2. In some cases, turning Stub Trimming off provides a better picture of the route. To do this, select **File** → **Main Properties** and turn off Stub Trimming in the General tab. When Stub Trimming is enabled, FPGA Editor does not display the entire route. If Stub Trimming is disabled, you can see the entire length of the routing segment. Stub Trimming is enabled in [Figure 12-5](#) and [Figure 12-6](#).
3. Search within the List1 window for `*dq*` under the All Nets pull-down. Select all of the DQ data bit nets (e.g., `main_00/top0/dq(0)`) within the window and highlight these nets by clicking the **Hilite** button in the right-hand column. This allows for visual inspection of the delay routes. Zoom into the area with the highlighted nets and verify that the placement looks like [Figure 12-5](#) or [Figure 12-6](#).

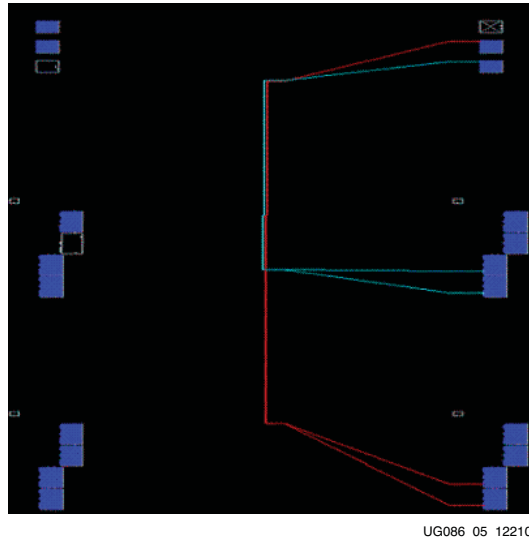


Figure 12-5: DQ Placement (Top/Bottom)

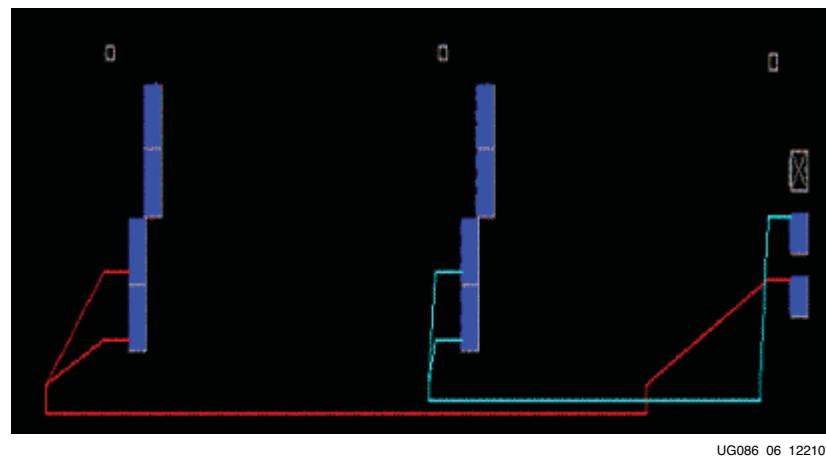


Figure 12-6: DQ Placement (Left/Right)

- Next, verify that the delays on the nets are consistent. Again, select all of the DQ data bit nets in the List1 window. This time click on the **Delay** button located in the right-hand column. This lists the worst-case delay for the DQ bits. Using this delay information, inconsistent routing can be quickly identified. There should be less than 75 ps of skew (ideally less than 50 ps) between the data nets. The delay values depend on the device speed grade and Top/Bottom versus Left/Right implementation but have been observed to range between 300–700 ps.

If preferred, export the delay information to view the report in an Excel spreadsheet. Select **File** → **Export** to export the delay information to a .csv file.

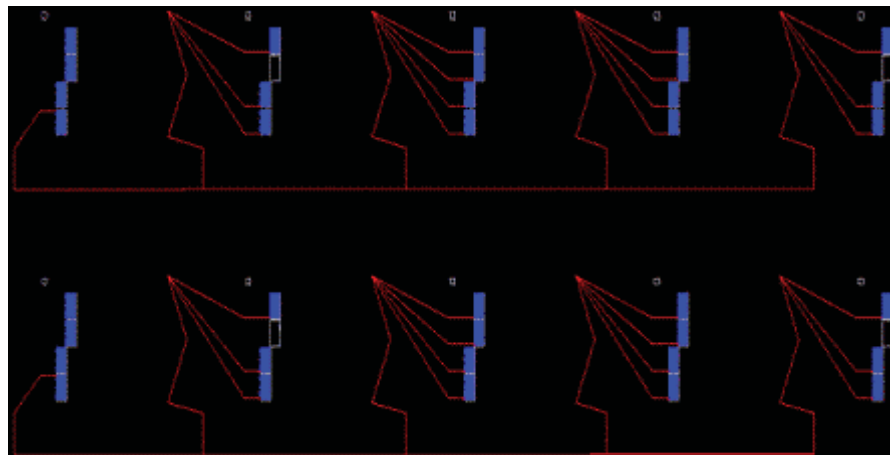
## DQS Routing

The delayed strobes (dqs\*\_delayed\_col\*) need to use the local clocking resources available in the device for the clock routing. The local routing resources used depend on the pin placement specified during generation in the MIG tool. Full hex lines that have low skew are located throughout the device. Left and right implementations use Vertical Full Hex (VFULLHEX) lines for local clock routing. Top and bottom implementations use VLONG, VFULLHEX, and HFULLHEX lines for local clock routing.

PAR routes from the Local Clock PAD to a series of LUTs to implement the scheme explained in detail in XAPP768c. From the output of the final LUT delay, the delayed strobe/Local Clock (dqs\*\_delayed\_col\*) routes to all of the FIFO bits.

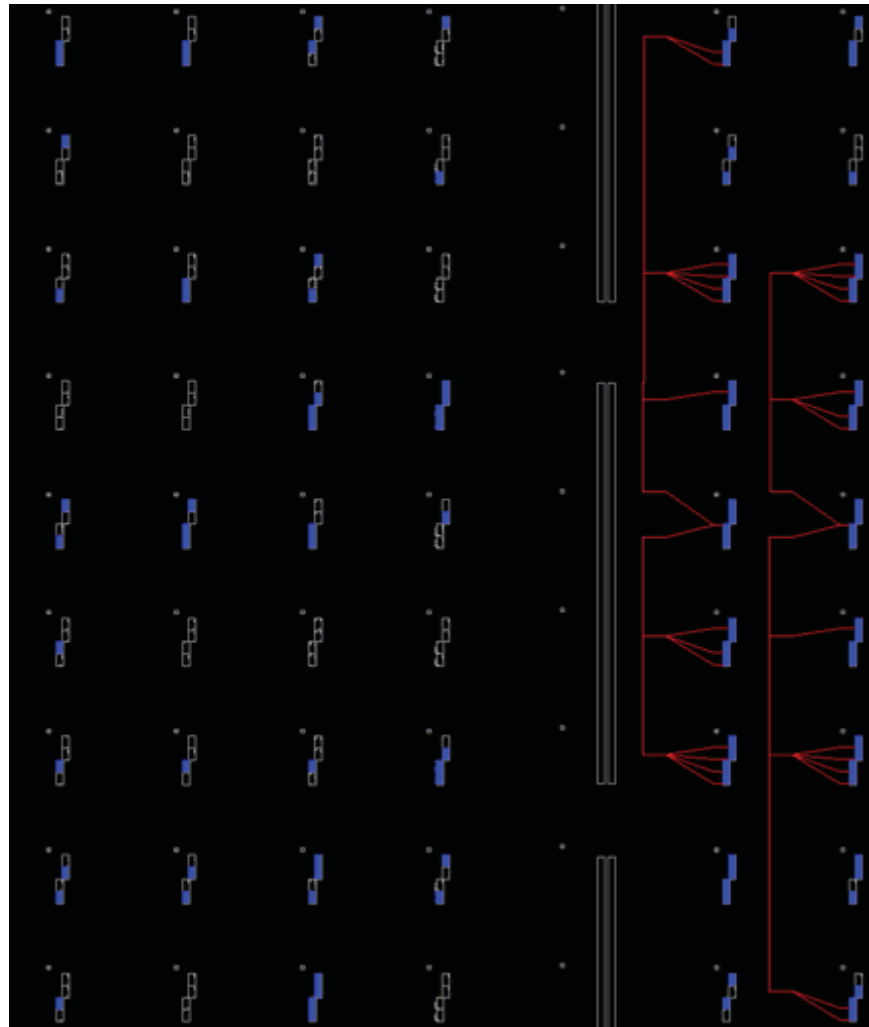
To verify the pinout and usage of the template router, the net skew and max delay on the local clock (dqs\*\_delayed\_col\*) must be within spec. To verify these values, open the PAR report (.par file) and scroll to the Clock Report section. For most Spartan-3 families, the Net Skew is less than 40 ps, and the Max Delay is approximately 550 ps. For Spartan-3A and Spartan-3A DSP devices, the Net Skew is less than 65 ps, and the Max Delay is approximately 400 ps.

The FPGA Editor can then be used to view the local clock placement. To view the template routes for the delayed strobes, search in the List1 window for \*dqs\*\_delayed\_col\* in the All Nets pull-down. Select all the nets (e.g., main\_00/top0/data\_path0/dqs0\_delayed\_col0) and select **Hilite** from the right-hand column. This command highlights the nets of interest. Then zoom into this range of highlighted signals to view the placement. If local clocking is used, one of the two structures shown in [Figure 12-7](#) and [Figure 12-8](#) is seen.



UG086\_08\_122107

Figure 12-7: Local Clock (Top/Bottom) for dqs\*\_delayed\_col\* LUT Delay Elements



UG086\_07\_122107

**Figure 12-8: Local Clock (Left/Right) for dqsj\*\_delayed\_col\*\_ LUT Delay Elements**

If the skew and delays are within spec and the layout for the Local Clock and Data bits match the above figures, the template routes for DQS have been properly implemented.

If the DQ or delayed DQS signals do not verify properly, ensure the environment variable XIL\_ROUTE\_ENABLE\_DATA\_CAPTURE was set and that the UCF follows the guidelines specified in [Appendix A](#).

## Debugging Physical Layer in Hardware

If problems are seen in hardware after verifying the correct implementation of the Spartan-3 Generation FPGA design, there are two common issues that cause problems with the data capture algorithm:

- Incorrect Loopback timing
- Incorrect delay on DQS for read capture

## Loopback Timing

The timing on the loopback signal is critical to the proper implementation of the data capture algorithm because the delayed loopback signal generates the write enable for the read data FIFOs. The causes for incorrect loopback timing are:

- Incorrect route delay on the loopback signal
  - ♦ The loopback signal must be delayed by the sum of the FPGA forward clock and the DQS trace length. This is most commonly implemented through a physical board trace.
- Changes to the MIG pinout after generation

The symptoms of incorrect loopback timing are:

- The first data in a burst is usually corrupted
- Depending on trace delays, only certain bits in the bus exhibit the problem

## Incorrect DQS Delay

The appropriate delay on the DQS strobe signals is required for proper implementation of the Spartan-3 Generation FPGA data capture algorithm. Common causes for incorrect DQS delay are:

- Mismatch in trace lengths for DQ and DQS
- Changes to the MIG pinout after generation
- Frequency changes without reimplementing of the design

If the delay on DQS is incorrect, the following symptoms can be seen in hardware:

- Incorrect data is seen intermittently
- Incorrect data is always seen

To debug either incorrect Loopback timing or incorrect DQS delay, insert a ChipScope™ Pro Virtual Input Output (VIO) core into the MIG design. The `tapfordqs1` signal located in the `cal_ctl.v/.vhd` source file should be added to the ChipScope VIO to view the number of taps in the delay paths. Use the VIO to increase or decrease the number of LUTs in the delay path while examining the resultant behavior in hardware. The number of taps increases/decreases for both the Loopback delay path and the DQS delay path. Once the appropriate number of LUT delays is found so the data corruption no longer occurs, the number of delays can then be changed within the source code. Changing the number of LUTs in the delay path can compensate for the incorrect loopback timing and incorrect DQS delay. See the ChipScope Pro User Guide [Ref 6] for detailed information on using ChipScope VIO.

## Proceed to General Board-Level Debug

If the above verification of design implementation and debug of the data capture algorithm did not resolve the issues seen in hardware, there could be a problem on the board itself. Proceed to the “[General Board-Level Debug](#)” section for further guidance.

## Debugging the Virtex-4 FPGA Direct Clocking Design

### Introduction

This section discusses internal signals to observe in order to assist in isolating problems that could occur during read data timing calibration in the Virtex-4 FPGA DDR2 Direct Clocking design. For more information on the calibration algorithm used in this design, refer to application note XAPP702. [Ref 18]

### Read Data Capture Timing Calibration

Read data timing calibration is executed over two stages:

- Stage 1: Aligning output of IDDR to internal (FPGA) clock
- Stage 2: Read Data Valid Calibration

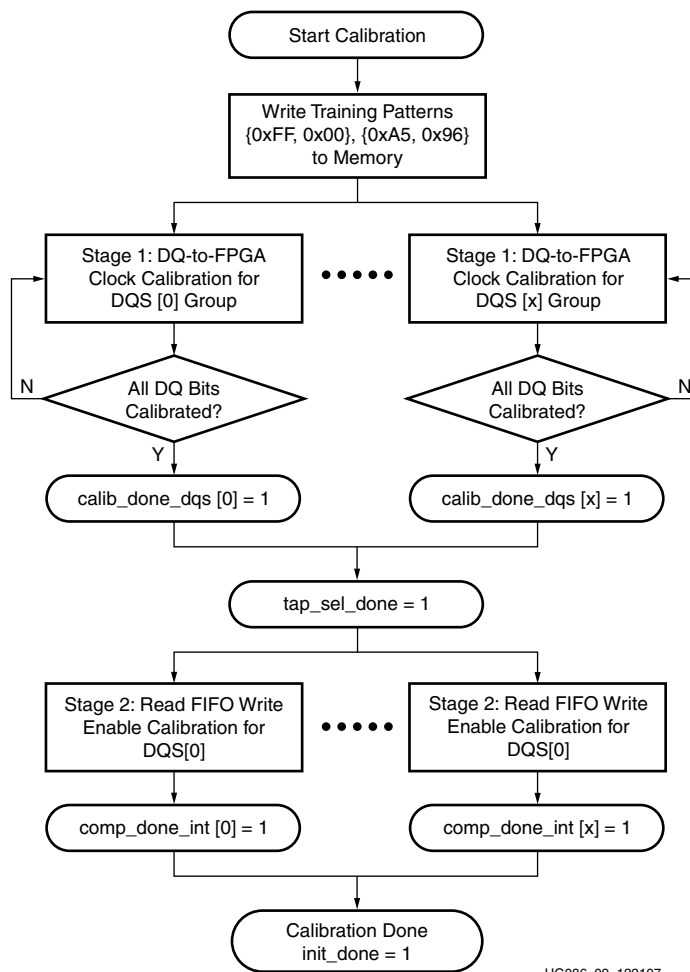
The calibration logic is parallel, in that multiple calibration units are instantiated, one for each DQS group (e.g., each calibration unit handles 4 or 8 DQ bits).

What can break during calibration?

- Stage 2 calibration checks for a specific sequence of data back from the memory
- Data bit issues (e.g., stuck-at-bit, PCB trace open/short) causes calibration to hang during Stage 2
  - ◆ Each calibration unit must be checked individually to pinpoint exactly which bit(s) failed and/or DQS groups failed

The overall calibration state machine flow diagram is shown in [Figure 12-9](#).





UG086 09 122107

Figure 12-9: Virtex-4 DDR2 Direct Clocking Overall Calibration Flowchart

## Signals of Interest

The status signals shown in Table 12-1 can be used to help determine where the failure occurs:

Table 12-1: Virtex-4 Direct Clocking Status Signals

Signal	Description
calib_done_dqs[x:0]	Asserted when individual Stage 1 calibration units have finished (one per DQS group)
tap_sel_done	Asserted when all Stage 1 calibration units have completed
comp_done_int[x:0]	Asserted when individual Stage 2 calibration units have finished (one per DQS group)
init_done	Asserted when all calibration stages successfully completed

## Proceed to General Board-Level Debug

If the above verification of design implementation and debug of the data capture algorithm does not resolve the issues seen in hardware, there could be a problem on the board itself. Proceed to the “[General Board-Level Debug](#)” section for further guidance.

# Debugging the Virtex-4 FPGA SerDes Design

## Introduction

This section discusses internal signals to observe in order to assist in isolating problems that could occur during read data timing calibration in the Virtex-4 FPGA DDR2 SerDes design. For more information on the calibration algorithm used in this design, refer to application note XAPP721. [\[Ref 22\]](#)

## Read Data Capture Timing Calibration

Read data timing calibration is executed over three stages:

- Stage 1: Aligning output of the first stage of the ISERDES to the FPGA clock
- Stage 2: Fine adjustment of Data-to-Strobe (DQ-to-DQS) capture timing into first stage of ISERDES
- Stage 3: Read data valid calibration

The calibration logic is parallel, in that multiple calibration units are instantiated, one for each DQS group (e.g., each calibration unit handles 4 or 8 DQ bits).

What can break during calibration?

- Calibration can hang at any of the stages. All stages look for a specific training pattern back from the memory. If it is not received, calibration sticks in an infinite loop reading back the data.
- Data bit issues (e.g., stuck-at-bit, PCB trace open/short) can cause calibration to hang
  - ◆ Each calibration unit must be checked individually to pinpoint exactly which bit(s) failed and/or DQS groups failed

The overall calibration state machine flow diagram is shown in [Figure 12-10](#).

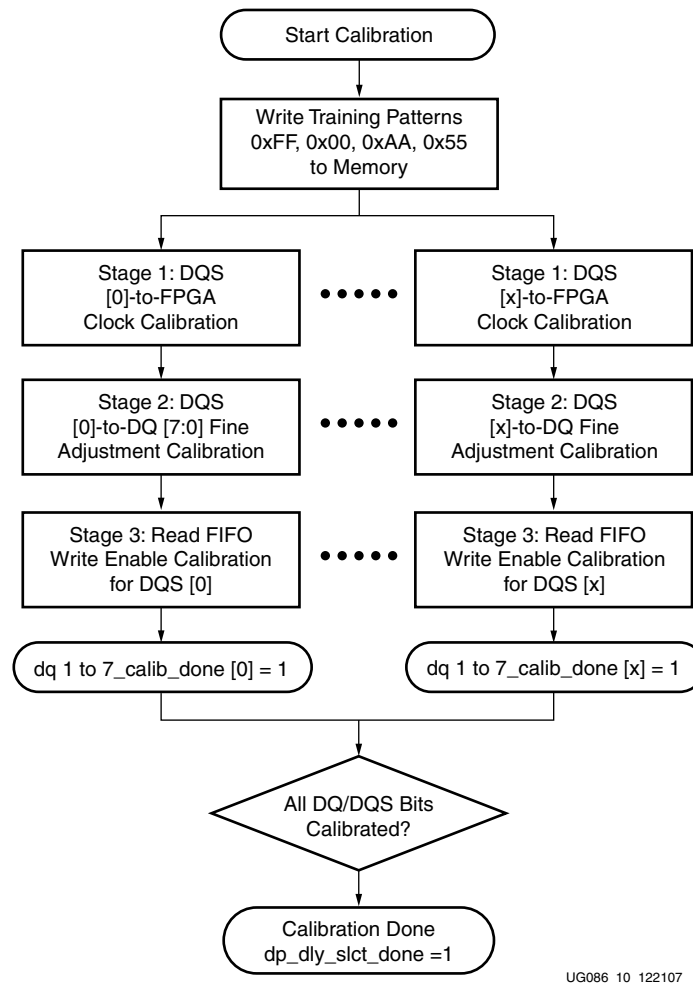


Figure 12-10: Virtex-4 DDR2 SerDes Overall Calibration Flowchart

## Signals of Interest

The status signals shown in Table 12-2 can be used to help determine where the failure occurs:

Table 12-2: Virtex-4 FPGA SerDes Status Signals

Signal	Description
calib_done_dqs[x:0]	Asserted when individual Stage 1 calibration units have finished (one per DQS group)
tap_sel_done	Asserted when all Stage 1 calibration units have completed
comp_done_int[x:0]	Asserted when individual Stage 2 calibration units have finished (one per DQS group)
init_done	Asserted when all calibration stages successfully completed

## Proceed to General Board-Level Debug

If the above verification of design implementation and debug of the data capture algorithm did not resolve the issues seen in hardware, there could be a problem on the board itself. Proceed to the [“General Board-Level Debug”](#) section for further guidance.

## Debugging the Virtex-5 FPGA Design

### Introduction

This section discusses internal signals to observe in order to assist in isolating problems that could occur during read data timing calibration in the Virtex-5 FPGA DDR2 design. Additional UCF and other parameter requirements of this design are also discussed. For more information on this design, refer to application note XAPP858 [Ref 26].

### Verify Placement and Routing

Historically, unlike the MIG Spartan-3 FPGA interface designs, most MIG Virtex-4 and Virtex-5 FPGA designs have had only pin location (LOC) and clock (PERIOD) constraints in the UCF. In some cases AREA\_GROUP constraints were included to assist with meeting timing. The MIG Virtex-5 FPGA DDR design does require location and internal timing constraints for specific read data capture related circuits.

The MIG Virtex-5 FPGA DDR2 adds a number of additional constraints to the design. This design requires properly setting both top-level parameters in HDL and constraints in the UCF that are pinout-dependent. The additional constraints in the UCF consists of location constraints for certain fabric-based resources, and internal timing (MAXDELAY) constraints. These constraints arise from changes to the read-capture path from previous revisions of MIG Virtex-5 FPGA DDR2 designs.

When creating a design in MIG, MIG automatically generates the proper HDL and UCF constraint values. However, if it becomes necessary to make changes to the MIG-generated pinout, these constraints must be manually modified. The procedure for doing so is discussed in [Appendix B, “Required UCF and HDL Modifications for Pinout Changes.”](#)

### Signals of Interest

The module PHY\_CALIB\_0.V/VHD contains the read capture timing calibration state machine.

The status signals shown in [Table 12-3](#) can be used to help determine where the failure occurs.

**Table 12-3: Virtex-5 FPGA SerDes Status Signals**

Signal	Description
phy_init_done	Asserted when both initialization of memory and read capture timing calibration has completed
calib_start[3:0]	Pulsed for one clock cycle as each calibration stage is entered
calib_done[3:0]	Driven to a static 1 as each calibration stage is finished
rd_data_rise	Captured (synchronized) rising edge read data from DDR2

Table 12-3: Virtex-5 FPGA SerDes Status Signals (Continued)

Signal	Description
rd_data_fall	Captured (synchronized) falling edge read data from DDR2
cal1_dq_count	Binary value indicating the current DQ bit being calibrated during Stage 1
cal2_dq_count	Binary value indicate the current DQS group being calibrated during Stage 2

### Physical Layer Debug Port

The Virtex-5 DDR2 design HDL contains an optional port to allow the user to observe and control the IDELAY tap values for the DQ, DQS, and DQS Gate signals after read capture timing calibration. This is described in [Appendix D](#).

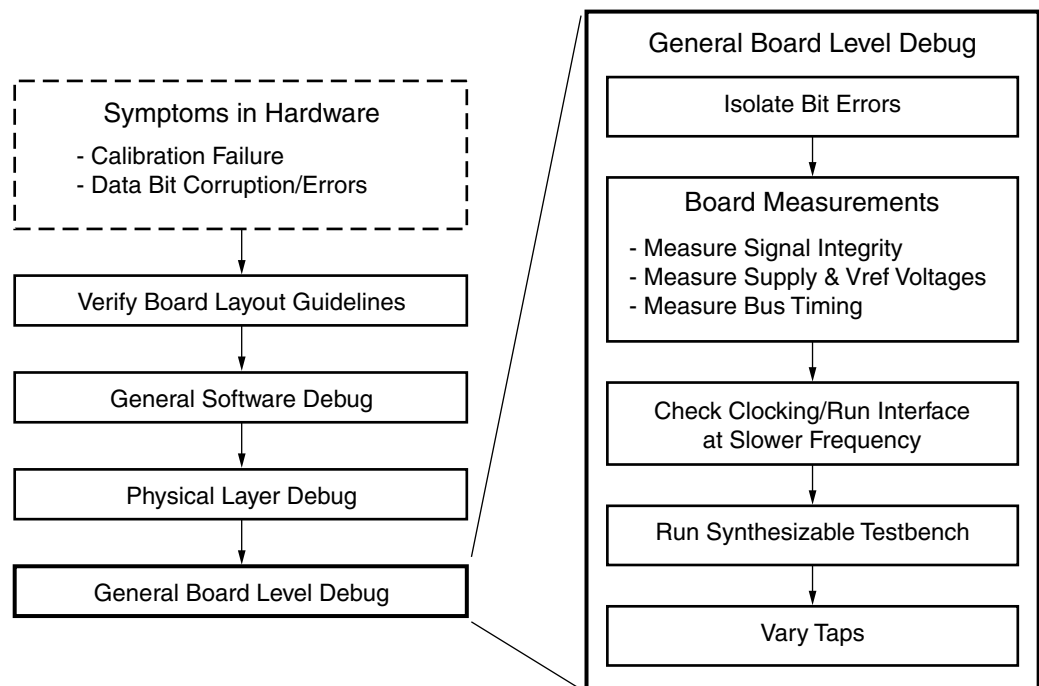
### Proceed to General Board-Level Debug

If the above verification of design implementation and debug of the data capture algorithm does not resolve the issues seen in hardware, there could be a problem on the board itself. Proceed to the “[General Board-Level Debug](#)” section for further guidance.

## General Board-Level Debug

### Overall Flow

The flowchart shown in [Figure 12-11](#) documents recommended steps to try during board-level debug.



UG086\_11\_122107

Figure 12-11: General Board-Level Debug Flowchart

## Isolating Bit Errors

In this step, the user stays in the HDL domain and tries to isolate when/where the bit errors are occurring.

When are the error(s) occurring?

- Data belonging to certain DQS groups?
- On accesses to certain addresses, banks, or ranks of memory?
  - ◆ For example, on designs that can support multiple varieties of DIMM modules, make sure to support all possible address and bank bit combinations
- Only occur for certain data patterns or sequences?
  - ◆ This can indicate a shorted or open connection on the PCB
  - ◆ This can also indicate an SSO or cross-talk issue
- Does the design use multiple DIMM sockets?
  - ◆ All MIG designs that support multiple DIMM sockets (“deep” configurations) calibrate only on the first DIMM socket, and the maximum frequency is reduced from the maximum achievable if only one rank of memory is used. This was done to account for both the additional loading and the fact that there are no inherent, process-related timing differences between the DIMM sockets. Factors that cause the timing to be different between the DIMMs—for example, PCB trace routing differences between the FPGA and each of the DIMMs—can result in read failures on all but the very first DIMM.

It might also be necessary to determine whether the data corruption is due to writes or reads. This can be difficult to determine because, if the writes are the issue, readback of the data appears corrupted as well. In addition, issues with control/address timing affect both writes and reads. Some experiments that can be tried to isolate the issue:

- If the errors are intermittent, have the controller issue a small initial number of writes, followed by continuous reads from those locations. Do the reads intermittently yield bad data? If so, this might point to a read problem.
- Check/vary the control and address timing:
  - ◆ For a heavily loaded control/address bus (as is the case for an unregistered or SO-DIMM), it might be necessary to use 2T timing to allow for more setup and hold time for the control/address signals.
  - ◆ Note that the chip select (CS\_N) signal to the memory remains a 1T signal, even though it can also have a heavy load. In this case, it might be necessary to advance the assertion of CS\_N by a quarter of a clock cycle. This requires changing the code for the CS\_N output flop to use CLK90 instead of CLK0.
- Check/Vary only write timing:
  - ◆ If on-die termination is used, check that the correct value is enabled in the DDR2 device and that the timing on the ODT signal relative to the write burst is correct.
  - ◆ For Virtex-5 designs, it is possible to use ODELAY to vary the phase of DQ relative to DQS. In addition, a PLL (rather than a DCM) can be used to generate CLK0 and CLK90 used for the write output timing. The phase outputs of a PLL can be fine-tuned, and in this way the phase between DQ and DQS can be varied.
- Vary only read timing:
  - ◆ Vary the LUT or IDELAY taps after calibration for the bits that are returning bad data. This affects only the read capture timing.

- ◆ For Virtex-4 and Virtex-5 FPGA designs, check the IDELAY values after calibration. (For the Virtex-5 DDR2 design, the PHY layer debug port can be used.) Look for variations between IDELAY values. IDELAY values should be very similar for DQs in the same DQS group.

## Board Measurements

Refer to the HW-Simulation Correlation Section in the ML561 User Guide [Ref 13] as a guide for expected bus signal integrity.

## Supply Voltage Measurements

Check the reference voltage levels:

- For I/O:
  - ◆ 1.8V: VCCO, DDR2 VDDQ
  - ◆ 0.9V: VREF
  - ◆ 0.9V: VTT Termination
- Internal:
  - ◆ 1.8V: DDR2 VDD, DDR2 VDDL
  - ◆ 2.5V: FPGA VCCAUX
  - ◆ 1.0V or 1.2V: VCCINT

Make sure to check these levels when the bus is active. It is possible these levels are correct when the bus is idle but droop when the bus is active.

## Clocking

If the memory interface is having issues running at the target speed, try running the interface at a lower speed.

- Unfortunately, not all designs can accommodate this, as it is dependent on the clock generation scheme used.
- Running at a lower speed increases marginal setup time and/or hold time due to PCB trace mismatch, poor SI, and excessive loading.

If excessive input/system clock jitter might be an issue, the onboard PLL can be used in Virtex-5 FPGA designs to filter input clock jitter.

## Synthesizable Testbench

MIG provides a “synthesizable testbench” containing a simple state machine. The state machine takes the place of the user-specific backend logic and issues a simple repeating write-read memory test. This can be used as an alternative to the user's backend logic to provide a test of the memory interface during initial hardware bring-up. The advantage of using the synthesizable testbench is that it rules out any issues with the user's backend logic interfacing with the MIG User Interface block.

The testbench has limitations. It only checks a limited number of memory locations, and the data pattern is a repeating pattern. The user can change the testbench code to expand its capabilities.

## Varying Read Capture Timing

For Virtex-4 and Virtex-5 FPGA designs, the IDELAY values for DQ and DQS can be varied post-calibration. The user can determine the extent of the read valid window in this way. The customer can also use this feature for margin testing. This feature is supported in HDL in the Virtex-5 FPGA DDR2 design. In other designs, the user must modify the HDL to add the hooks to vary the IDELAY taps.

For Spartan-3 FPGA designs, LUTs are used to delay the DQS and the loopback signal. The user can modify the code to use a different number of LUT delays to change the DQ-DQS timing, but there is a much larger granularity (approximately 250–600 ps) than with the IDELAY element of Virtex-4 and Virtex-5 FPGAs.





## *Section VI: Appendices*

*Appendix A, "Memory Implementation Guidelines"*

*Appendix B, "Required UCF and HDL Modifications for Pinout Changes"*

*Appendix C, "WASSO Limit Implementation Guidelines"*

*Appendix D, "Debug Port"*



## 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:

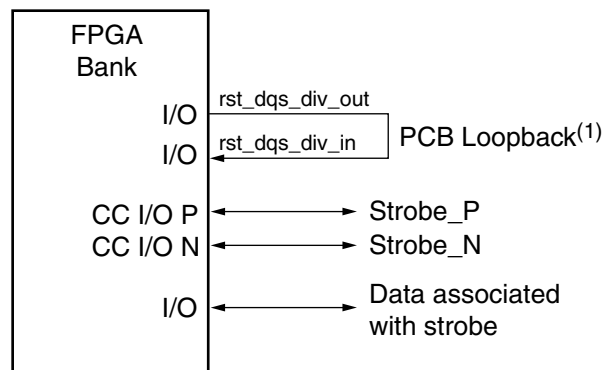
- ◆ DDR/DDR2 SDRAM
- ◆ QDRII SRAM
- ◆ RLDRAM II

UG079 [Ref 9] and UG199 [Ref 13] provide more detailed analysis. UG072 [Ref 8] and UG203 [Ref 11] provide 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.



**Notes:**

1. Only Spartan FPGA designs require the loopback signal.

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

## Timing Analysis

MIG generates timing analysis spreadsheets for all designs of Virtex-5, Virtex-4, and Spartan families under the `docs` folder. Each design has different timing analysis spreadsheets for `read_data_timing`, `write_data_timing`, and `addr_cntrl_timing`.

Evaluation of the PERIOD constraint by the static timing analyzer is not sufficient to determine if the memory interface is functional at a particular frequency. The PERIOD constraint covers the internal timing between synchronous elements. These spreadsheets cover the concept of timing budgets for the interface between the FPGA and memory device.

The spreadsheets provide information about the data valid window and the margins available at the selected frequency. They also provide information about different uncertainty parameters that are to be considered for timing analysis.

## Pin Assignments

MIG generates pin assignments for a memory interface based on certain rules depending on the design technique, but does not provide the best possible pin assignment for every board implementation. During layout it might be necessary to swap pin locations depending on the number of layers available and the interface topology. The best way to change the pin assignment is to first apply changes on a byte basis then swap bits within a byte. Calculate the PCB loopback length, if required, after pin swapping and trace matching. The following rules of thumb are provided to help designers determine how pins can be swapped.

Any changes to the pin assignments require modifications to the UCF provided by MIG and might require changes to the source code depending on the changes made.

For all MIG Virtex™ and Spartan™ FPGA designs, the address and control pins can be swapped with each other as needed to avoid crossing of the nets on the printed circuit board.

## Spartan-3/3E/3A/3A DSP FPGA Memory Implementation Guidelines for DDR/DDR2 SDRAM Interfaces

This section outlines general pin assignment guidelines for DDR/DDR2 SDRAM implementation. However, additional guidelines should be followed when targeting Spartan-3/-3E/-3A/-3A DSP devices.

MIG generates a UCF that follows the guidelines listed below. Xilinx recommends using the pinout created by MIG. Follow the guidelines below if the MIG pinout is modified.

The IOBs for DQ bits must be placed five tiles above or six tiles below the IOB tile for the associated DQS bit. This is necessary because the MIG design uses low-skew routing resources to route DQS to the data capture FIFOs corresponding to that DQS. See XAPP768c [Ref 23] for more information on the Spartan-3 FPGA data capture technique. This application note can be downloaded from the web page entitled *Memory Interfaces: Resources for Registered Users* located at:

<http://www.xilinx.com/support/software/memory/protected/index.htm>

*Example:*

If DQS is placed in either W3 or W4 (these two IOBs share a tile) in an XC3S1500-FG676, the following +5 tiles can be used for DQ placement:

W1/W2  
U7/V7  
V4/V5  
V2/V3  
U5/U6

The following –6 tiles can be used for DQ placement:

W5/V6  
W6/W7  
Y1/Y2  
AA1/AA2  
Y4/Y5  
AA3/AA4

**Caution!** Unbonded tiles (even though they cannot be used) count toward this +5/–6 guideline. Consequently, it is possible that a pinout that meets the above requirements for a specific bus width cannot be supported on a larger device in the same package (even though the package is “pinout compatible”). MIG can be used to generate a pinout compatible design for multiple devices in the same package.

To verify the pin placement of the DQ and DQS bits, you can check the net skew and delay values in FPGA Editor and the “Clock Report” section of the design’s PAR report (.par file). See the Debug section of the ug086 for steps to verify the DQ and DQS skew and delay values.

- The `rst_dqs_div_in` and `rst_dqs_div_out` IOBs must be placed in the center of the DQ bits. As an example, if the data bus is 64 bits wide, `rst_dqs_div_in` and `rst_dqs_div_out` should be placed between DQ[31] and DQ[32]. If this is not done, the data capture might not be reliable. This is necessary because the MIG design uses the RST\_DQS\_DIV feedback loop to generate a write enable to all the data capture FIFOs. See XAPP768c [Ref 23] for further information on the Spartan-3 FPGA design.
- Spartan-3 FPGA architectures only have two FIFOs per CLB. Because each bit of data requires two FIFOs (one for rising edge data and one for falling edge data), the MIG designs use two columns of CLBs. One CLB column is dedicated for the odd numbered bits and one is dedicated for the even numbered bits. Due to Spartan-3 routing restrictions, pad0 (top) must be assigned to the first column CLBs and pad 1 (bottom) assigned to second column of CLBs. With this routing implementation, the DQ lines from both pads has the same route delay.
- The CK/CK\_N, address, RAS\_N, CAS\_N, WE\_N, CS\_N, and ODT must be placed together in banks that are on the same side of the device. This helps to avoid clock skew on these signals that are registered on the rising edge of CK.

For memory interfaces that do not provide a signal to indicate when the read data is valid, a data-valid signal must be provided on the PCB. This loopback is used as a write-enable signal for the Read Data FIFOs. A strobe is used to latch the data. Two pins are needed per design: one to output the signal and one to input the return signal. The length of the loopback is defined as:

$$\text{PCB loopback} = \text{CLK delay to memory} + \text{strobe delay}$$

Spartan-3/3E/3A/3A DSP FPGA designs have specific pin placement rules that are followed by MIG to generate the pin assignments. A byte can be swapped with another

byte as long as all the necessary signals associated with that byte are changed (strobe, data, and data mask). Within a byte, only even-numbered bits can be swapped with other even-numbered bits (with the same rule applying for odd-numbered bits) because two copies of the DQS strobe are internally generated: one copy for even-numbered bits and one for odd-numbered bits. Each copy is delayed a specific amount relative to the placement of the even (or odd) Read Data FIFOs. As an example, in a byte bits 0 and 2 can be swapped but bits 0 and 1 cannot be swapped. The UCF provided by MIG contains LOC constraints that must be changed to match the swapped pin assignments.

## XIL\_ROUTE\_ENABLE\_DATA\_CAPTURE

The local clocking scheme used to capture data in all MIG Spartan-3 Generation FPGA memory designs requires place and route (PAR) template routes to properly place the delayed strobe and data bits. Template routing is required to properly route the delayed strobe (`dqs*_delayed_col*`), as well as the data (`dq` bits) in MIG Spartan-3 Generation FPGA DDR and DDR2 SDRAM designs. For the data bits to be routed properly, the environment variable `XIL_ROUTE_ENABLE_DATA_CAPTURE` must be enabled when PAR is run. This environment variable is set in the implementation script file `ise_flow.bat` provided in the `/par` MIG output directory. The user must set this environment variable when running the design using the GUI mode from `create_ise.bat`.

## Virtex-4 FPGA Direct Clocking Pins

1. For flexibility of design techniques, it is recommended that all strobe signals be placed on clock-capable inputs (such as DQS, CQ, and 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 their associated strobe. Data is captured with an internal FPGA clock. Data is delayed through the IDELAY element to make it center-aligned with the FPGA clock. The strobe is used to find the data delay with respect to the FPGA clock.
3. Address and control signals are to be placed together in the same bank (see “[Memory-Specific Guidelines](#),” page 405 for exceptions) or placed in banks near each other to minimize the route delays for these signals inside the FPGA.
4. DDRII SRAM ONLY: 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. This loopback is used as a write-enable signal for the Read Data FIFOs. A strobe is used to latch the data. Two pins are needed per bank: one to output the signal and one to input the return signal. The length of the loopback is:

$$\text{PCB loopback} = \text{CLK delay to memory} + \text{strobe delay}$$

Virtex-4 FPGA Direct clocking designs that place the strobe on clock-capable I/O should follow the pin-swapping recommendations for the Virtex-4 SerDes and Virtex-5 FPGA designs. If the strobe is not placed on clock-capable I/O, an entire DQS group (containing data, strobe, and data mask) can be swapped with any other DQS group in same bank. The initial pinout that MIG selects also affects the amount of calibration logic MIG generates. MIG generates one calibration unit for each bank that contains data bits. Therefore, a DQS group cannot be swapped with other byte groups on different banks without appropriate modification to the source code. Within a DQS group, data bits can be swapped with other data bits, and the data signals should be placed on pins near the associated DQS strobe.

## Virtex-4 FPGA SerDes Clocking and Virtex-5 FPGA Pins

1. All strobe signals must be placed on clock-capable inputs (such as DQS, CQ, and 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 their associated strobe. Data is captured in the ISERDES block using the strobe signal. The strobe is passed through the BUFIO to delay it with respect to the data input.
3. Address and control signals are to be placed together in the same bank (see “[Memory-Specific Guidelines](#),” page 405 for exceptions) or placed in banks near each other to minimize the route delays for these signals inside the FPGA.

Virtex-4 SerDes clocking and Virtex-5 FPGA designs must place the strobe on clock-capable I/O with the data for the said strobe placed in the same bank. A byte can be swapped with another byte as long as all the necessary signals associated with that byte (strobe, data, and data mask) are located in the same bank. Within a bank, strobes can be swapped with other strobes while the rest of the pins in a bank can be swapped as needed.

The Virtex-5 FPGA DDR2 design uses a combination of the IOB flop (IDDR) and fabric-based flops for read data capture. This requires the use of pinout-dependent directed-routing and location constraints. If pinouts are changed manually, the UCF must be modified. Refer to [Appendix B, “Required UCF and HDL Modifications for Pinout Changes”](#) for details.

## 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  $50\Omega$  to  $V_{TT}$  at the load (see [Figure A-2](#)). A split  $100\Omega$  termination to  $V_{CCO}$  and  $100\Omega$  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).

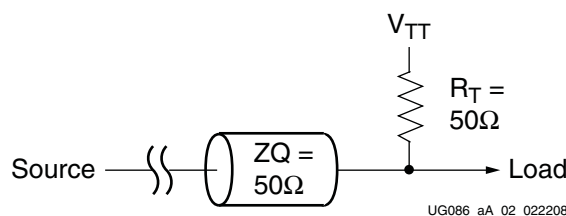


Figure A-2:  $50\Omega$  Termination to  $V_{TT}$

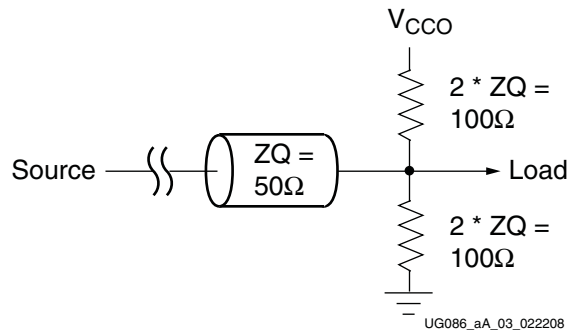


Figure A-3: 100Ω Split Termination to V<sub>CC0</sub> and GND

- 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).

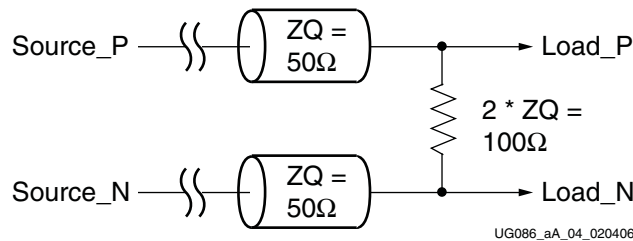


Figure A-4: 100Ω Differential Termination

- 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.
- DCI can be used at the FPGA as long as the DCI rules are followed (such as VRN/VRP).

## I/O Standards

These rules apply to the I/O standard selection for DDR SDRAMs:

- MIG-generated designs use the SSTL2 CLASS II I/O standard by default for memory address and control signals, and use the SSTL2 CLASS II I/O standard for memory data, data-mask, and data-strobe signals. When DCI is selected in MIG, DCI for SSTL2 CLASS I can be applied only to memory interface signals that are inputs to the FPGA.
- The user can select CLASS II or CLASS I I/O standards from MIG. When SSTL2 CLASS II is selected in MIG, it is applied to all the memory interface signals.
- When DCI is selected in MIG, the DCI I/O standard is applied to all the memory interface signals.

These rules apply to the I/O standard selection for DDR2 SDRAMs:

- MIG-generated designs use the SSTL18 CLASS II I/O standard by default for all memory interface signals. When DCI is selected in MIG, DCI for SSTL18 CLASS II is applied on input, output, and in-out memory interface signals.
- The user can select between CLASS II or CLASS I I/O standards from MIG. When SSTL18 CLASS I is selected in MIG, the I/O standard for bidirectional signals remains SSTL18 CLASS II.



- When DCI is selected in MIG for SSTL18 CLASS I, the DCI I/O standard is applied only to memory interface signals that are inputs or in-outs to the FPGA.

## Trace Lengths

Trace length matching must also include the package delay information. The PARTGen utility [Ref 29] generates a `.pkg` file that contains the package trace length in microns for every pin of the device under consideration.

For example, to obtain the package delay information for a Virtex-5 LX50T-FF1136 device used on an ML561 board, issue the following command within a DOS command shell:

```
partgen -v xc5v1x50tff1136
```

This generates an `xc5v1x50tff1136.pkg` file in the current directory with package trace length information for each pin (unit: micron or  $\mu\text{m}$ ). Use the typical 6.5 fs per micron (6.5 ps per millimeter) conversion formula to obtain the corresponding electrical propagation delay. While applying specific trace-matching guidelines for each of the memory interfaces as described below, consider this additional package delay term for the overall electrical propagation delay.

## 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. Be sure to include the package delay when determining the effective trace length. These internal delays can be found through the PACE tool.

## 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 or placed in banks near each other.  
If all control signals cannot fit in one bank, CK, ODT, and CKE should be selected first for placement 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 FPGA MIG 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).

The loading of address (A, BA), command (RAS\_N, CAS\_N, WE\_N), and control (CS\_N, ODT) signals depends on various factors, such as speed requirements, termination topology, use of unbuffered DIMMs, and multiple rank DIMMs.

The address and command signals should be implemented with 2T clocking, i.e., asserted for two cycles, so these signals can handle higher loading without impacting the timing budget. Virtex-4 FPGA SerDes designs and Virtex-5 FPGA DDR2 designs are implemented with 2T clocking of address and command signals.

The control signals (CS\_N and ODT) have 1T clocking, and so their replication is recommended when the loading is higher. If the application is pin-limited to implement lighter loading on critical clock signals going to memory, it might be necessary to use an external PLL to generate multiple copies of the clock signals.

For descriptions of 1T and 2T clocking, refer to Micron technical note TN-47-01 [Ref 32].

## 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.

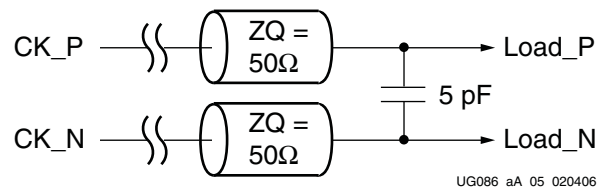


Figure A-5: 5 pF Differential Termination on Clocks

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.
4. The Virtex-5 DDR2 interface requires that if parallel termination is used at the memory end, it must be ODT rather than external termination resistor(s). This is a requirement of the read capture scheme used.

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 at 333 MHz:

1. ± 25 ps maximum electrical delay between any DQ and its associated DQS/DQS#
2. ± 50 ps maximum electrical delay between any address and control signals and the corresponding CK/CK#
3. ± 100 ps maximum electrical delay between any DQS/DQS# and CK/CK#

## 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, if the Use CC option is selected; otherwise any I/O pin is used. P is connected to the P side and N is connected 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.

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. 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 k $\Omega$  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.

### I/O Standards

These rules apply to the I/O Standard selection for QDRII SRAM.

- MIG-generated designs use the HSTL CLASS I I/O standard by default for all memory interface signals.
- When DCI is selected in MIG, the DCI standard for HSTL CLASS I is applied only to memory interface signals that are inputs to FPGA.

### Trace Lengths

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

1.  $\pm 25$  ps maximum electrical delay between data and its associated CQ.
2.  $\pm 50$  ps maximum electrical delay between address and control signals.
3.  $\pm 100$  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 if the Use CC option is selected in the tool; otherwise normal I/O pins are used. P is connected to the P side and N is connected to the N side of the pair.
2. The DQ bits of a byte are placed in the same bank as the associated QK.  
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.

### I/O Standards

These rules apply to the I/O Standard selection for RLDRAM II:

- MIG-generated designs use the HSTL CLASS II I/O standard by default for all memory interface signals. When DCI is selected in MIG, DCI for HSTL CLASS II is applied on input, output, and in-out memory interface signals.
- The user can change the I/O standard to HSTL CLASS I. When DCI is selected in MIG, DCI for HSTL CLASS I is applied only to the memory interface signals that are inputs to the FPGA.
- To have HSTL CLASS I on the required pins, the user must manually edit the UCF constraint file for the corresponding design generated.

### Trace Lengths

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

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

## Required UCF and HDL Modifications for Pinout Changes

---

### Introduction

The Virtex™-5 FPGA DDR2 design generated by MIG 2.0 (or later) requires a large number of UCF constraints whose values are dependent on the specific pinout of the DQ and DQS bits. In addition, there are two top-level HDL parameters whose values are also pinout dependent. These UCF constraints and HDL parameters are not present for designs generated with MIG 1.73 or earlier.

MIG generates a user constraints file (UCF) and HDL code with the correct constraints and top-level parameters based on the pinout, and in this case, the user does not need to know the specific rules and procedures for generating these constraints. However, it is necessary for the user to *manually* generate these constraints if any of these three conditions exist:

- The user has a pinout based on a DDR2 design generated using an older version of MIG (for example, MIG 1.7), and it is desired to up-rev the design to the MIG 2.0 (or later) version of the DDR2 interface.
  - ◆ The older MIG-generated pinout is compatible with the MIG 2.0 (or later) version of the design, but, the user *must* generate the additional constraints required by MIG 2.0.
  - ◆ MIG 2.0 (or later) has a slightly different algorithm for selecting the DQ and DM (data mask) sites, choosing different pins for the DM and some of the corresponding DQ pins. Therefore, running MIG 2.0 or later with the same bank pinout selection setting used for the original pre-MIG 2.0 design could result in a UCF and HDL top-level file with some incorrect constraints and parameters, such as DQ and DM being allocated to different pins. However, the user can use the MIG 2.0 or later UCF as a baseline for modifications.
- The user has generated a design using MIG 2.0 (or later), but needs to make modifications to the pinout (for example, swapping DQ bit locations).
- The user has generated a pinout independent of MIG.

**Caution!** *This is not recommended.* MIG should be used to generate the pinout. If an independently generated pinout must be used, a UCF should be generated using MIG and used as a baseline for constraint modifications.

These additional constraints are required because of changes to the read data capture architecture used in this design: specifically, a combination of the IOB-based IDDR flop and flops located in the FPGA fabric is used instead of the ISERDES to capture read data. A circuit to gate a glitch on the DQS strobe at the end of read bursts added with the MIG 2.0 or later design also requires additional constraints.

## UCF / HDL Constraint Generation Procedure

The following is a step-by-step procedure required to generate the additional UCF constraints and HDL parameters required for the Virtex-5 MIG 2.0 or later versions of MIG design. The specific reasons why these changes need to be made are discussed later in this section.

1. Use MIG 2.0 or later versions of MIG to generate a UCF using the same parameters as were used to generate the original pre-MIG 2.0 DDR2 design; in particular, the clock frequency and data width must be the same. Substitute the location (LOC) constraints for the existing user pinout into this UCF. This file is used as a baseline for further modifications. (It is also possible to start with a pre-MIG 2.0 design and add the constraints.)
2. Use the Xilinx ISE utility PARTGEN to generate a package file for the specific target device. This is used to determine correct (pinout-specific) values for many of the UCF constraints:

```
partgen -v <part number> (e.g. partgen -v xc5v1x330tff1738)
```

3. UCF constraints must be modified to match the user-specific pinout:
  - a. Verify (no modification required for this step) in the UCF the presence of FROM-TO constraints that define multi-cycle paths. These are generated by 2.0 or later versions of MIG, and their values are not pinout dependent. These constraints help meet internal (fabric) timing at the higher frequencies that MIG supports. At lower frequencies of operation, these multi-cycle path constraints might not be required to meet internal timing and can be removed. One of these multi-cycle path constraints is shown below:

```
NET "clk0"      TNM = FFS "TNM_CLK0";
NET "clk90"     TNM = FFS "TNM_CLK90";
# MUX Select for either rising/falling CLK0 for 2nd stage
  read capture
INST "*/u_phy_calib_0/gen_rd_data_sel*.u_ff_rd_data_sel"
  TNM = "TNM_RD_DATA_SEL";
TIMESPEC "TS_MC_RD_DATA_SEL" = FROM "TNM_RD_DATA_SEL"
  TO "TNM_CLK0" "TS_SYS_CLK" * 4;
```

- b. Modify the UCF to set site locations for DQS Gate IDDR and IODELAY LOC constraints based on the user pinout. This process must be repeated for each DQS group:
    - i. In the PARTGEN package file, locate the line in which the “pin name” column value corresponds to the pin location of the DDR2\_DQS\_N[x] pin (that is, the “N” side of the differential strobe).
    - ii. Use the XY-location in the “pad name” column on that line, and substitute this for the LOC = ILOGIC\_xxxx and LOC = IODELAY\_xxxx constraints for that DQS group.
    - iii. For example, for a design using an XC5VLX50T-FF1136, if DDR2\_DQS\_N[0] is on pin N30, the corresponding pin name (IOB) XY-location is X0Y176. The correct values for the DQS Gate circuit IDDR and IODELAY LOC constraints are:

```
INST "*/gen_dqs[0].u_iob_dqs/u_iddr_dq_ce
  LOC = "ILOGIC_X0Y176";
INST "*/gen_dqs[0].u_iob_dqs/u_iodelay_dq_ce
  LOC = "IODELAY_X0Y176";
```

- c. Modify UCF to set the correct site location for a fabric flop driving the DQS gate signal. This flop must be placed close to the corresponding DQS gate IODELAY. This process below must be repeated for each DQS group:
- i. To determine the IDDR and IODELAY locations, locate in the PARTGEN package file the line in which the “pin name” column value corresponds to the pin location of the DDR2\_DQS\_N[x] pin (that is, the “N” side of the differential strobe).
  - ii. Use the XY-coordinate in the “nearest CLB” column on that line, and substitute this for the LOC = SLICE\_xxxx constraint for that DQS group.
  - iii. For example, for a design using an XC5VLX50T-FF1136, if DDR2\_DQS\_N[0] is on pin N30, the corresponding “nearest CLB” is X0Y88. The correct value for the DQS Gate circuit fabric flop LOC constraint is:

```
INST */u_phy_calib_0/gen_gate[0].u_en_dqs_ff"
LOC = SLICE_X0Y88;
```

- d. (NOTE: No modification required for this step.) Verify the MAXDELAY constraints that limit the length of nets associated with the DQS Gate control signal. This constrains the path for all DQS groups:

```
NET */u_phy_io_0/en_dqs*" MAXDELAY = 600 ps;
NET */u_phy_io_0/gen_dqs*.u_iob_dqs/en_dqs_sync"
MAXDELAY = 850 ps;
```

- e. (NOTE: No modification required for this step.) Verify the FROM-TO constraint that defines the path between the DQS Gate driving IDDR and the clock enable inputs to each of the data (DQ) capture IDDRs in that DQS Group. Note that this value is frequency dependent and is automatically calculated by MIG based on the memory bus clock frequency. An example for 333 MHz is shown below:

```
INST */gen_dqs[*].u_iob_dqs/u_iddr_dq_ce"
TNM = "TNM_DQ_CE_IDDR";
INST */gen_dq[*].u_iob_dq/gen_stg2*.u_iddr_dq"
TNM = "TNM_DQS_FLOPS";
TIMESPEC "TS_DQ_CE" = FROM "TNM_DQ_CE_IDDR"
TO "TNM_DQS_FLOPS" 1.60 ns;
```

- f. Modify RPM origin (RLOC\_ORIGIN) constraints for each DQ I/O. Part of the read data capture occurs in the fabric, and the relative placement of the flip-flops is fixed using a *relationally placed macro* (RPM) defined in the HDL. Each DQ has a read capture RPM associated with it, and each one must be placed correctly relative to the DQ I/O pin. This process must be repeated for each DQ data bit:
- i. Locate the correct line in the package file for the DQ of interest based on its pin number.
  - ii. Note the value in the “pad name” column. The X-coordinate of this entry is used to determine which I/O column (left = 0, center = 1, or right = 2) the DQ pin is located on.
  - iii. Note the value in the “diff pair” column. This determines whether the DQ pin is located on the slave or master site of a differential I/O pair. If the value ends in an “S”, that site is a slave site.
  - iv. If that site is a slave site, refer to the corresponding master site. This is the adjacent line whose “diff pair” entry has the same numeric value, but ends in “M” for the next step in this process (determining “nearest CLB” value). For example, for a design using an XC5VLX50T-FF1136, if DDR2\_DQ[0] is on pin T6, the “diff pair” entry for this location is 67S, which indicates it is a slave location. For the purposes of determining the “nearest CLB” location in the



next step, refer to the line above it, corresponding to location R6 (“diff pair” = 67M).

- v. Refer to the “nearest CLB” value. (Again, if this is a slave site, refer to the “nearest CLB” value for the corresponding master site.)

If the DQ site is on the *left* column, use this value directly in the RLOC\_ORIGIN constraint. For example, on an XC5VLX50T-FF1136, for a DQ pin at U25, the “nearest CLB” is X0Y80. The RLOC\_ORIGIN is:

```
INST */gen_dq[0].u_iob_dq/gen_stg2_*.u_ff_stg2a_rise"
RLOC_ORIGIN = X0Y80;
```

If the DQ site is on the *center* or *right* column, subtract 4 from the X-coordinate indicated by the “nearest CLB” value, and use this as the RLOC\_ORIGIN. For example, on an XC5VLX50T-FF1136, for a DQ located at F13, the “nearest CLB” is X52Y100. Subtracting 4 from the X-coordinate yields X48Y100. The RLOC\_ORIGIN is:

```
INST */gen_dq[0].u_iob_dq/gen_stg2_*.u_ff_stg2a_rise"
RLOC_ORIGIN = X48Y100;
```

4. The values of two top-level HDL parameters/generics—DQS\_IO\_COL and DQ\_IO\_MS—must be modified to reflect the user's specific pinout. These must be properly set in order for the HDL to correctly choose which RPMs and directed routing constraints to instantiate for each DQ read capture circuit:
  - a. Modify the value for DQS\_IO\_COL. This parameter is an array indicating the I/O column location of each of the DQS I/Os.
    - i. The length of this parameter is  $= 2 * (\# \text{ of DQS I/Os}) = 2 * \text{DQS\_WIDTH}$ .
    - ii. Determine which column each DQS I/O is located on. As in previous steps, this can be determined from the PARTGEN package file. Locate the correct line in the package file for the DQS of interest based on its pin number. The X-coordinate of this entry is used to determine which I/O column (left = 0, center = 1, or right = 2).
    - iii. Each element of the parameter consists of two bits which indicate the I/O column location of each DQS. Set the entry to 00 for the left column, 01 for the center column, or 10 for the right column. 11 is a reserved value and must not be used. The least significant two elements of the array correspond to DQS[0].  
For example, for a 32-bit, 4-strobe design with DQS[0,1] located in the left I/O column, DQS[2] located in the center I/O column, and DQS[3] located in the right I/O column, DQS\_IO\_COL is 8 bits long, and must be set to 10010000.  
**Note:** This configuration is not recommended; it is used here for illustrative purposes only.
  - b. Modify value for DQ\_IO\_MS. This parameter is an array indicating whether each DQ pin occupies a master or slave I/O location.
    - i. The length of this parameter = # of DQ I/O = DQ\_WIDTH
    - ii. Determine whether each DQ is located on a master or on a slave site. This can be determined from the “diff pair” column in the PARTGEN package file.
    - iii. Each element of the parameter is one bit, and indicates whether the corresponding DQ occupies a master or slave I/O location. Set to 0 for slave, and to 1 for master. The least significant element of the array corresponds to DQ[0].



For example, for an 8-bit, 2-strobe design with DQ[0,2,4,6,7] on master I/O locations and the other DQs on slave I/O locations, DQ\_IO\_MS is 8 bits long, and must be set to 11010101.

- c. Modify the values assigned to DQ\_IO\_MS and DQS\_IO\_COL parameters/generics in the top-level MIG (VHDL or Verilog) module based on the results of the above steps.

The remainder of this appendix describes the reasons why these additional constraints are required.

## Read Data Capture Block Diagram

The read capture path used for the MIG 2.0 or later versions of MIG Virtex-5 DDR2 interface consists of the following sub-blocks:

- DQ is initially captured using DQS in the IOB using the IDELAY and IDDR elements
- Data is transferred to the FPGA (CLK0) clock domain using a series of flops located in the fabric. The location of these flops, and the routes between the IDDR and fabric flops, must be carefully defined.
- For each DQS, a circuit is added to disable the clock enable (CE) pin to each of the corresponding DQ capture IDDRs at the end of a read burst (“DQS Gate”).

Figure B-1 shows the read capture path architecture for the MIG 2.0 or later versions of Virtex-5 DDR2 design, as well as the various portions of the capture path that are affected by the additional UCF constraints and top-level HDL parameters.

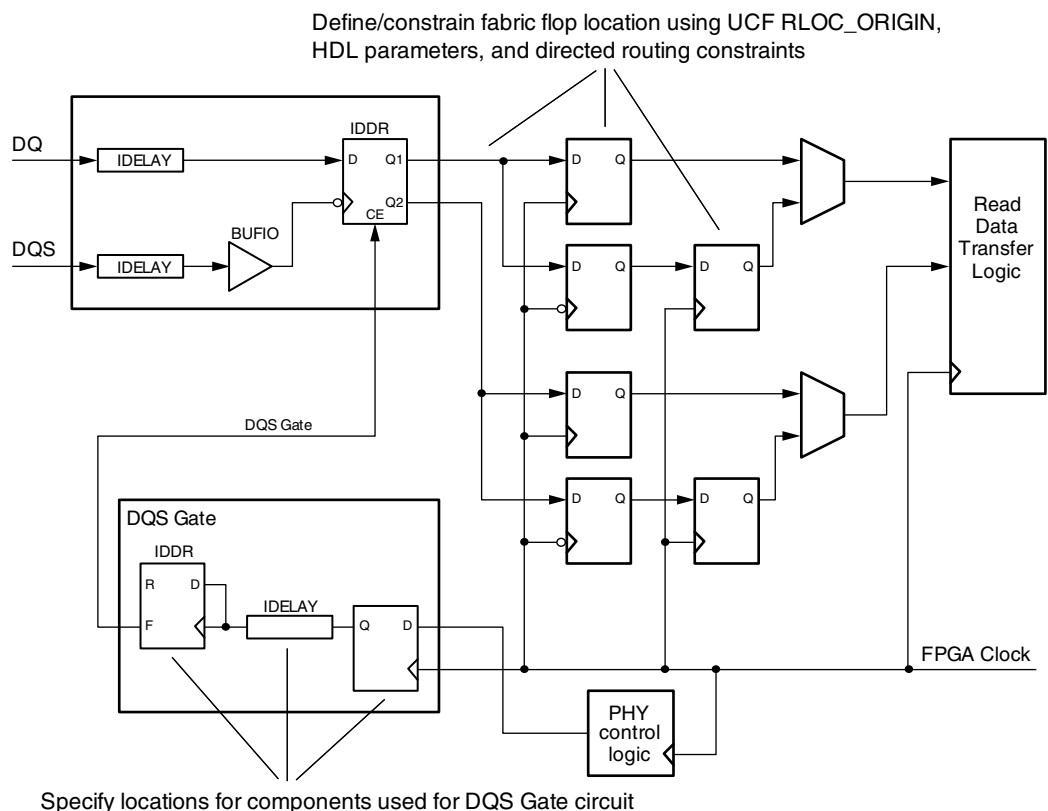


Figure B-1: Virtex-5 DDR2 Read Capture Path, MIG 2.0 or Later

## UCF / HDL Changes Overview

If the user needs to modify a MIG-generated pinout, the areas where specific constraints and parameters need to be modified are:

- HDL code top-level parameters:
  - ◆ The parameters DQS\_IO\_COL and DQ\_IO\_MS must be set according to the pin locations chosen for the DQS, and DQ IOB respectively. The rules for determining this value are outlined in section [“Setting HDL Code Top-Level Placement Parameters.”](#)
- User constraints file (UCF):
  - ◆ For each DQ pin, an RLOC\_ORIGIN must be specified. This sets the origin of the RPM for the read-capture path logic immediately next to the DQ IOB. The value of the RLOC\_ORIGIN is determined by the pin location for the corresponding DQ IOB. The rules for determining this value are outlined in section [“Setting UCF Constraints,”](#) page 416.
  - ◆ For each DQS pin:
    - a location constraint pair for an IDELAY (input delay element) and IDDR (input DDR flip-flop) must be specified. These two elements are used in the DQS Gate circuit, of which there is one per DQS group. The value of the LOC constraint for the IDELAY and IDDR are determined by the pin location for the corresponding DQS IOB. The rules for determining this value are outlined in section [“Setting UCF Constraints.”](#)
    - a location constraint for a single fabric flop must be specified. This locks the flop used to drive the DQS Gate signal close to its corresponding IDELAY. This is required to reduce the total net delay on this route, and therefore the delay fluctuations on this line due to voltage/temperature. The rules for determining this value are outlined in section [“Setting UCF Constraints.”](#)

## Setting HDL Code Top-Level Placement Parameters

The read capture path consists of dedicated circuit elements (the IDDR flop and IDELAY) embedded in the IOB, along with several flip-flops in the FPGA fabric. The placement of these fabric flip-flops is critical to providing maximum timing margin for read data capture. These flip-flops must be placed in close proximity both to each other and to the IOB. In addition, the route delays from the IOB to these fabric flip-flops must be kept as short as possible to reduce the absolute delay of each route, as well as to reduce the skew between routes from the IOB to different fabric flip-flops.

*Relationally placed macros* (RPMs) are defined within the Virtex-5 DDR2 interface HDL code. RPMs allow fixed relative placement of basic logic elements (for example, flip-flops) with respect to each other. In addition, directed routing constraints (also known as “DIRT strings”) are embedded in the code to specify the exact routing resources used for the routes from the IOB to the fabric flip-flops. RPMs and directed routing constraints are portable between different device and package combinations in the same FPGA family (for example, between XC5VLX50T-FF1136 and XC5VLX330-FG1760). There are several factors that determine RPMs and DIRT strings, which are discussed below.

There are different sets of RPM and directed routing constraints embedded in the HDL code because one set cannot account for all possible routing conditions across all pins of a device. Choosing which RPM set to enable is done on a DQ-by-DQ basis, and is determined by each DQ and DQS pin location.

In particular, the following conditions determine which set of RPM and directed routing constraints is selected for each DQ:

- **The I/O column location for the entire DQS group strobe:** Each Virtex-5 device has its IOBs arranged into three (left, center, right) columns. Each DQS group, consisting of DQ, DQS, and DM pins, must be located on the same I/O bank. (This means they must also be located on the same I/O column.) The location of fabric slice sites near the IOBs differs between the three I/O columns; therefore, different RPM sets must be supported, depending on the I/O column used.

Note that different DQS groups could be located on I/O banks in different I/O columns. Although this is allowed strictly according to I/O-placement rules, placing DQS groups in different I/O columns might make it harder for the tools to meet internal PERIOD timing. For example, internal nets need to be routed further to access DQ/DQS pins spread out across different columns.

- **Whether the DQ pin is located on a master or slave I/O:** Virtex-5 FPGA I/Os are arranged in pairs to allow for their possible use as differential pairs. The pin descriptions given in the Virtex-5 device pinout tables [Ref 12] indicate whether an I/O is the slave or master I/O for that pair. For example, on an LX50T-FF1136, pins AE22 and AD23 form an I/O pair. AE22 (IO\_L5P\_17) is the master I/O, and AD23 (IO\_L5N\_17) is the slave I/O. The status of the IOB as either a master or a slave site determines which fabric slices it uses for the read capture logic.

The following top-level parameters must be properly set in order for the code to correctly choose which RPMs and directed routing constraints to use for each DQ:

- **DQS\_IO\_COL:** This parameter is an array indicating the I/O column location of each of the DQS I/Os:
  - ◆ Length of parameter =  $2 * (\# \text{ of DQS I/O}) = 2 * \text{DQS\_WIDTH}$
  - ◆ Each element of the parameter consists of two bits that indicate the I/O column location of each DQS. Set the entry to 00 for the left column, 01 for the center column, or 10 for the right column. The 11 setting is a reserved value and must not be used. Column directionality is determined by the view as seen by FPGA Editor.
 

**Note:** This is the *opposite* of the view shown in the bank selection pane of the MIG 2.0 (or later) version of Wizard.
  - ◆ The least significant element of the array corresponds to DQS[0].
  - ◆ For example, for a 32-bit, 4-strobe design with DQS[0,1] located in the left I/O column, DQS[2] located in the center I/O column, and DQS[3] located in the right I/O column (a configuration that is not recommended, but is given here for illustrative purposes), DQS\_IO\_COL is 8 bits long and must be set to 10010000.
- **DQ\_IO\_MS:** This parameter is an array indicating whether each DQ pin occupies a master or slave I/O location.
  - ◆ Length of parameter =  $\# \text{ of DQ I/O} = \text{DQ\_WIDTH}$
  - ◆ Each element of the parameter is one bit, and indicates whether the corresponding DQ occupies a master or slave I/O location. Set to 0 for slave and 1 for master.
  - ◆ The least significant element of the array corresponds to DQ[0].
  - ◆ For example, for an 8-bit, 2-strobe design, with DQ[0,2,4,6,7] on master I/O locations and the other DQs on slave I/O locations, DQ\_IO\_MS is 8 bits long and must be set to 11010101.

## Setting UCF Constraints

Beyond the typical constraints found in a UCF (for example, PERIOD timing constraint, pinout LOC and IOSTANDARD constraints for I/O), the Virtex-5 FPGA DDR2 interface also requires that four other classes of constraints be added to the UCF:

1. **Location (LOC) constraints for the IDELAY and IDDR blocks used for every DQS Gate circuit.** There is one DQS Gate circuit per DQS I/O.
2. **RPM origin (RLOC\_ORIGIN) constraints for each DQ I/O.** These constraints exactly locate each RPM and directed routing set (as mentioned in “Setting HDL Code Top-Level Placement Parameters”) by the corresponding DQ IOB.
3. **MAXDELAY constraints to limit the delay timing-critical paths related to IOB timing.** This is not required to meet any specific cycle-to-cycle timing requirement, but rather to limit any post-calibration voltage/temperature related changes in the net delay. Voltage/temperature variations on a particular net increases as the total net delay increases.

*It is critical to reduce the delay on the DQS gate control input.* This signal is generated in the CLK0 clock domain and synchronized via an IDELAY to the DQS domain. The synchronization between the CLK0 and DQS domains on this control net is established once during initial calibration, which accounts for the static delay component of these nets. However, post-calibration changes in net delay are not accounted for, and must be minimized.

#### 4. FROM-TO constraints:

- a. One FROM-TO constraint limits the DQS Gate path from the IDDR to the DQ CE pins to approximately one-half clock cycle. This ensures that the DQ clock enables are deasserted before any possible DQS glitch at the end of the read postamble can arrive at the input to the IDDR. This value is clock-frequency dependent:

```
INST "*/gen_dqs*.u_iob_dqs/u_iddr_dq_ce"
  TNM = "TNM_DQ_CE_IDDR";
INST "*/gen_dq*.u_iob_dq/gen_stg2*.u_iddr_dq"
  TNM = "TNM_DQS_FLOPS";
TIMESPEC "TS_DQ_CE" = FROM "TNM_DQ_CE_IDDR"
  TO "TNM_DQS_FLOPS" 1.6 ns;
```

- b. Additional FROM-TO constraints define multi-cycle paths in the design. These are added to help meet internal (fabric) timing at the higher supported frequencies. At lower frequencies of operation, these multi-cycle path constraints might not be required and can be removed.

Constraint classes (1) and (2) mentioned in this section is discussed. Classes (3) and (4) is not discussed; their values do not need to change if the pinout is modified.

## Determining FPGA Element Site Locations

Setting the correct UCF constraints requires that the user have knowledge of the correct site location to use. For example, setting the correct location constraint for the IDELAY for a DQS Gate circuit requires that the user know the site name for the location where the corresponding DQS\_N pin is placed. For example, on an XC5VLX50T-FF1136, if DQS\_N[0] is located on pin C13, the user must know that the site name for this I/O is IOB\_X2Y216, so that the correct LOC constraint can be set to:

```
INST "*/gen_dqs[0].u_iob_dqs/u_idelay_dq_ce" LOC = "IODELAY_X2Y216";
```

There are two ways in which the correct site name can be determined:

- Use **FPGA Editor** to graphically determine the correct site name.
- Use **PARTGEN** to generate a package file in text format. From the package file, the correct site name can be extracted. PARTGEN can be invoked to generate package files for a specific device/package combination using the following command:

```
partgen -v <part number> (e.g. partgen -v xc5v1x330tff1738)
```

Once the package file is created, the user can search the appropriate file for the site of interest.

## Setting DQS Gate Circuit Location Constraints

Each DQS Gate circuit requires the use of an IDELAY and IDDR flip-flop in addition to fabric-based slice resources. The IDELAY and IDDR for each DQS Gate circuit, as well as the fabric flop driving the IDELAY, must be manually located in the UCF. There are three constraints for every DQS in the design.

The IDELAY and IDDR must be taken from an IOB site where these resources are available, specifically an IOB site that is used only as an output or is totally unused. This can be one of the following:

- The DQS\_N negative-side I/O site of the DQS differential I/O pair of the corresponding DQS group. A differential I/O pair does not use the input-side resources on the N-side leg of the pair.
- The DM output site for the corresponding DQS group. The DM is an output-only site, and its input-side resources are available for use by the DQS Gate circuit.
- Any IOB site that is either output-only, or unused.

The best site to use is the site that is closest in proximity on the FPGA die to the four or eight DQ I/O sites in that DQS group. This reduces the routing delay on the clock enable control from the DQS Gate circuit to its corresponding DQ sites. At higher frequencies, this can often be the critical timing path, as there is only about half a clock cycle for this path. MIG always chooses to place the IDELAY and IDDR on the DQS\_N site for the corresponding DQS group. However, depending on the particular user pinout, there might be a better site available. The user might have to relocate the DQS Gate location(s) to other sites in order to meet timing.

The IDELAY and IDDR for a given DQS Gate circuit must be placed at the same site. They cannot be placed on different sites.

The following are the constraints used for locating the IDELAY and IDDR:

```
INST "*/gen_dqs[<x>].u_iob_dqs/u_iddr_dq_ce" LOC = "ILOGIC_<SITE>";
INST "*/gen_dqs[<x>].u_iob_dqs/u_idelay_dq_ce" LOC = "IDELAY_<SITE>";
```

where <x> denotes the DQS group number, and <SITE> denotes the I/O site name.

For example, on an XC5VLX50T-FF1136, if DQS\_N[0] is placed on pin K9 and this site is chosen to locate IDELAY and IDDR for the DQS Gate circuit for DQS[0], the constraints are:

```
INST "*/gen_dqs[0].u_iob_dqs/u_iddr_dq_ce" LOC = "ILOGIC_X2Y218";
INST "*/gen_dqs[0].u_iob_dqs/u_idelay_dq_ce" LOC = "IDELAY_X2Y218";
```

The fabric flop driving the IDELAY with the DQS Gate control pulse must also be location-constrained to be near the corresponding IDELAY/IDDR. The rules for determining this are:

1. Locate the IOB to where the corresponding IDELAY and IDDR are location constrained.

- Use the appropriate package file to find the “nearest CLB” (see “[Setting RLOC\\_ORIGIN Constraints](#)”). Location-constrain this flop to this location.

For example, on an XC5VLX50T-FF1136, if DQS\_N[0] is placed on pin N30, the location constraint for the corresponding DQS Gate fabric flop is:

```
INST */gen_dqs[0].u_iob_dqs/u_iodelay_dq_ce" LOC = "IODELAY_X2Y218";
```

The reason for this requirement is to minimize the net delay from the output of this flop to the synchronizing IDELAY. (See the discussion of why MAXDELAY constraints are used in this design in section “[Setting UCF Constraints](#).”) It is possible to *not* constrain this flop to a specific location or to constrain it to a different location as long as the corresponding MAXDELAY for this net can be met (that is, by allowing MAP to place this flop).

## Setting RLOC\_ORIGIN Constraints

The RPMs for the fabric-based portion of the read capture path defined in the HDL code only specify a relative placement for each of the fabric flip-flops. An absolute origin on the FPGA chip for each RPM must also be specified, and this is done in the UCF via an RLOC\_ORIGIN constraint. There is one RLOC\_ORIGIN constraint for every data bit in the design.

Each RLOC\_ORIGIN looks like:

```
INST */gen_dq[<x>].u_iob_dq/gen_stg2_*.u_ff_stg2a_fall"
RLOC_ORIGIN = <SITE>;
```

where <x> denotes the DQ number and <SITE> denotes the appropriate fabric slice site as determined below.

The rules for determining the correct RLOC\_ORIGIN are based on the assumption that the user is referencing the appropriate device package file (generated from PARTGEN). Alternatively, the user can use a tool such as FPGA Editor to locate the correct site coordinates for each RLOC\_ORIGIN constraint.

The output of the package file looks like:

```
# PartGen J.37
#   pad          pin  vref  vcco  function          nearest  diff  tracelength
#   name         name  bank  bank  name              CLB      pair  (um)
pin  OPAD_X0Y3    AN4   -1    -1    MGTTP1_122       N.A.    N.A.  15397
pin  IPAD_X0Y19    V17    0     0     VN_0             N.A.    N.A.  3209
pin  IOB_X1Y159    L21    1     1     IO_L0P_A19_1     X28Y79  0M    6760
pin  IOB_X1Y158    L20    1     1     IO_L0N_A18_1     X27Y79  0S    6739
pin  IOB_X1Y157    L15    1     1     IO_L1P_A17_1     X28Y78  1M    8739
pin  IOB_X1Y156    L16    1     1     IO_L1N_A16_1     X27Y78  1S    6258
```

Each column represents one I/O site. The “pin name” column indicates the pin number for that site. The other column of interest is “nearest CLB”, which indicates the site name/coordinates for the nearest fabric slice to that IOB. This determines the correct RLOC\_ORIGIN value.

Unfortunately, the corresponding “nearest CLB” value cannot necessarily be used directly as the RLOC\_ORIGIN for a DQ. Instead, depending upon which I/O column (left, center, or right) the DQ is located in, an offset might need to be subtracted from the “nearest CLB” value to determine the RLOC\_ORIGIN setting.

The process is as follows:

- Locate the correct line in the package file for the DQ of interest based on its pin number.

2. Determine which I/O column (left, center, right) the DQ pin resides on. This can be determined from the package file (by looking at the “nearest CLB” value and noting its X-coordinate value), or by other means, such as FPGA Editor.
3. Look in the “diff pair” column to see if the site is a slave or master site. If it is a slave site, refer to the line in the package file for the corresponding master site for its “nearest CLB” information. For example, in the above package file, if the DQ is placed on L20 (a slave site), the line above for L21 (the corresponding master site) is referred to. This is because the master and slave site for a given I/O pair has the same RLOC\_ORIGIN value.
4. Refer to the “nearest CLB” value:
  - a. If the DQ site is on the left column, use this value directly in the RLOC\_ORIGIN constraint. For example, on an XC5VLX50T-FF1136, for a DQ pin located at U25, the “nearest CLB” is X0Y80. The RLOC\_ORIGIN is:

```
INST */gen_dq[0].u_iob_dq/gen_stg2_*.u_ff_stg2a_fall"
RLOC_ORIGIN = X0Y80;
```

Note that the same RLOC\_ORIGIN value is used if the DQ is on T25, since T25 is the slave complement to the master I/O at U25.

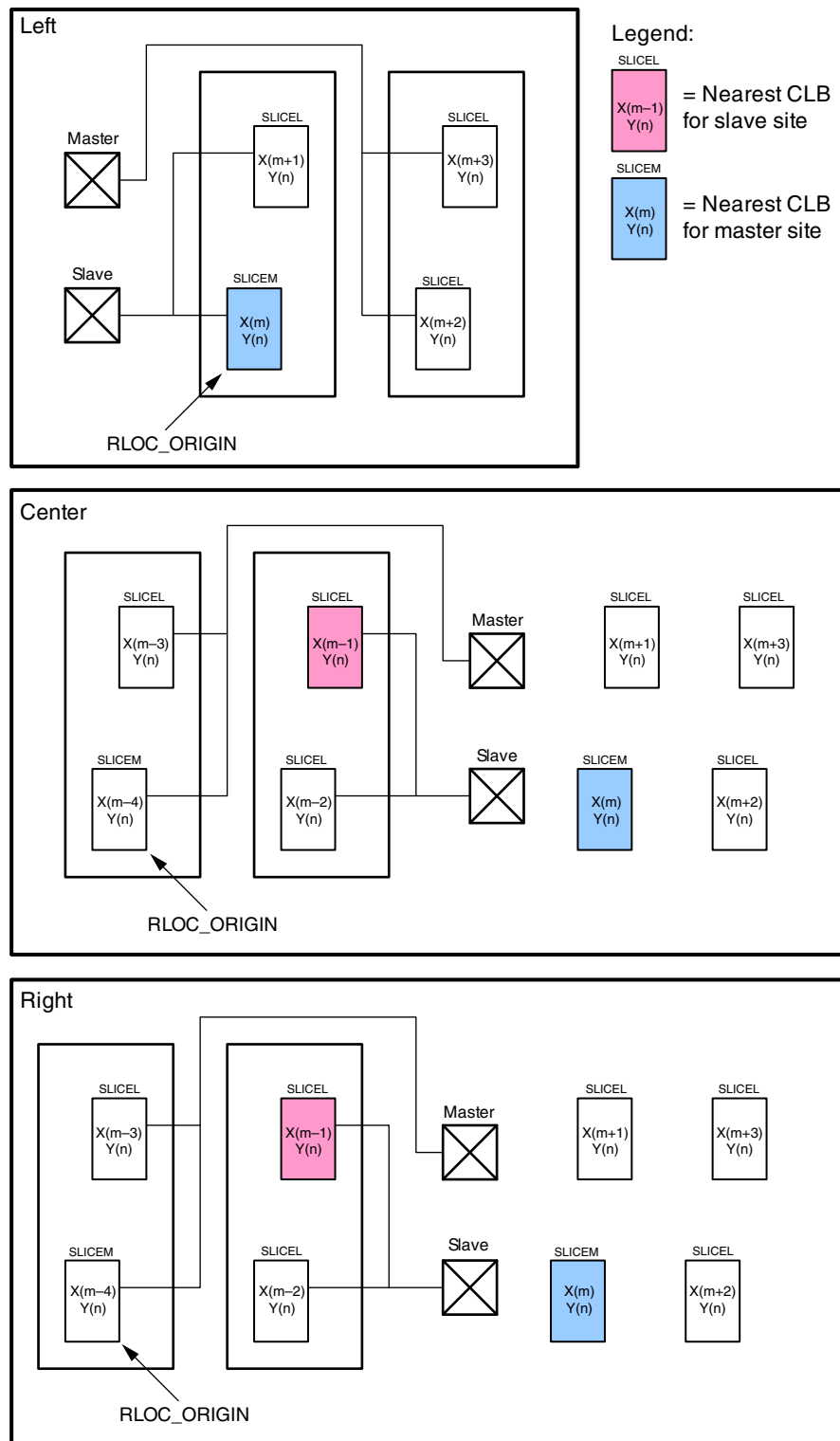
- b. If the DQ site is on the center or right column, subtract 4 from the X-coordinate indicated by the “nearest CLB” value, and use this as the RLOC\_ORIGIN. For example, on an XC5VLX50T-FF1136, for a DQ located at F13, the “nearest CLB” is X52Y100. Subtracting 4 from the X-coordinate yields X48Y100.

```
INST */gen_dq[0].u_iob_dq/gen_stg2_*.u_ff_stg2a_fall"
RLOC_ORIGIN = X48Y100;
```

The relationship between the “nearest CLB” as indicated by the package file, and the actual RPM is shown below for left, center, and right columns. Note that the RLOC\_ORIGIN values for center and right columns are calculated in the exact same manner.

[Figure B-2, page 420](#) shows the spatial relationship between the IOBs and the location of the slices that contain the flip-flops used for read data capture.





UG086\_aB\_02\_122607

Figure B-2: Calculation of RLOC\_ORIGIN



## Verifying UCF/HDL Modifications

The user can verify that the modifications to the UCF and HDL top-level files are correct through the following:

- All timing constraints (PERIOD, MAXDELAY, FROM-TO) must be met.
- The Place and Route (PAR) report must be checked to ensure that all directed routing constraints (DIRT) have been successfully routed.
  - ◆ These directed routing constraints fix the internal net routing between the IDDR and fabric-based flops. These paths are not covered by timing constraints. The user must instead verify that these directed routing constraints have been successfully routed.
  - ◆ There are two directed routing constraints for every data bit. For example, for a 72-bit design, there are 144 directed routing constraints that must be routed. The relevant PAR report section looks like:

```
INFO:ParHelpers:199 - All "EXACT" mode Directed Routing
constrained nets successfully routed. The number of
constraints found: 144, number successful: 144
```

- ◆ Failure by PAR to route certain directed routing constraints might indicate incorrect values for the HDL top-level parameters DQ\_IO\_COL and/or DQ\_IO\_MS. Another symptom of incorrect UCF or HDL values is the inadvertent placement of two RPMs for two different DQ capture circuits in the same SLICE locations (MAP error message shown below):

```
ERROR:Place:292 - The components
my_design/u_ddr2_top_0/u_mem_if_top_0/stg3b_out_fall_30 and
my_design/u_ddr2_top_0/u_mem_if_top_0/stg3b_out_fall_17 seem to
be placed / locked to the same site SLICE_X96Y42
```



# WASSO Limit Implementation Guidelines

---

This appendix provides information about WASSO (Weighted Average Simultaneous Switching Output) limit implementation in the bank selection from MIG. The number of pins selected in a bank should not exceed the WASSO limit. It is recommended to use WASSO calculator before the number of pins selected in a bank. MIG implements the WASSO for Virtex™-4 and Virtex-5 designs.

Ground bounce must be controlled to ensure proper operation of high-performance FPGA devices. Particular attention must be applied to minimizing board-level inductance during PCB layout.

When multiple output drivers change state at the same time, power supply disturbance occurs. These disturbances can cause undesired transient behavior in output drivers, input receivers, or in internal logic. These disturbances are often referred to as Simultaneous-Switching Output (SSO) noise. The SSO limits govern the number and type of I/O output drivers that can be switched simultaneously while maintaining a safe level of SSO noise.

SSO of an individual bank is calculated by summing the SSO contributions of the individual I/O standards in the bank. The SSO contribution is the percentage of full utilization of any one I/O standard in any one bank. WASSO calculation is the done by combining the SSO contributions of all I/O in a bank into a single figure.

WASSO calculation differs for Virtex-4 and Virtex-5 devices:

- *Virtex-4 User Guide* [Ref 7] provides more information on WASSO calculation for Virtex-4 devices.
- *Virtex-5 FPGA User Guide* [Ref 10] provides more information on WASSO calculation for Virtex-5 devices.

A Microsoft Excel-based spreadsheet entitled “WASSO Calculator” is provided to automate these calculations. The WASSO calculator uses PCB geometry, such as board thickness, via diameter, and breakout trace width and length, to determine board inductance. It determines the smallest undershoot and logic-Low threshold voltage among all input devices, calculates the average output capacitance, and determines the SSO allowance by taking into account all of the board-level design parameters mentioned in this document. In addition, the WASSO calculator performs checks to ensure the overall design does not exceed the SSO allowance.

The Virtex-4 FPGA WASSO Calculator [Ref 30] and the Virtex-5 FPGA WASSO Calculator [Ref 31] can be downloaded from the Xilinx website.



## Debug Port

---

### Overview

Starting with MIG 2.2, the memory controller interface design HDL for Virtex™-5, Virtex-4, and Spartan™-3 FPGAs adds ports to the top-level design file to allow debugging and monitoring of the physical layer read timing calibration logic and timing. This port consists of signals brought to the top-level HDL from the Read Calibration module (where the read timing calibration logic resides). These signals provide information for debugging hardware issues when calibration does not complete or read timing errors are observed in the system even after calibration completes. For Virtex FPGA designs, these signals also allow the user to adjust the read capture timing by adjusting the various IDELAY elements used for data synchronization. Whereas, for Spartan-3 FPGA designs, these signals allow the user to adjust the read capture timing by adjusting the delays on data\_strobe and rst\_dqs\_div signals.

Specifically, the Debug port allows the user to:

- Observe calibration status signals.
- Observe current tap values for IDELAYs used for read data synchronization for Virtex FPGA designs.
- Observe current tap\_delay values for Spartan-3 FPGA designs.
- Dynamically vary these tap values. Possible uses of this functionality include:
  - ◆ Debug read data corruption issues
  - ◆ Support periodic readjustment of the read data capture timing by adjusting the tap values
  - ◆ Use as a tool during product margining to determine actual timing margin available on read data captures

### Enabling the Debug Port

For Virtex-5 FPGA memory controller designs, the Debug port is enabled by setting the top-level HDL parameter DEBUG\_EN to 1. To disable the Debug port, set DEBUG\_EN to 0. This prevents the synthesis of additional logic required to support the Debug port (e.g., logic to allow dynamic adjustment of the IDELAY taps).

For Virtex-4 FPGA memory controller designs, the Debug port is enabled by setting the Debug Signals option in MIG.

## Signal Descriptions

The tables in this section provide the Debug port signal descriptions for the various memory and FPGA combinations. All the signal directions are with respect to the RTL design.

- Table D-1, “DDR2 SDRAM Signal Descriptions (Virtex-5 FPGAs),” page 426
- Table D-2, “DDR SDRAM Signal Descriptions (Virtex-5 FPGAs),” page 429
- Table D-3, “QDR II SRAM Signal Descriptions (Virtex-5 FPGAs),” page 432
- Table D-4, “DDR SDRAM Signal Descriptions (Virtex-4 FPGAs),” page 436
- Table D-5, “DDR II SRAM Signal Descriptions (Virtex-4 FPGAs),” page 438
- Table D-6, “QDR II SRAM Signal Descriptions (Virtex-4 FPGAs),” page 440
- Table D-7, “RLDRAM II Signal Descriptions (Virtex-4 FPGAs),” page 442
- Table D-8, “DDR/DDR2 SDRAM Signal Descriptions (Spartan-3 FPGAs),” page 443

### Virtex-5 FPGA: DDR2 SDRAM

All debug ports signals are clocked using the half-frequency clock (clkdiv). Increment and decrement control signals (e.g., dbg\_idel\_up\_all) must be provided synchronously with clkdiv. IDELAY select signals, such as dbg\_sel\_all\_idel\_dqs and dbg\_sel\_idel\_dqs can change asynchronous to clkdiv, but must meet setup and hold requirements on clkdiv on cycles when the corresponding increment/decrement control signal is asserted.

Table D-1: DDR2 SDRAM Signal Descriptions (Virtex-5 FPGAs)

Bus Name	I/O	Width	Description
dbg_calib_done	O	4	Each bit is driven to a static 1 as each stage of calibration is completed. dbg_calib_done[0] corresponds to Stage 1.
dbg_calib_dq_tap_cnt	O	6*DQ_WIDTH	6-bit tap count for each DQ IDELAY. dbg_calib_dq_tap_cnt[5:0] corresponds to DQ[0].
dbg_calib_dqs_tap_cnt	O	6*DQS_WIDTH	6-bit tap count for each DQS IDELAY. dbg_calib_dqs_tap_cnt[5:0] corresponds to DQS[0].
dbg_calib_gate_tap_cnt	O	6*DQS_WIDTH	6-bit tap count for each DQS Gate IDELAY. dbg_calib_gate_tap_cnt[5:0] corresponds to the DQS Gate for DQS[0].
dbg_calib_rd_data_sel	O	DQS_WIDTH	Each bit indicates which polarity of the FPGA clock (clk0) is used to synchronize the captured read data from the DQ IDDR for a DQS group. 1: The rising edge of clk0 synchronizes DDR2 rising edge data. The falling edge of clk0 synchronizes DDR2 falling edge data. 0: The falling edge of clk0 synchronizes DDR2 rising edge data. The rising edge of clk0 synchronizes DDR2 falling edge data. calib_rd_data_sel[0] corresponds to DQS[0].

**Table D-1: DDR2 SDRAM Signal Descriptions (Virtex-5 FPGAs) (Continued)**

Bus Name	I/O	Width	Description
dbg_calib_rden_dly	O	5*DQS_WIDTH	5-bit value indicating the number of clk0 clock cycles of delay between when a read command is issued by the controller and the synchronization of valid data in the clk0 clock domain. Each DQS group has its own distinct value. dbg_calib_rden_dly[4:0] corresponds to DQS[0].
dbg_calib_gate_dly	O	5*DQS_WIDTH	5-bit value indicating the number of clk0 clock cycles of delay between the end of a read burst and the assertion of DQS Gate. Each DQS group has its own distinct value. dbg_calib_gate_dly[4:0] corresponds to DQS[0].
dbg_calib_err	O	2	Asserted when an error is detected during calibration during stages 3 and/or 4. This appears as a 4-bit bus in the HDL. However, only bits [3:2] are used. dbg_calib_err[2] corresponds to stage 3, and dbg_calib_err[3] corresponds to stage 4. Stages 1 and 2 do not have error signals.
dbg_idel_up_all	I	1	Increments the tap value for all IDELAYs (DQ, DQS, and DQS Gate) used for read data synchronization. Tap values are incremented by one for every clkdiv cycle that this signal is held High.
dbg_idel_down_all	I	1	Decrements the tap value for all DELAYs (DQ, DQS, and DQS Gate) used for read data synchronization. Tap values are decremented by one for every clkdiv cycle that this signal is held High.
dbg_sel_all_idel_dq	I	1	Selects the functionality for dbg_idel_up_dq and dbg_idel_down_dq: 1: All DQ IDELAYs are adjusted. 0: Only the IDELAY for the DQ bit specified by dbg_sel_idel_dq is adjusted. If neither dbg_idel_up_dq nor dbg_idel_down_dq is active in a clkdiv cycle, this signal is a don't care.
dbg_sel_idel_dq	I	$\log_2(\text{DQS\_WIDTH} * \text{DQ\_PER\_DQS})$	When dbg_sel_all_idel_dq = 1, determines the specific DQ IDELAY to vary using dbg_idel_up_dq or dbg_idel_down_dq. If neither dbg_idel_up_dq nor dbg_idel_down_dq is active in a clkdiv cycle, this signal is a don't care.

Table D-1: DDR2 SDRAM Signal Descriptions (Virtex-5 FPGAs) (Continued)

Bus Name	I/O	Width	Description
dbg_idel_up_dq	I	1	Increments the tap value for all DQ IDELAYs. The DQ IDELAY(s) affected are given by dbg_sel_all_idel_dq and dbg_sel_idel_dq. Tap value(s) are incremented by one for every clkdiv cycle that this signal is held High.
dbg_idel_down_dq	I	1	Decrements the tap value for all DQ IDELAYs. The DQ IDELAY(s) affected are given by dbg_sel_all_idel_dq and dbg_sel_idel_dq. Tap value(s) are decremented by one for every clkdiv cycle that this signal is held High.
dbg_sel_all_idel_dqs	I	1	Selects the functionality for dbg_idel_up_dqs and dbg_idel_down_dqs: 1: All DQS IDELAYs are adjusted. 0: Only the IDELAY for the DQS specified by dbg_sel_idel_gate is adjusted. If neither dbg_idel_up_dqs nor dbg_idel_down_dqs is active in a clkdiv cycle, this signal is a don't care.
dbg_sel_idel_dqs	I	$\log_2(\text{DQS\_WIDTH})$	When dbg_sel_sll_idel_dqs = 1, determines the specific DQS IDELAY to vary using dbg_idel_up_dqs or dbg_idel_down_dqs. If neither dbg_idel_up_dqs nor dbg_idel_down_dqs is active in a clkdiv cycle, this signal is a don't care.
dbg_idel_up_dqs	I	1	Increments the tap value for all DQS IDELAYs. The DQS IDELAY(s) affected are given by dbg_sel_all_idel_dqs and dbg_sel_idel_dqs. Tap value(s) are incremented by one for every clkdiv cycle that this signal is held High.
dbg_idel_down_dqs	I	1	Decrements the tap value for all DQS IDELAYs. The DQS IDELAY(s) affected are given by dbg_sel_all_idel_dqs and dbg_sel_idel_dqs. Tap value(s) are decremented by one for every clkdiv cycle that this signal is held High.
dbg_sel_all_idel_gate	I	1	Selects the functionality for dbg_idel_up_gate and dbg_idel_down_gate: 1: All DQS Gate IDELAYs are adjusted. 0: Only the IDELAY for the DQS Gate specified by dbg_sel_idel_gate is adjusted.
dbg_sel_idel_gate	I	$\log_2(\text{DQS\_WIDTH})$	When dbg_sel_all_idel_gate = 1, determines the specific DQS Gate IDELAY to vary using dbg_idel_up_gate or dbg_idel_down_gate.



**Table D-1: DDR2 SDRAM Signal Descriptions (Virtex-5 FPGAs) (Continued)**

Bus Name	I/O	Width	Description
dbg_idel_up_gate	I	1	<p>Increments the tap value for all DQS Gate IDELAYs. The DQS Gate IDELAY(s) affected are given by dbg_sel_all_idel_gate and dbg_sel_idel_gate.</p> <p>Tap value(s) are incremented by one for every clkdiv cycle that this signal is held High. If neither dbg_idel_up_gate nor dbg_idel_down_gate is active in a clkdiv cycle, this signal is a don't care.</p>
dbg_idel_down_gate	I	1	<p>Decrements the tap value for all DQS Gate IDELAYs. The DQS IDELAY(s) affected are given by dbg_sel_all_idel_gate and dbg_sel_idel_gate.</p> <p>Tap value(s) are decremented by one for every clkdiv cycle that this signal is held High. If neither dbg_idel_up_gate nor dbg_idel_down_gate is active in a clkdiv cycle, this signal is a don't care.</p>

## Virtex-5 FPGA: DDR SDRAM

All debug port signals are clocked using the design clock frequency (clk90). Increment and decrement control signals (e.g., dbg\_idel\_up\_all) must be provided synchronously with clk90. IDELAY select signals, such as dbg\_sel\_all\_idel\_dqs and dbg\_sel\_idel\_dqs, can change asynchronous to clk90, but must meet setup and hold requirements on clk90 on cycles when the corresponding increment/decrement control signal is asserted.

**Table D-2: DDR SDRAM Signal Descriptions (Virtex-5 FPGAs)**

Bus Name	I/O	Width	Description
dbg_calib_done	O	4	Each bit is driven to a static 1 as each stage of calibration is completed. dbg_calib_done[0] corresponds to Stage 1.
dbg_calib_dq_tap_cnt	O	6*DQ_WIDTH	6-bit tap count for each DQ IDELAY. dbg_calib_dq_tap_cnt[5:0] corresponds to DQ[0].
Dbg_calib_dqs_tap_cnt	O	6*DQS_WIDTH	6-bit tap count for each DQS IDELAY. Dbg_calib_dqs_tap_cnt[5:0] corresponds to DQS[0].
Dbg_calib_gate_tap_cnt	O	6*DQS_WIDTH	6-bit tap count for each DQS Gate IDELAY. Dbg_calib_gate_tap_cnt[5:0] corresponds to the DQS Gate for DQS[0].
dbg_calib_rden_dly	O	5*DQS_WIDTH	5-bit value indicating the number of clk90 clock cycles of delay between when a read command is issued by the controller and the synchronization of valid data in the clk90 clock domain. Each DQS group has its own distinct value. dbg_calib_rden_dly[4:0] corresponds to DQS[0].
dbg_calib_gate_dly	O	5*DQS_WIDTH	5-bit value indicating the number of clk90 clock cycles of delay between the end of a read burst and the assertion of DQS Gate. Each DQS group has its own distinct value. dbg_calib_gate_dly[4:0] corresponds to DQS[0].

Table D-2: DDR SDRAM Signal Descriptions (Virtex-5 FPGAs) (Continued)

Bus Name	I/O	Width	Description
dbg_calib_err	O	4	Asserted when an error is detected during calibration during stages 3 and/or 4. This appears as a 4-bit bus in the HDL. However, only bits [3:2] are used. dbg_calib_err[2] corresponds to stage 3, and dbg_calib_err[3] corresponds to stage 4. Stages 1 and 2 do not have error signals.
dbg_idel_up_all	I	1	Increases the tap value for all IDELAYs (DQ, DQS, and DQS Gate) used for read data synchronization. Tap values are incremented by one for every clk90 cycle that this signal is held High.
dbg_idel_down_all	I	1	Decrements the tap value for all IDELAYs (DQ, DQS, and DQS Gate) used for read data synchronization. Tap values are decremented by one for every clk90 cycle that this signal is held High.
dbg_sel_all_idel_dq	I	1	Selects the functionality for dbg_idel_up_dq and dbg_idel_down_dq: 1: All DQ IDELAYs are adjusted. 0: Only the IDELAY for the DQ bit specified by dbg_sel_idel_dq is adjusted. If neither dbg_idel_up_dq nor dbg_idel_down_dq is active in a clk90 cycle, this signal is a don't care.
dbg_sel_idel_dq	I	$\log_2(\text{DQS\_WIDTH} * \text{DQ\_PER\_DQS})$	When dbg_sel_add_idel_dq = 1, determines the specific DQ IDELAY to vary using dbg_idel_up_dq or dbg_idel_down_dq. If neither dbg_idel_up_dq nor dbg_idel_down_dq is active in a clk90 cycle, this signal is a don't care.
dbg_idel_down_dq	I	1	Increases the tap value for all DQ IDELAYs. The DQ IDELAY(s) affected are given by dbg_sel_all_idel_dq and dbg_sel_idel_dq. Tap value(s) are incremented by one for every clk90 cycle that this signal is held High.
dbg_sel_all_idel_dqs	I	1	Decrements the tap value for all DQ IDELAYs. The DQ IDELAY(s) affected are given by dbg_sel_all_idel_dq and dbg_sel_idel_dq. Tap value(s) are decremented by one for every clk90 cycle that this signal is held High.
dbg_sel_idel_dqs	I	1	Selects the functionality for dbg_idel_up_dqs and dbg_idel_down_dqs: 1: All DQS IDELAYs are adjusted. 0: Only the IDELAY for the DQS specified by dbg_sel_idel_gate is adjusted. If neither dbg_idel_up_dqs nor dbg_idel_down_dqs is active in a clk90 cycle, this signal is a don't care.

Table D-2: DDR SDRAM Signal Descriptions (Virtex-5 FPGAs) (Continued)

Bus Name	I/O	Width	Description
dbg_idel_up_dqs	I	$\log_2(\text{DQS\_WIDTH})$	When <code>dbg_sel_add_idel_dqs = 1</code> , determines the specific DQS IDELAY to vary using <code>dbg_idel_up_dqs</code> or <code>dbg_idel_down_dqs</code> . If neither <code>dbg_idel_up_dqs</code> nor <code>dbg_idel_down_dqs</code> is active in a <code>clk90</code> cycle, this signal is a don't care.
dbg_idel_down_dqs	I	1	Increments the tap value for all DQS IDELAYs. The DQS IDELAY(s) affected are given by <code>dbg_sel_all_idel_dqs</code> and <code>dbg_sel_idel_dqs</code> . Tap value(s) are incremented by one for every <code>clk90</code> cycle that this signal is held High.
dbg_sel_all_idel_gate	I	1	Decrements the tap value for all DQS IDELAYs. The DQS IDELAY(s) affected are given by <code>dbg_sel_all_idel_dqs</code> and <code>dbg_sel_idel_dqs</code> . Tap value(s) are decremented by one for every <code>clk90</code> cycle that this signal is held High.
dbg_sel_idel_gate	I	1	Selects the functionality for <code>dbg_idel_up_gate</code> and <code>dbg_idel_down_gate</code> : 1: All DQS Gate IDELAYs are adjusted. 0: Only the IDELAY for the DQS Gate specified by <code>dbg_sel_idel_gate</code> is adjusted.
dbg_idel_up_gate	I	$\log_2(\text{DQS\_WIDTH})$	When <code>dbg_sel_add_idel_gate = 1</code> , determines the specific DQS Gate IDELAY to vary using <code>dbg_idel_up_gate</code> or <code>dbg_idel_down_gate</code> .
dbg_idel_down_gate	I	1	Increments the tap value for all DQS Gate IDELAYs. The DQS Gate IDELAY(s) affected are given by <code>dbg_sel_all_idel_gate</code> and <code>dbg_sel_idel_gate</code> . Tap value(s) are incremented by one for every <code>clk90</code> cycle that this signal is held High. If neither <code>dbg_idel_up_gate</code> nor <code>dbg_idel_down_gate</code> is active in a <code>clk90</code> cycle, this signal is a don't care.

## Virtex-5 FPGA: QDRII SRAM

All the debug input port signals are clocked using the design clock frequency (`clk0`). Increment and decrement control signals (e.g., `dbg_idel_up_all`) as well as the IDELAY select signals must be provided synchronously with `clk0`.

### Note:

1. All Data (Q) in a given calibration group has the same IDELAY tap value.
2. For x36 component designs, calibration group has both CQ and CQ# and their corresponding Data (Q) calibrated, hence the debug logic is applied to both CQ and CQ#. For x18 component designs, the calibration group has only CQ and its corresponding Data (Q) calibrated. Thus the designer must ignore the debug logic related to CQ# (e.g., `dbg_idel_up_cq_n`). The synthesis tool prunes the CQ# related logic anyway.

Table D-3: QDRII SRAM Signal Descriptions (Virtex-5 FPGAs)

Bus Name	I/O	Width	Description
dbg_idel_up_all	I	1	Increments the tap value for all IDELAYs (Q, CQ, CQ#) used for read data synchronization. Tap values are incremented by one for every clk0 cycle that this signal is held High.
dbg_idel_down_all	I	1	Decrements the tap value for all IDELAYs (Q, CQ, CQ#) used for read data synchronization. Tap values are decremented by one for every clk0 cycle that this signal is held High.
dbg_sel_all_idel_cq	I	1	Selects the functionality for dbg_idel_up_cq and dbg_idel_down_cq: 1: All CQ IDELAYs are adjusted. 0: Only the IDELAY for the CQ specified by dbg_sel_idel_cq is adjusted. If neither dbg_idel_up_cq nor dbg_idel_down_cq is active in the clk0 cycle, this signal is a don't care.
dbg_sel_idel_cq	I	CQ_WIDTH	When any dbg_sel_idel_cq bit is set to 1, it determines the specific CQ IDELAY to vary using dbg_idel_up_cq or dbg_idel_down_cq. If neither dbg_idel_up_cq nor dbg_idel_down_cq is active in the clk0 cycle, this signal is a don't care.
dbg_idel_up_cq	I	1	Increments the tap value for all CQ IDELAYs. The CQ IDELAY(s) affected are given by dbg_sel_all_idel_cq and dbg_sel_idel_cq. Tap value(s) are incremented by one for every clk0 cycle that this signal is held High.
dbg_idel_down_cq	I	1	Decrements the tap value for all CQ IDELAYs. The CQ IDELAY(s) affected are given by dbg_sel_all_idel_cq and dbg_sel_idel_cq. Tap value(s) are decremented by one for every clk0 cycle that this signal is held High.
dbg_sel_all_idel_cq_n	I	1	Selects the functionality for dbg_idel_up_cq_n and dbg_idel_down_cq_n: 1: All CQ# IDELAYs are adjusted. 0: Only the IDELAY for the CQ# specified by dbg_sel_idel_cq_n is adjusted. If neither dbg_idel_up_cq_n nor dbg_idel_down_cq_n is active in the clk0 cycle, this signal is a don't care.
dbg_sel_idel_cq_n	I	CQ_WIDTH	When any dbg_sel_idel_cq_n bit is set to 1, it determines the specific CQ# IDELAY to vary using dbg_idel_up_cq_n or dbg_idel_down_cq_n. If neither dbg_idel_up_cq_n nor dbg_idel_down_cq_n is active in the clk0 cycle, this signal is a don't care.

**Table D-3: QDRII SRAM Signal Descriptions (Virtex-5 FPGAs) (Continued)**

Bus Name	I/O	Width	Description
dbg_idel_up_cq_n	I	1	Increases the tap value for all CQ# IDELAYs. The CQ# IDELAY(s) affected are given by dbg_sel_all_idel_cq_n and dbg_sel_idel_cq_n. Tap value(s) are incremented by one for every clk0 cycle that this signal is held High.
dbg_idel_down_cq_n	I	1	Decrements the tap value for all CQ# IDELAYs. The CQ# IDELAY(s) affected are given by dbg_sel_all_idel_cq_n and dbg_sel_idel_cq_n. Tap value(s) are decremented by one for every clk0 cycle that this signal is held High.
dbg_sel_all_idel_q_cq	I	1	Selects the functionality for dbg_idel_up_q_cq and dbg_idel_down_q_cq: 1: All Data (Q) IDELAYs are adjusted. 0: Only the IDELAYs for Data (Q) in the calibration group of CQ specified by dbg_sel_idel_q_cq are adjusted. If neither dbg_idel_up_q_cq nor dbg_idel_down_q_cq is active in the clk0 cycle, this signal is a don't care.
dbg_sel_idel_q_cq	I	CQ_WIDTH	When any dbg_sel_idel_q_cq bit is set to 1, it determines all the Data (Q) IDELAYs in the calibration group of CQ to vary using dbg_idel_up_q_cq or dbg_idel_down_q_cq. If neither dbg_idel_up_q_cq nor dbg_idel_down_q_cq is active in the clk0 cycle, this signal is a don't care.
dbg_idel_up_q_cq	I	1	Increases the tap value for all Data (Q) IDELAYs in the calibration group of CQ. The Data (Q) IDELAYs in the calibration group that is affected are given by dbg_sel_all_idel_q_cq and dbg_sel_idel_q_cq. Tap value(s) are incremented by one for every clk0 cycle that this signal is held High.
dbg_idel_down_q_cq	I	1	Decrements the tap value of all Data (Q) IDELAYs in the calibration group of CQ. The Data (Q) IDELAYs in the calibration group of CQ that is affected are given by dbg_sel_all_idel_q_cq and dbg_sel_idel_q_cq. Tap value(s) are decremented by one for every clk0 cycle that this signal is held High.

Table D-3: QDRII SRAM Signal Descriptions (Virtex-5 FPGAs) (Continued)

Bus Name	I/O	Width	Description
dbg_sel_all_idel_q_cq_n	I	1	Selects the functionality for dbg_idel_up_q_cq_n and dbg_idel_down_q_cq_n: 1: All Data (Q) IDELAYs are adjusted. 0: Only the IDELAYs of all Data (Q) in the calibration group of CQ# specified by dbg_sel_idel_q_cq_n are adjusted. If neither dbg_idel_up_q_cq_n nor dbg_idel_down_q_cq_n is active in the clk0 cycle, this signal is a don't care.
dbg_sel_idel_q_cq_n	I	CQ_WIDTH	When any dbg_sel_idel_q_cq_n bit is set to 1, it determines all the Data (Q) IDELAYs in the calibration group of CQ# to vary using dbg_idel_up_q_cq_n or dbg_idel_down_q_cq_n. If neither dbg_idel_up_q_cq_n nor dbg_idel_down_q_cq_n is active in the clk0 cycle, this signal is a don't care.
dbg_idel_up_q_cq_n	I	1	Increments the tap value of all Data (Q) IDELAYs in the calibration group of CQ#. The Data (Q) IDELAYs in the calibration group of CQ# that is affected are given by dbg_sel_all_idel_q_cq_n and dbg_sel_idel_q_cq_n. Tap value(s) are incremented by one for every clk0 cycle that this signal is held High.
dbg_idel_down_q_cq_n	I	1	Decrements the tap value of all Data (Q) IDELAYs in the calibration group of CQ#. The Data (Q) IDELAYs in the calibration group of CQ# that is affected are given by dbg_sel_all_idel_q_cq_n and dbg_sel_idel_q_cq_n. Tap value(s) are decremented by one for every clk0 cycle that this signal is held High.
dbg_init_count_done	O	1	When set to 1, indicates the completion of memory initialization.
dbg_q_cq_init_delay_done	O	CQ_WIDTH	When set to 1, indicates the completion of the first stage calibration with respect to CQ.
dbg_q_cq_init_delay_done_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for each group of Data (Q) bits IDELAY associated with CQ. dbg_q_cq_init_delay_done_tap_count[5:0] corresponds to CQ[0].
dbg_q_cq_n_init_delay_done	O	CQ_WIDTH	When set to 1, indicates the completion of the first stage calibration with respect to CQ#.
dbg_q_cq_n_init_delay_done_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for each group of Data (Q) bits IDELAY associated with CQ#. dbg_q_cq_n_init_delay_done_tap_count[5:0] corresponds to CQ#[0].

Table D-3: QDRII SRAM Signal Descriptions (Virtex-5 FPGAs) (Continued)

Bus Name	I/O	Width	Description
dbg_cq_cal_done	O	CQ_WIDTH	When set to 1, indicates the completion of the second stage calibration with respect to CQ.
dbg_cq_cal_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for each CQ IDELAY. dbg_cq_cal_tap_count[5:0] corresponds to CQ[0].
dbg_cq_n_cal_done	O	CQ_WIDTH	When set to 1, indicates the completion of the second stage calibration with respect to CQ#.
dbg_cq_n_cal_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for each CQ# IDELAY. dbg_cq_n_cal_tap_count[5:0] corresponds to CQ#[0].
dbg_we_cal_done_cq	O	CQ_WIDTH	When set to 1, indicates the completion of the read enable calibration of the Data (Q) in the calibration group of each CQ.
dbg_we_cal_done_cq_n	O	CQ_WIDTH	When set to 1, indicates the completion of the read enable calibration of the Data (Q) in the calibration group of CQ#.
dbg_cq_q_data_valid	O	CQ_WIDTH	When set to 1, indicates the data valid signal for the Data (Q) in the calibration group of each CQ.
dbg_cq_n_q_data_valid	O	CQ_WIDTH	When set to 1, indicates the data valid signal for the Data (Q) in the calibration group of each CQ#.
dbg_cal_done	O	1	When set to 1, indicates the completion of the Data (Q) calibration process.
dbg_data_valid	O	1	When set to 1, indicates the data valid signal for the Read Data (Q) after calibration.

## Virtex-4 FPGA: DDR SDRAM

All the debug input port signals are clocked using the design clock frequency (clk). Increment and decrement control signals (e.g., dbg\_idel\_up\_all) as well as the IDELAY select signals must be provided synchronously with clk.

**Note:**

1. All Read Data (DQ) in a given calibration group has the same IDELAY tap value.
2. The READENABLE value is determined by the number of banks used for the allocation of data and strobe signals.
3. A calibration group is determined by the number of Data (DQ) and corresponding Strobes (DQS) together in a single bank.

Table D-4: DDR SDRAM Signal Descriptions (Virtex-4 FPGAs)

Bus Name	I/O	Width	Description
dbg_idel_up_all	I	1	Increments the tap value for all Read Data (DQ) IDELAYs used for data synchronization. Tap values are incremented by one for every clk cycle that this signal is held High.
dbg_idel_down_all	I	1	Decrements the tap value for all Read Data (DQ) IDELAYs used for data synchronization. Tap values are decremented by one for every clk cycle that this signal is held High.
dbg_sel_all_idel_dq	I	1	Selects the functionality for dbg_idel_up_dq and dbg_idel_down_dq: 1: All Read Data (DQ) IDELAYs are adjusted. 0: Only the IDELAY for the Read Data (DQ) specified by dbg_sel_idel_dq is adjusted. If neither dbg_idel_up_dq nor dbg_idel_down_dq is active in the clk cycle, this signal is a don't care.
dbg_sel_idel_dq	I	READENABLE	When any dbg_sel_idel_dq bit is set to 1, it determines the specific set of Read Data (DQ) IDELAYs in the calibration group to vary using dbg_idel_up_dq or dbg_idel_down_dq. If neither dbg_idel_up_dq nor dbg_idel_down_dq is active in the clk cycle, this signal is a don't care.
dbg_idel_up_dq	I	1	Increments the tap value for all Read Data (DQ) IDELAYs in the calibration group. The Read Data (DQ) IDELAYs in the calibration group, which are affected, are given by dbg_sel_all_idel_dq and dbg_sel_idel_dq. Tap value(s) are incremented by one for every clk cycle that this signal is held High.
dbg_idel_down_dq	I	1	Decrements the tap value for all Read Data (DQ) IDELAYs in the calibration group. The Read Data (DQ) IDELAYs in the calibration group, which are affected, are given by dbg_sel_all_idel_dq and dbg_sel_idel_dq. Tap value(s) are decremented by one for every clk cycle that this signal is held High.
dbg_dqs_first_edge_detect	O	READENABLE	When set to 1, indicates the detection of the first edge of DQS in each calibration group.
dbg_dqs_first_edge_tap_count	O	6*READENABLE	A 6-bit tap count for each DQS IDELAY in each calibration group. This value determines the number of IDELAY taps incremented for first edge detection of DQS.
dbg_dqs_second_edge_detect	O	READENABLE	When set to 1, indicates the detection of the second edge of DQS in each calibration group.



Table D-4: DDR SDRAM Signal Descriptions (Virtex-4 FPGAs) (Continued)

Bus Name	I/O	Width	Description
dbg_dqs_second_edge_tap_count	O	6*READENABLE	A 6-bit tap count for each DQS IDELAY in each calibration group. This value determines the number of IDELAY taps incremented for second edge detection.
dbg_dqs_tap_sel_done	O	READENABLE	When set to 1, indicates that the calibration process of the center-aligning DQS with respect to clk in each calibration group is complete.
dbg_dqs_tap_count	O	6*READENABLE	A 6-bit tap count for DQS IDELAY in each calibration group. This value determines the number of IDELAY taps incremented. The maximum counter value cannot be more than 64, since the maximum taps that an IDELAY element can be incremented is only 64 taps.
dbg_data_tap_count	O	6*READENABLE	A 6-bit tap count for each group of Read Data (DQ) IDELAYs in each calibration group. The counter value indicates the number of tap delays that are to be applied on group of Read Data (DQ) IDELAYs.
dbg_data_tap_sel_done	O	READENABLE	When set to 1, indicates the completion of delaying the group of Read Data (DQ) IDELAYs in each calibration group. The number of taps that are to be delayed is determined by dbg_data_tap_count.
dbg_first_rising	O	READENABLE	1: The first edge detected is rising edge. 0: The first edge detected is falling edge.
dbg_ctrl_dummyread_start	O	1	When set to 1, indicates that the read data calibration is in progress.

## Virtex-4 FPGA: DDRII SRAM

All the debug input port signals are clocked using the design clock frequency (clk\_0). Increment and decrement control signals (e.g., dbg\_idel\_up\_all) as well as the IDELAY select signals must be provided synchronously with clk\_0.

**Note:**

1. All Data (DQ) in a given calibration group has the same IDELAY tap value.
2. A calibration group is determined by the number of Data (DQ) associated with each CQ.

Table D-5: DDRII SRAM Signal Descriptions (Virtex-4 FPGAs)

Bus Name	I/O	Width	Description
dbg_idel_up_all	I	1	Increments the tap value for all Read Data (DQ) IDELAYs used for data synchronization. Tap values are incremented by one for every clk_0 cycle that this signal is held High.
dbg_idel_down_all	I	1	Decrements the tap value for all Read Data (DQ) IDELAYs used for data synchronization. Tap values are decremented by one for every clk_0 cycle that this signal is held High.
dbg_sel_all_idel_data_cq	I	1	Selects the functionality for dbg_idel_up_data_cq and dbg_idel_down_data_cq: 1: All DQ IDELAYs are adjusted. 0: Only the IDELAY for all the DQ IDELAYs in the calibration group specified by dbg_sel_idel_data_cq is adjusted. If neither dbg_idel_up_data_cq nor dbg_idel_down_data_cq is active in the clk_0 cycle, this signal is a don't care.
dbg_sel_idel_data_cq	I	CQ_WIDTH	When any dbg_sel_idel_data_cq bit is set to 1, it determines all the Read Data (DQ) IDELAYs in a calibration group to vary using dbg_idel_up_data_cq or dbg_idel_down_data_cq. If neither dbg_idel_up_data_cq nor dbg_idel_down_data_cq is active in the clk_0 cycle, this signal is a don't care.
dbg_idel_up_data_cq	I	1	Increments the tap value for all Read Data (DQ) IDELAYs in a calibration group. The Read Data (DQ) IDELAYs in a calibration group which are affected are given by dbg_sel_all_idel_data_cq and dbg_sel_idel_data_cq. Tap value(s) are incremented by one for every clk_0 cycle, this signal is held High.
dbg_idel_down_data_cq	I	1	Decrements the tap value for all Read Data (DQ) IDELAYs in a calibration group. The Read Data (DQ) IDELAYs in a calibration group, which are affected, are given by dbg_sel_all_idel_data_cq and dbg_sel_idel_data_cq. Tap value(s) are decremented by one for every clk_0 cycle, this signal is held High.
dbg_cq_first_edge_detect	O	CQ_WIDTH	When set to 1, indicates the detection of the first edge of CQ in each calibration group.
dbg_cq_first_edge_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for CQ IDELAY in each calibration group. This value determines the number of IDELAY taps incremented for first edge detection. dbg_cq_first_edge_tap_count[5:0] corresponds to CQ[0]
dbg_cq_second_edge_detect	O	CQ_WIDTH	When set to 1, indicates the detection of the second edge of CQ in each calibration group.

**Table D-5: DDRII SRAM Signal Descriptions (Virtex-4 FPGAs) (Continued)**

Bus Name	I/O	Width	Description
dbg_cq_second_edge_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for CQ IDELAY in each calibration group. This value determines the number of IDELAY taps incremented for second edge detection. dbg_cq_second_edge_tap_count[5:0] corresponds to CQ[0].
dbg_cq_tap_sel_done	O	CQ_WIDTH	When set to 1, indicates that the calibration process of the center-aligning CQ with respect to clk_0 in each calibration group is completed.
dbg_cq_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for CQ IDELAY in each calibration group. This value determines the number of IDELAY taps incremented. The maximum counter value cannot be more than 64, since the maximum taps that an IDELAY element can be incremented is only 64 taps.
dbg_data_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for all Read Data (DQ) IDELAYs in each calibration group. The counter value indicates the number of tap delays that are to be applied on all Read Data (DQ) IDELAYs in each calibration group.
dbg_data_tap_sel_done	O	CQ_WIDTH	When set to 1, indicates the completion of delaying all the Read Data (DQ) IDELAYs in each calibration group. The number of taps that are to be delayed is determined by dbg_data_tap_count.
dbg_first_rising	O	CQ_WIDTH	1: The first edge detected is rising edge. 0: The first edge detected is falling edge.
dbg_rdcmd2valid_cnt	O	5*CQ_WIDTH	A 5-bit counter to calculate number of clocks from controller read command to data valid for group of Read Data (DQ) associated with specific CQ.
dbg_dly_cal_done	O	1	When set to 1, indicates the completion of the Read Data (DQ) calibration process.

## Virtex-4 FPGA: QDRII SRAM

All the debug input port signals are clocked using the design clock frequency (clk\_0). Increment and decrement control signals (e.g., dbg\_idel\_up\_all) as well as the IDELAY select signals must be provided synchronously with clk\_0.

**Note:**

1. All Data (Q) in a given calibration group has the same IDELAY tap value.
2. A calibration group is determined by the number of Data (Q) associated with each CQ.

Table D-6: QDRII SRAM Signal Descriptions (Virtex-4 FPGAs)

Bus Name	I/O	Width	Description
dbg_idel_up_all	I	1	Increases the tap value for all Data (Q) IDELAYs used for data synchronization. Tap values are incremented by one for every clk_0 cycle that this signal is held High.
dbg_idel_down_all	I	1	Decrements the tap value for all Data (Q) IDELAYs used for data synchronization. Tap values are decremented by one for every clk_0 cycle that this signal is held High.
dbg_sel_all_idel_data_cq	I	1	Selects the functionality for dbg_idel_up_data_cq and dbg_idel_down_data_cq: 1: All Q IDELAYs are adjusted. 0: Only the IDELAY for all Data (Q) in the calibration specified by dbg_sel_idel_data_cq is adjusted. If neither dbg_idel_up_data_cq nor dbg_idel_down_data_cq is active in the clk_0 cycle, this signal is a don't care.
dbg_sel_idel_data_cq	I	CQ_WIDTH	When any dbg_sel_idel_data_cq bit is set to 1, it determines all the Data (Q) IDELAYs in the calibration group to vary using dbg_idel_up_data_cq or dbg_idel_down_data_cq. If neither dbg_idel_up_data_cq nor dbg_idel_down_data_cq is active in the clk_0 cycle, this signal is a don't care.
dbg_idel_up_data_cq	I	1	Increases the tap value for all Data (Q) IDELAYs in the calibration group. The Data (Q) IDELAYs in the calibration group which are affected are given by dbg_sel_all_idel_data_cq and dbg_sel_idel_data_cq. Tap value(s) are incremented by one for every clk_0 cycle that this signal is held High.
dbg_idel_down_data_cq	I	1	Decrements the tap value for all Data (Q) IDELAYs in the calibration group. The Data (Q) IDELAYs in the calibration group, which are affected, are given by dbg_sel_all_idel_data_cq and dbg_sel_idel_data_cq. Tap value(s) are decremented by one for every clk_0 cycle that this signal is held High.
dbg_cq_first_edge_detect	O	CQ_WIDTH	When set to 1, indicates the detection of the first edge of CQ in each calibration group.
dbg_cq_first_edge_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for CQ IDELAY in each calibration group. This value determines the number of IDELAY taps incremented for first edge detection. dbg_cq_first_edge_tap_count[5:0] corresponds to CQ[0].
dbg_cq_second_edge_detect	O	CQ_WIDTH	When set to 1, indicates the detection of the second edge of CQ in each calibration group.

**Table D-6: QDR II SRAM Signal Descriptions (Virtex-4 FPGAs) (Continued)**

Bus Name	I/O	Width	Description
dbg_cq_second_edge_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for CQ IDELAY in each calibration group. This value determines the number of IDELAY taps incremented for second edge detection. dbg_cq_second_edge_tap_count[5:0] corresponds to CQ[0].
dbg_cq_tap_sel_done	O	CQ_WIDTH	When set to 1, indicates that the calibration process of the center-aligning CQ with respect to clk_0 in each calibration group.
dbg_cq_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for CQ IDELAY in each calibration group. This value determines the number of IDELAY taps incremented. The maximum counter value cannot be more than 64; since the maximum taps that an IDELAY element can be incremented is only 64 taps.
dbg_data_tap_count	O	6*CQ_WIDTH	A 6-bit tap count for all Data (Q) IDELAYs in each calibration group. The counter value indicates the number of tap delays that are to be applied on all Data (Q) IDELAYs in each calibration group.
dbg_data_tap_sel_done	O	CQ_WIDTH	When set to 1, indicates the completion of delaying all the Data (Q) IDELAYs in the calibration group. The number of taps that are to be delayed is determined by dbg_data_tap_count.
dbg_first_rising	O	CQ_WIDTH	1: The first edge detected is rising edge. 0: The first edge detected is falling edge.
dbg_rdcmd2valid_cnt	O	5*CQ_WIDTH	A 5-bit counter to calculate number of clocks from controller read command to data valid for group of Data (Q) associated with specific CQ.
dbg_dly_cal_done	O	1	When set to 1, indicates the completion of the Data (Q) calibration process.

## Virtex-4 FPGA: RLDRAM II

All the debug input port signals are clocked using the design clock frequency (clkglob). Increment and decrement control signals (e.g., dbg\_idel\_up\_all) as well as the IDELAY select signals must be provided synchronously with clkglob.

**Note:**

1. All Read Data (DQ) in a given calibration group has the same IDELAY tap value.
2. A calibration group is determined by the number of Read Data (DQ) associated with each QK.

Table D-7: RLD RAM II Signal Descriptions (Virtex-4 FPGAs)

Bus Name	I/O	Width	Description
dbg_idel_up_all	I	1	Increments the tap value for all Read Data (DQ) IDELAYs used for read data synchronization. Tap values are incremented by one for every clkglob cycle that this signal is held High.
dbg_idel_down_all	I	1	Decrements the tap value for all Read Data (DQ) IDELAYs used for read data synchronization. Tap values are decremented by one for every clkglob cycle that this signal is held High.
dbg_sel_all_idel_data_qk	I	1	Selects the functionality for dbg_idel_up_data_qk and dbg_idel_down_data_qk: 1: All Read Data (DQ) IDELAYs are adjusted. 0: Only the IDELAYs for all the Read Data (DQ) in a calibration group specified by dbg_sel_idel_data_qk is adjusted. If neither dbg_idel_up_data_qk nor dbg_idel_down_data_qk is active in the clkglob cycle, this signal is a don't care.
dbg_sel_idel_data_qk	I	QK_WIDTH	When any dbg_sel_idel_data_qk bit is set to 1, it determines all the Data (DQ) IDELAYs in a calibration group to vary using dbg_idel_up_data_qk or dbg_idel_down_data_qk. If neither dbg_idel_up_data_qk nor dbg_idel_down_data_qk is active in the clkglob cycle, this signal is a don't care.
dbg_idel_up_data_qk	I	1	Increments the tap value for all Data (DQ) IDELAYs in a calibration group. The Data (DQ) IDELAYs in a calibration group, which are affected, are given by dbg_sel_all_idel_data_qk and dbg_sel_idel_data_qk. Tap value(s) are incremented by one for every clkglob cycle that this signal is held High.
dbg_idel_down_data_qk	I	1	Decrements the tap value for all Data (DQ) IDELAYs in a calibration group. The Data (DQ) IDELAYs in a calibration group, which are affected, are given by dbg_sel_all_idel_data_qk and dbg_sel_idel_data_qk. Tap value(s) are decremented by one for every clkglob cycle that this signal is held High.
dbg_qk_first_edge	O	QK_WIDTH	When set to 1, indicates the detection of the first edge of QK in a calibration group.
dbg_qk_first_edge_tap_count	O	6*QK_WIDTH	A 6-bit tap count for QK IDELAY in each calibration group. This value determines the number of IDELAY taps incremented for first edge detection. dbg_qk_first_edge_tap_count[5:0] corresponds to QK[0].
dbg_qk_second_edge	O	QK_WIDTH	When set to 1, indicates the detection of the second edge of QK in a calibration group.

**Table D-7: RLDRAM II Signal Descriptions (Virtex-4 FPGAs) (Continued)**

Bus Name	I/O	Width	Description
dbg_qk_second_edge_tap_count	O	6*QK_WIDTH	A 6-bit tap count for QK IDELAY in each calibration group. This value determines the number of IDELAY taps incremented for second edge detection. dbg_qk_second_edge_tap_count[5:0] corresponds to QK[0].
dbg_qk_tap_count	O	6*QK_WIDTH	A 6-bit counter for QK IDELAY in each calibration group. This value determines the number of IDELAY taps incremented. The maximum counter value cannot be more than 64, since the maximum taps that an IDELAY element can be incremented is only 64 taps.
dbg_first_rising	O	QK_WIDTH	1: Indicates that the first edge detected is rising edge. 0: Indicates that the first edge detected is falling edge.
dbg_qk_tap_sel_done	O	QK_WIDTH	When set to 1, indicates that the calibration process of the center-aligning clkglob with respect to that particular QK is complete.
dbg_data_tap_count	O	6*QK_WIDTH	A 6-bit tap count for all the Read Data (DQ) IDELAYs in a calibration group. The counter value indicates the number of tap delays that are to be applied on group of Read Data (DQ) IDELAYs.
dbg_data_tap_sel_done	O	QK_WIDTH	When set to 1, indicates the completion of delaying the all the Read Data (DQ) IDELAYs in a calibration group. The number of taps that are to be delayed is determined by dbg_data_tap_count.

## Spartan-3 FPGA: DDR/DDR2 SDRAMs

**Table D-8: DDR/DDR2 SDRAM Signal Descriptions (Spartan-3 FPGAs)**

Signal Name	I/O	Width	Description
dbg_delay_sel	O	5	Tap value from the calibration logic used to delay the strobe and rst_dqs_div.
dbg_rst_calib	O	1	Used to stop new tap_values from calibration logic to strobe and rst_dqs_div during memory read operations.
dbg_phase_cnt	O	5	Phase count gives the number of LUTs in the clock phase.
dbg_cnt	O	6	Counter used in the calibration logic.
dbg_trans_onedtct	O	1	Asserted when the first transition is detected.
dbg_trans_twodtct	O	1	Asserted when the second transition is detected.
dbg_enb_trans_two_dtct	O	1	Enable signal for dbg_trans_twodtct.
vio_out_dqs_en	I	1	Enable signal for strobe tap selection.
vio_out_dqs	I	5	Used to change the tap values for strobes.
vio_out_rst_dqs_div_en	I	1	Enable signal for rst_dqs_div tap selection.
vio_out_rst_dqs_div	I	5	Used to change the tap values for rst_dqs_div.

## Adjusting the Tap Delays

The Debug port can be used for dynamic adjustment of tap delays. This can be initiated either through a Xilinx Virtual I/O (VIO) module or through other custom control logic.

### Virtex FPGA Designs

This section describes the procedure for adjusting the IDELAY taps for the DDR2 SDRAM Virtex-5 FPGA design. This tap adjusting procedure is applicable for DDR2 SDRAM and DDR SDRAM Virtex-5 FPGA designs only.

1. If all IDELAY taps used in the DDR2 interface (for all DQ, DQS, and DQS Gate) must be adjusted at once:
  - a. Assert either `dbg_sel_idel_up_all` or `dbg_sel_idel_down_all`. For every `clkdiv` cycle where one or the other of these two signals is asserted, the IDELAY taps are incremented or decremented by 1.
  - b. To exactly control the amount of adjustment when using VIO to control these signals, the user should make sure these control signals are set to generate a single pulse one clock cycle wide when selected.
2. If all DQ IDELAYs must be adjusted at once:
  - a. Set `dbg_sel_all_idel_gate = 1`.
  - b. Use `dbg_idel_up_dq` or `dbg_idel_down_dq` to either increment or decrement all DQ IDELAYs at once. As is the case with `dbg_sel_idel_up_all`, these control signals increment or decrement the IDELAY tap count by 1 for every `clkdiv` cycle they are asserted.
3. If only a specific DQ IDELAY must be adjusted:
  - a. Set `dbg_sel_all_idel_dq = 0`.
  - b. Set `dbg_sel_idel_dq` to indicate the specific DQ IDELAY to be adjusted. For example, for a 32-bit DDR2 interface where `DQ[10]` must be adjusted, the user sets `dbg_sel_idel_dq[4:0] = 01010`.
  - c. Use `dbg_idel_up_dq` or `dbg_idel_down_dq` to either increment or decrement the specified DQ IDELAY.
4. The procedure for adjusting all or individual DQS or DQS Gate IDELAY tap values is the same as outlined in [step 2](#) and [step 3](#), except that separate ports are provided for DQS and DQS Gate IDELAY adjustment.

This next procedure is for the QDRII SRAM Virtex-5 FPGA design:

1. If all IDELAY taps used in the QDRII interface (for all Read Data (Q) and Strobes (CQ, CQ#)) must be adjusted at once:
  - a. Assert either `dbg_sel_idel_up_all` or `dbg_sel_idel_down_all`. For every `clk0` cycle where one or the other of these two signals is asserted, the IDELAY taps are incremented or decremented by 1.
  - b. To exactly control the amount of adjustment when using VIO to control these signals, the user should make sure these control signals are set to generate a single pulse one clock cycle wide when selected.
2. If all CQ or CQ# IDELAYs must be adjusted at once:
  - a. Use `dbg_idel_up_cq` or `dbg_idel_down_cq` to either increment or decrement all CQ IDELAYs at once, when `dbg_sel_all_idel_cq` is set to 1.



- b. Use `dbg_idel_up_cq_n` or `dbg_idel_down_cq_n` to either increment or decrement all CQ# IDELAYs at once, when `dbg_sel_all_idel_cq_n` is set to 1.  
As is the case with `dbg_sel_idel_up_all` or `dbg_sel_idel_down_all`, these control signals increment or decrement the IDELAY tap count by 1 for every `clk0` cycle they are asserted.
- 3. If only a specific CQ or CQ# IDELAY must be adjusted:
  - a. Set `dbg_sel_all_idel_cq = 0` and set `dbg_sel_idel_cq` to indicate the specific CQ IDELAY to be adjusted. For example, for a x36 QDRII component interface with a 72-bit data width where CQ[1] must be adjusted, the user sets `dbg_sel_idel_cq[1:0] = 10`.
  - b. Set `dbg_sel_all_idel_cq_n = 0` and set `dbg_sel_idel_cq_n` to indicate the specific CQ# IDELAY to be adjusted. For example, for a x36 QDRII component interface with a 72-bit data width where CQ#[1] must be adjusted, the user sets `dbg_sel_idel_cq_n[1:0] = 10`.
  - c. Use `dbg_idel_up_cq` or `dbg_idel_down_cq` to either increment or decrement the specified CQ IDELAY.
- 4. The procedure for adjusting all or calibration group Read Data (Q) IDELAY tap values is the same as outlined in [step 2](#) and [step 3](#), except that separate ports are provided for Read Data (Q) IDELAY adjustment.

The above mentioned tap adjustment procedure is applicable for QDRII SRAM Virtex-5 FPGA designs and DDR SDRAM, DDRII SRAM, RLDRAM II, QDRII SRAM Virtex-4 FPGA designs.

## Spartan-3 FPGA Designs

The procedure for adjusting the tap delay values is as follows:

- 1. Adjust the tap delay values for all the strobes (DQS):
  - a. Set `vio_out_dqs_en = 1`.
  - b. Use `vio_out_dqs[4:0]` to change the tap values (see [Table D-9](#)).

**Table D-9: Tap Values for Strobes**

<b>vio_out_dqs[4:0]</b>	<b>Tap Value</b>
01111 (0x0F)	Tap 1
10111 (0x17)	Tap 2
11011 (0x1B)	Tap 3
11101 (0x1D)	Tap 4
11110 (0x1E)	Tap 5
11111 (0x1F)	Tap 6

- 2. Adjust the tap delay values for `rst_dqs_div` (loopback signal):
  - a. Set `vio_out_rst_dqs_div_en = 1`.
  - b. Use `vio_out_rst_dqs_div[4:0]` to change the tap values (see [Table D-10](#)).

Table D-10: Tap Values for Loopback Signal

<code>vio_out_rst_dqs_div[4:0]</code>	Tap Value
01111 (0x0F)	Tap 1
10111 (0x17)	Tap 2
11011 (0x1B)	Tap 3
11101 (0x1D)	Tap 4
11110 (0x1E)	Tap 5
11111 (0x1F)	Tap 6

3. Adjust the tap delay values for all the strobes (DQS) and `rst_dqs_div`:
  - a. Set `vio_out_dqs_en` = 1.
  - b. Set `vio_out_rst_dqs_div_en` = 1.
  - c. Set the tap values for `rst_dqs_div` and all the strobes from [Table D-9](#) and [Table D-10](#) by changing `vio_out_dqs[4:0]` and `vio_out_rst_dqs_div[4:0]`.

## Sample Control/Monitoring of the Debug Port

HDL code for the Spartan-3, Virtex-4, and Virtex-5 FPGA Debug ports can be generated from MIG by selecting the Debug Signals option. Spartan-3 FPGA designs use VIO, ILA, and ICON cores generated using the ChipScope™ Pro tool to monitor the calibration signals and tap values, as well as allow dynamic adjustment of the tap delay values. Virtex-4 and Virtex-5 FPGA designs use VIO cores generated using the ChipScope Pro tool to monitor both calibration status and IDELAY tap values, as well as allow dynamic adjustment of the IDELAY tap values.