

# **External Memory Interface Handbook**

## **Volume 1: Altera Memory Solution Overview, Design Flow, and General Information**

 [Subscribe](#)  
 [Send Feedback](#)

Last updated for Quartus Prime Design Suite: 16.0

**EMI\_GS**  
2016.05.02

101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)

**ALTERA**  
now part of Intel

# Contents

<b>Introduction to Altera Memory Solution.....</b>	<b>1-1</b>
Memory Solutions.....	1-1
Protocol Support Matrix.....	1-3
Arria 10 EMIF Future Protocol Support.....	1-5
Document Revision History.....	1-6
<b>Recommended Design Flow.....</b>	<b>2-1</b>
Getting Started With External Memory Interfaces.....	2-3
Selecting Your External Memory Device.....	2-4
Selecting Your FPGA.....	2-4
Planning Your Pin Requirements.....	2-5
Planning Your FPGA Resources.....	2-6
Determining Your Board Layout.....	2-6
Specifying Parameters for Your External Memory Interface.....	2-6
Performing Functional Simulation.....	2-7
Adding Design Constraints.....	2-7
Compiling Your Design and Verifying Timing.....	2-8
Verifying and Debugging External Memory Interface Operation.....	2-8
Document Revision History.....	2-9
<b>Selecting Your Memory.....</b>	<b>3-1</b>
DDR SDRAM Features.....	3-2
DDR2 SDRAM Features.....	3-3
DDR3 SDRAM Features.....	3-3
QDR, QDR II, and QDR II+ SRAM Features.....	3-4
RLDRAM II and RLDARAM 3 Features.....	3-4
LPDDR2 Features.....	3-6
LPDDR3 Features.....	3-6
Memory Selection.....	3-6
Example of High-Speed Memory in Embedded Processor.....	3-9
Example of High-Speed Memory in Telecom.....	3-10
Document Revision History.....	3-12
<b>Selecting Your FPGA Device.....</b>	<b>4-1</b>
Memory Standards.....	4-1
I/O Interfaces.....	4-2
Wraparound Interfaces.....	4-2
Read and Write Leveling.....	4-2
Dynamic OCT.....	4-2
Device Settings Selection.....	4-3

Device Speed Grade.....	4-3
Device Operating Temperature.....	4-3
Device Package Size.....	4-3
Device Density and I/O Pin Counts.....	4-3
Document Revision History.....	4-5

# Introduction to Altera Memory Solution

1

2016.05.02

EMI\_GS



Subscribe



Send Feedback

The following topics provide an overview of Altera's External Memory Interface solutions.

Altera provides the fastest, most efficient, and lowest latency memory interface IP cores. Altera's external memory interface IP is designed to easily interface with today's higher speed memory devices.

Altera supports a wide variety of memory interfaces suitable for applications ranging from routers and switches to video cameras. You can easily implement Altera's intellectual property (IP) using the memory MegaCore functions through the Quartus Prime software. The Quartus Prime software also provides external memory toolkits that help you test the implementation of the IP in the FPGA device.

Refer to the External Memory Interface Spec Estimator page for the maximum speeds supported by Altera FPGAs.

## Related Information

- [External Memory Interface Spec Estimator](#)

- [Introduction to Altera IP Cores](#)

Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.

- [Creating Version-Independent IP and Qsys Simulation Scripts](#)

Create simulation scripts that do not require manual updates for software or IP version upgrades.

- [Project Management Best Practices](#)

Guidelines for efficient management and portability of your project and IP files.

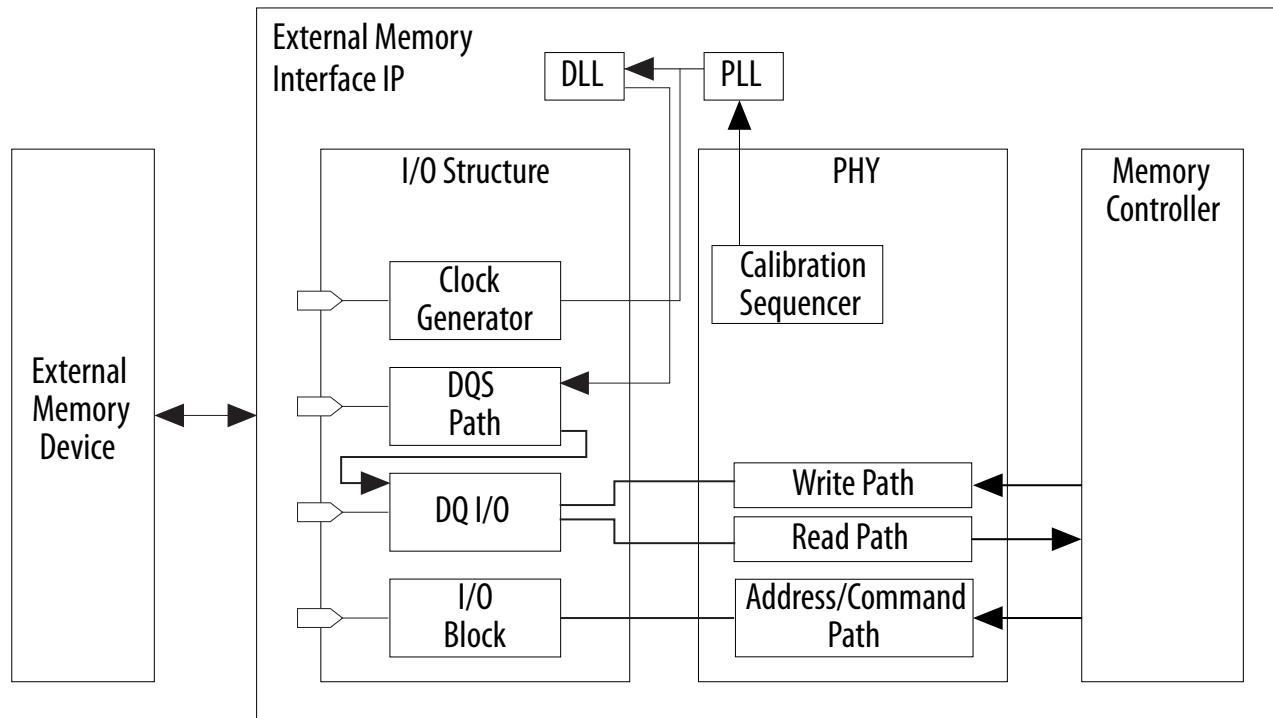
## Memory Solutions

Altera FPGAs achieve optimal memory interface performance with external memory IP. The IP provides the following components:

- Physical layer interface (PHY) which builds the data path and manages timing transfers between the FPGA and the memory device.
- Memory controller which implements all the memory commands and protocol-level requirements.
- Multi-port front end (MPFE) which allows multiple components inside the FPGA device to share a common memory interface. The MPFE is available in Arria V and Cyclone V devices.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

**Figure 1-1: Memory Interface Architecture**

Altera's FPGAs provide two types of memory solutions, depending on device family: soft memory IP and hard memory IP. The soft memory IP gives you the flexibility to design your own interfaces to meet your system requirements and still benefit from the industry leading performance. The hard memory IP is designed to give you a complete out-of-the-box experience when designing a memory controller.

The following table lists features of the soft and hard memory IP.

**Table 1-1: Features of the Soft and Hard Memory IP**

Soft Memory IP	Hard Memory IP
<ul style="list-style-type: none"> <li>Includes hardened PHY with soft controller.</li> <li>Allows maximum flexibility in choosing location, size, and configuration of the memory interface.</li> <li>Can optionally be used in PHY-only mode to integrate with a custom user-designed controller.</li> </ul>	<ul style="list-style-type: none"> <li>Includes hardened PHY, hardened controller, and hardened MPFE.</li> <li>Supports maximum performance and lowest latency.</li> <li>May have a fixed location on a device and/or a fixed pinout for address and command signals.</li> <li>Simplifies the overall integration of a memory interface and provides an out-of-the-box experience for every designer.</li> </ul>

Altera provides modular memory solutions that allow you to customize your memory interface design to a variety of configurations:

- PHY with your own controller
- PHY with Altera controller
- PHY with Altera controller and a multiport front end. (MPFE is a configurable block available for hard interfaces in Arria V and Cyclone V devices.)

You can also build a custom PHY, a custom controller, or both, as desired.

#### Related Information

- [Volume 3: Reference Material](#)
- [ALTDLL and ALTDQ\\_DQS Megafunctions User Guide](#)
- [ALTDQ\\_DQS2 Megafunction User Guide](#)
- [Altera PHYLite for Memory Megafunction User Guide](#)
- [Functional Description: Arria 10 EMIF IP](#)
- [Functional Description: MAX 10 EMIF IP](#)
- [External Memory Interface Spec Estimator](#)
- [Introduction to Altera IP Cores](#)  
Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- [Creating Version-Independent IP and Qsys Simulation Scripts](#)  
Create simulation scripts that do not require manual updates for software or IP version upgrades.
- [Project Management Best Practices](#)  
Guidelines for efficient management and portability of your project and IP files.

## Protocol Support Matrix

The following table lists the device family and IP architecture support for each memory protocol in the current release of the Altera Complete Design Suite.

Figure 1-2: Protocol Support Matrix (1) (2) (3) (4) (5)

Protocol	MAX 10	Arria 10	Stratix V / Arria V GZ	Arria V GX, GT, SX, ST	Cyclone V	Stratix IV	Stratix III	Arria II GZ	Arria II GX	Clock Rate	Hard / Soft PHY	Burst length	Sequencer	Controller
DDR4	—	•	—	—	—	—	—	—	—	Quarter	Hard	8	Hard Nios	Hard
DDR3	—	•	—	—	—	—	—	—	—	Quarter	Hard	8	Hard Nios	Hard
	—	•	—	—	—	—	—	—	—	Half	Hard	8	Hard Nios	Hard
	—	—	—	U	U	—	—	—	—	Full	Hard	8	Nios II	HPC II
	U	—	U	U	U	U	U	U	A	Half	Soft	8	Nios II	HPC II
	—	—	U	U	—	—	—	—	—	Quarter	Soft	8	Nios II	HPC II
DDR2	—	—	—	U	U	—	—	—	—	Full	Hard	4,8	Nios II	HPC II
	—	—	U	—	—	U	U	U	A	Full	Soft	4,8	Nios II	HPC II
	U	—	U	U	U	U	U	U	A	Half	Soft	4,8	Nios II	HPC II
LPDDR3	—	•	—	—	—	—	—	—	—	Quarter / Half	Hard	8	Hard Nios	Hard
LPDDR2	U	—	—	U	U	—	—	—	—	Half	Soft	4,8,16	Nios II	HPC II
	—	—	—	—	U	—	—	—	—	Full	Hard	4,8,16	Nios II	HPC II
RLDRAM 3	—	—	U	—	—	—	—	—	—	Half	Soft	2,4,8	Nios II	—
	—	—	U	—	—	—	—	—	—	Quarter	Soft	2,4,8	Nios II	—
	—	•	—	—	—	—	—	—	—	Quarter	Hard	2,4,8	Hard Nios	3rd Party
RLDRAM II	—	—	U	—	—	U	U	U	—	Full	Soft	2,4,8	RTL	RLDRAM II
	—	—	U	U	—	U	U	U	—	Half	Soft	4,8	Nios II	RLDRAM II
	—	—	U	—	—	U	U	U	—	Half	Soft	4,8	RTL	RLDRAM II
QDR II/II+	—	—	U	—	—	U	U	U	U	Full	Soft	2,4	RTL	QDR II/II+
	—	—	U	U	—	U	U	U	—	Half	Soft	4	Nios II	QDR II/II+
	—	—	U	—	—	U	U	U	U	Half	Soft	4	RTL	QDR II/II+
	—	•	—	—	—	—	—	—	—	Full	Hard	2,4	Hard Nios	QDR II/II+
	—	•	—	—	—	—	—	—	—	Half	Hard	4	Hard Nios	QDR II/II+
QDR II+ Xtreme	—	•	—	—	—	—	—	—	—	Half	Hard	4	Hard Nios	QDR II/II+
QDR IV	—	•	—	—	—	—	—	—	—	Quarter	Hard	2	Hard Nios	QDR IV

Notes to Table:

1. U = Supported by UniPHY-based IP.
2. A = Supported by ALTMEMPHY-based IP. Refer to the *External Memory Interface Handbook* for the Quartus II software version 12.1 or earlier for information about ALTMEMPHY-based IP.
3. — = Not supported.
4. . = Supported in Arria 10 device.
5. The RTL-based sequencer is not available for QDR II or RLDRAM II interfaces targeting Arria V devices.

For more information about the controllers with the UniPHY IP, refer to the *Functional Descriptions* section in Volume 3 of the *External Memory Interface Handbook*.

For more information on the Arria 10 External Memory Interface IP, see *Functional Description—Arria 10 EMIF IP*.

For more information on the MAX 10 External Memory Interface IP, see *Functional Description—MAX 10 EMIF IP*.

For more information on the Arria 10 PHYLite IP, see the *PHYLite IP Megafunction User Guide*.

#### Related Information

- [External Memory Interface Spec Estimator](#)
- [Introduction to Altera IP Cores](#)  
Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.
- [Creating Version-Independent IP and Qsys Simulation Scripts](#)  
Create simulation scripts that do not require manual updates for software or IP version upgrades.
- [Project Management Best Practices](#)  
Guidelines for efficient management and portability of your project and IP files.

## Arria 10 EMIF Future Protocol Support

The following table lists planned future memory protocol support for Arria 10 EMIF IP.

Protocol	Current Support	Future Support
DDR4	<ul style="list-style-type: none"><li>• Hard PHY and Hard Controller</li><li>• Hard PHY only</li></ul>	Yes (LRDIMM, RDIMM, x4 DQ/DQS)
DDR3	<ul style="list-style-type: none"><li>• Hard PHY and Hard Controller</li><li>• Hard PHY only</li></ul>	Yes (LRDIMM, RDIMM, x4 DQ/DQS)
DDR2	No	Yes (via Altera PHYLite for Memory)
LPDDR3	Yes	Yes
LPDDR2	No	Yes (via Altera PHYLite for Memory)
QDR II / II+ / QDR II+ Xtreme	Hard PHY and Soft Controller	Yes
RLLDRAM 3	Hard PHY only	Yes
RLLDRAM II	No	Yes (via Altera PHYLite for Memory)

#### Related Information

- [External Memory Interface Spec Estimator](#)
- [Introduction to Altera IP Cores](#)  
Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.

- Creating Version-Independent IP and Qsys Simulation Scripts**

Create simulation scripts that do not require manual updates for software or IP version upgrades.

- Project Management Best Practices**

Guidelines for efficient management and portability of your project and IP files.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	<ul style="list-style-type: none"> <li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> <li>Changed Arria 10 EMIF current support for LPDDR3 to yes.</li> <li>Added LPDDR3 to product support matrix.</li> </ul>
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	<ul style="list-style-type: none"> <li>Added QDR IV and MAX 10 support to the <i>Protocol Support Matrix</i>.</li> </ul>
August 2014	2014.08.15	<ul style="list-style-type: none"> <li>Added information for Quartus II software versions 14.0 and 14.0 Arria 10 Edition to <i>Altera Memory Types, PHY, and Controllers in the Quartus II Software</i> table.</li> <li>Added QDR II, QDR II+, and QDR II+ Xtreme support for Arria 10 to the <i>Protocol Support Matrix</i>.</li> <li>Updated DDR3, DDR4, QDR II+ / QDR II+ Extreme, and RLDRAM 3 support in the <i>Arria 10 EMIF Future Protocol Support</i> table.</li> </ul>
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>Added Arria 10 and DDR4 information to <i>Protocol Support Matrix</i> and <i>Memory Solutions</i>.</li> <li>Combined <i>Soft and Hard Memory IP</i> and <i>Memory Solutions</i> sections.</li> <li>Removed HardCopy III/IV from <i>Protocol Support Matrix</i>.</li> <li>Added note to <i>Protocol Support Matrix</i> that RTL-based sequencer is not available for QDR II or RLDRAM II interfaces targeting Arria V devices</li> </ul>
November 2012	2.0	<ul style="list-style-type: none"> <li>Added Arria V GZ information.</li> <li>Added RLDRAM III information to <i>Protocol Support Matrix</i> and <i>Memory Solutions</i>.</li> </ul>
June 2012	1.2	Change to Table 1-3.
June 2012	1.1	<ul style="list-style-type: none"> <li>Added <i>Protocol Support Matrix</i>.</li> <li>Added Feedback icon.</li> </ul>
November 2011	1.0	Initial release.

## Related Information

- [External Memory Interface Spec Estimator](#)

- [Introduction to Altera IP Cores](#)

Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.

- [Creating Version-Independent IP and Qsys Simulation Scripts](#)

Create simulation scripts that do not require manual updates for software or IP version upgrades.

- [Project Management Best Practices](#)

Guidelines for efficient management and portability of your project and IP files.

# Recommended Design Flow

2

2016.05.02

EMI\_GS



Subscribe



Send Feedback

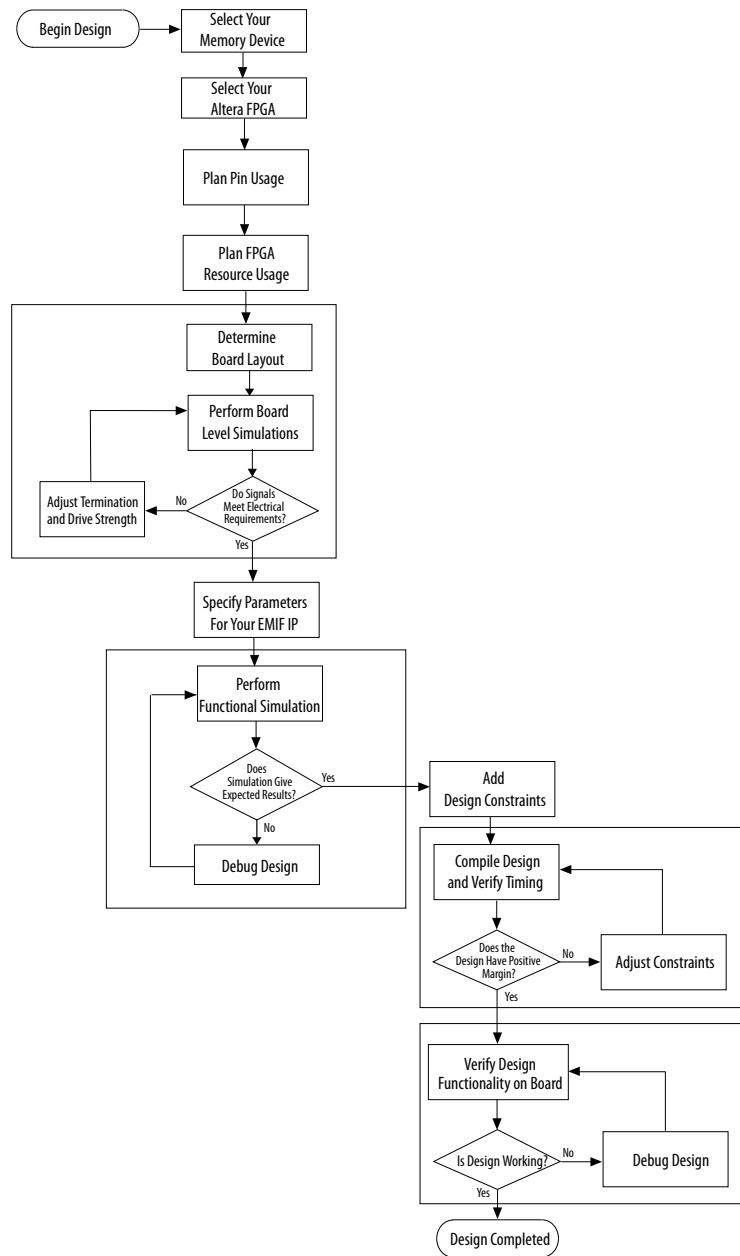
Altera recommends that you create an example top-level file with the desired pin outs and all interface IP instantiated, which enables the Quartus® Prime software to validate your design and resource allocation before PCB and schematic sign off.

The following figure shows the design flow to provide the fastest out-of-the-box experience with external memory interfaces in Altera devices. This design flow assumes that you are using Altera IP to implement the external memory interface.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

Figure 2-1: External Memory Interfaces Design Flowchart



Refer to *Getting Started with External Memory Interfaces* for guidance in performing the recommended steps in creating a working and robust external memory interface.

#### Related Information

- [Getting Started With External Memory Interfaces](#) on page 2-3
- [Introduction to Altera IP Cores](#)

Provides general information about all Altera IP cores, including parameterizing, generating, upgrading, and simulating IP.

- **Creating Version-Independent IP and Qsys Simulation Scripts**

Create simulation scripts that do not require manual updates for software or IP version upgrades.

- **Project Management Best Practices**

Guidelines for efficient management and portability of your project and IP files.

## Getting Started With External Memory Interfaces

To create your external memory interface, you must complete several high-level tasks. This topic outlines the major tasks in the design flow, and provides links to detailed procedures for each task.

Refer to this section for a big-picture view of the overall design process, and for links to related information for each task.

### The High-Level Tasks

1. **Selecting Your External Memory Device** on page 2-4

Different memory types excel in different areas. As a first step in planning your external memory interface, you must determine the memory type that best meets the requirements of your system.

2. **Selecting Your FPGA** on page 2-4

Different Altera FPGA devices support different memory types; not all Altera devices support all memory protocols and configurations. Before you start your design, you must select an Altera device, which supports the memory standard and configurations you plan to use.

3. **Planning Your Pin Requirements** on page 2-5

Before you can specify parameters for your external memory interface, you must determine the pin requirements.

4. **Planning Your FPGA Resources** on page 2-6

Before you can specify parameters for your external memory interface, you must determine the FPGA resource requirements.

5. **Determining Your Board Layout** on page 2-6

Before you can specify parameters for your external memory interface, you must determine the necessary board-related settings for your IP.

6. **Specifying Parameters for Your External Memory Interface** on page 2-6

After you have determined all the necessary requirements, you can parameterize your external memory interface.

7. **Performing Functional Simulation** on page 2-7

Simulate your design to determine correct operation, timing closure, and overall latency.

8. **Adding Design Constraints** on page 2-7

Design constraints establish the timing characteristics of your IP and the physical locations of I/O and routing resources.

9. **Compiling Your Design and Verifying Timing** on page 2-8

When you compile your design, the TimeQuest Timing Analyzer generates timing reports for your design.

10. **Verifying and Debugging External Memory Interface Operation** on page 2-8

Operational problems can generally be attributed to one of the following: resource and planning problems, interface configuration problems, functional problems, signal integrity problems, or timing problems.

## Selecting Your External Memory Device

Different memory types excel in different areas. As a first step in planning your external memory interface, you must determine the memory type that best meets the requirements of your system.

1. Determine your requirements for the following:

- bandwidth
- speed
- data capacity
- latency
- power consumption

2. Compare your requirements to the specifications for available memory protocols to find the memory device appropriate for your application.

### Related Information

#### [Selecting Your Memory](#)

## Selecting Your FPGA

Different Altera FPGA devices support different memory types; not all Altera devices support all memory protocols and configurations. Before you start your design, you must select an Altera device, which supports the memory standard and configurations you plan to use.

1. Determine the I/O interface that best suits your design requirements.
2. Determine whether your design requires read or write levelling circuitry.

Some Altera FPGAs support read and write levelling, to apply or remove skew from an interface on a DQS group basis.

3. Determine whether your design requires dynamic calibrated on-chip termination (OCT).

Some Altera FPGAs provide dynamic OCT, allowing a specified series termination to be enabled during writes and parallel termination to be enabled during reads. Dynamic OCT can simplify your PCB termination schemes.

4. Consult the Altera Product Selector to find the Altera FPGA that provides the combination of features that your design requires.
5. Refer to the Ordering Information section of the appropriate device handbook, to determine the correct ordering code for the device that you require. Consider the following characteristics in determining the correct ordering code:

- Speed grade: Affects performance, timing closure, and power consumption. The device with the smallest speed grade number is the fastest device.
- Operating temperature: Altera FPGAs are divided into the following temperature categories:
  - Commercial grade—Used for all device families. Operating temperature ranges from 0 degrees C to 85 degrees C.
  - Industrial grade—Used for all device families. Operating temperature ranges from -40 degrees C to 100 degrees C.
  - Military grade—Used for Stratix IV device families. Operating temperature ranges from -55 degrees C to 125 degrees C.
  - Automotive grade—Used for Cyclone V device families. Operating temperature ranges from -40 degrees C to 125 degrees C.
- Package size: Refers to the physical size of the FPGA device, and corresponds to the number of pins. For example, the package size for the smallest Stratix IV device is 29 mm x 29 mm, categorized under the F780 package option, where F780 refers to a device with 780 pins.
- Device density: Refers to the number of logic elements, such as PLLs and memory blocks. Devices with higher density contain more logic elements in less area.
- I/O pin counts: The number of I/O pins required on an FPGA depends on the memory standard, the number of memory interfaces, and the memory data width.

**Tip:** For additional, device-specific, information, refer to the External Memory Interface chapter in the device handbook for your Altera device.

#### Related Information

- [Selecting Your FPGA Device](#)
- [Altera Product Selector](#)
- [Altera Device Handbooks](#)

## Planning Your Pin Requirements

Before you can specify parameters for your external memory interface, you must determine the pin requirements. You should use the Quartus Prime software for final pin fitting; however, you can estimate whether you have enough pins for your memory interface.

1. Determine how many read data pins are associated per read data strobe or clock pair.
2. Check the device density and packaging information for your FPGA to determine whether you can implement your interface in one I/O bank, or on one side of the device, or on two adjacent sides.
3. Calculate the number of other memory interface pins needed, including any other clocks (write clock or memory system clock), address, command, RUP, RDN, RZQ, and any other pins to be connected to the memory components. Ensure you have enough pins to implement the interface in one I/O bank or one side or on two adjacent sides.
4. Apply the General Pin-Out Guidelines, and observe any device- or protocol-specific guidelines or exceptions applicable to your design situation.

#### Related Information

- [Planning Pin and FPGA Resources](#)
- [External Memory Interface Spec Estimator](#)

## Planning Your FPGA Resources

Before you can specify parameters for your external memory interface, you must determine the FPGA resource requirements. The FPGA resources required by your design depend on many factors, including the memory interface frequency, timing requirements, and the IP that your design uses.

1. Determine the PLLs and clock networks that your design requires.
2. If multiple PLLs are required for multiple controllers that cannot be shared, ensure that enough PLL resources are available within each quadrant to support your interface number requirements.
3. Determine whether cascading of PLLs is appropriate for your design.
4. Determine the appropriate DLL usage for your design. If multiple external memory interfaces must share DLL resources, ensure that the frequency and mode requirements are compatible.
5. Determine the registers, memory blocks, OCT blocks, and other FPGA resources required by your design.

### Related Information

- [Planning Pin and FPGA Resources](#)
- [External Memory Interface Spec Estimator](#)

## Determining Your Board Layout

Before you can specify parameters for your external memory interface, you must determine the necessary board-related settings for your IP.

1. Review the recommended board design guidelines for your external memory interface protocol.
2. Select the termination scheme and drive strength settings for all the memory interface signals connected between the FPGA and the external memory device.
3. Perform board-level simulations to determine the optimal settings for best signal integrity, appropriate timing margins, and sufficient eye opening.
  - Successful board-level simulation is often an iterative process, experimenting with different combinations of drive strength, terminations, IP board parameters, and timing results.
  - Ensure that your simulation applies the latest FPGA and memory device IBIS models, board trace characteristics, drive strength, and termination settings.
  - You might identify board-level timing uncertainties such as crosstalk, ISI, or slew rate deration during simulation. If you identify such timing uncertainties, adjust the Board Settings in the IP Catalog with the slew rate deration, ISI/crosstalk, and board skews to ensure the accuracy of the TimeQuest timing margins report.

### Related Information

- [DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines](#)
- [Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines](#)
- [LPDDR2 SDRAM Board Design Guidelines](#)
- [QDR II and QDR IV SRAM Board Design Guidelines](#)
- [RLDRAM II and RLDRAM 3 Board Design Guidelines](#)

## Specifying Parameters for Your External Memory Interface

After you have determined all the necessary requirements, you can parameterize your external memory interface.

1. In the parameter editor, set the parameters for the external memory IP for your target memory interface.
  - Refer to *Specifying IP Core Parameters and Options* for information about using the IP Catalog and parameter editor.
  - Refer to *Implementing and Parameterizing Memory IP* for detailed information about parameterizing external memory interface IP.
2. Specify the correct parameters for each of the following:
  - Memory interface data rate, width, and configuration.
  - Necessary deratings for tIS, tIH, tDH, and tDS parameters, as appropriate.
  - Board skew parameters based on actual board simulation.
3. Connect the local signals from the PHY and controller to your driver logic, and the memory interface signals from the PHY to the top-level pins.
  - It is important that you connect the local interface signals from the PHY or controller correctly to your own logic. If you do not connect these local interface signals, you might encounter problems with insufficient pins when you compile your design.
  - Logic that is not connected may be optimized away during compilation, resulting in problems later.
  - If you want to use your own custom memory controller with the Altera PHY, you can refer to the example top-level file as an example for connecting your controller.

#### Related Information

- [Implementing and Parameterizing Memory IP](#)
- [Functional Description—HPC II Controller](#)
- [Functional Description—Hard Memory Interface](#)
- [Functional Description—QDR II Controller](#)
- [Functional Description—QDR IV Controller](#)
- [Functional Description—RLDRAM II Controller](#)
- [Functional Description—RLDRAM 3 PHY-Only IP](#)
- [Functional Description—Arria 10 EMIF](#)
- [Functional Description—MAX 10 EMIF](#)

## Performing Functional Simulation

Simulate your design to determine correct operation, timing closure, and overall latency.

1. Simulate your design using the RTL functional model.
2. Use the IP functional simulation model with your own driver logic, testbench, and a memory model, to ensure correct read and write transactions to the memory.
3. You may need to prepare the memory functional model by setting the speed grade and device bus mode.

#### Related Information

- [Simulating Memory IP](#)

## Adding Design Constraints

Design constraints establish the timing characteristics of your IP and the physical locations of I/O and routing resources.

1. Add timing constraints.
2. Add pin assignments.
3. Add pin location assignments.
4. Ensure that the example top-level file or your top-level logic is set as top-level entity.
5. Adjust optimization techniques, to ensure the remaining unconstrained paths are routed with the highest speed and efficiency, as follows:
  - a. In the Quartus Prime software, click **Assignments > Settings**.
  - b. In the **Settings** dialog box, select the **Compiler Settings** category.
  - c. In the **Compiler Settings** dialog box, click **Advanced Settings (Synthesis)** and set the **Optimization Technique** value to **Speed**.
  - d. In the **Compiler Settings** dialog box, click **Advanced Settings (Fitter)** and set **Optimize hold timing to All Paths**. Turn on **Optimize multi-corner timing**. Set **Fitter Effort** to **Standard Fit**.

#### Related Information

##### [Analyzing Timing of Memory IP](#)

## Compiling Your Design and Verifying Timing

When you compile your design, the TimeQuest Timing Analyzer generates timing reports for your design.

1. Compile your design by clicking **Processing > Start Compilation**.  
Memory timing scripts run automatically as part of **Report DDR**.
2. Verify timing closure using all available models, and evaluate the timing reports generated by the TimeQuest Timing Analyzer.  
As required, adjust the constraints described in [Adding Design Constraints](#) to resolve timing or location issues.
3. Iteratively recompile your IP and evaluate the timing results as necessary to achieve the required timing margins.

#### Related Information

- [Analyzing Timing of Memory IP](#)
- [Implementing and Parameterizing Memory IP](#)

## Verifying and Debugging External Memory Interface Operation

Operational problems can generally be attributed to one of the following: resource and planning problems, interface configuration problems, functional problems, signal integrity problems, or timing problems.

- Refer to *Debugging Memory IP* and the *External Memory Interface Debug Toolkit* for information on resolving operational problems.

#### Related Information

- [Debugging Memory IP](#)
- [External Memory Interface Debug Toolkit](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	<ul style="list-style-type: none"><li>Revised the <i>External Memory Interfaces Design Flowchart</i>.</li><li>Removed the <i>Design Checklist</i> and added <i>Getting Started With External Memory Interfaces</i>, and associated subtopics.</li></ul>
August 2014	2014.08.15	Removed MegaWizard Plug-In Manager flow and added IP Catalog Flow to <i>External Memory Interfaces Design Flowchart</i> .
December 2013	2013.12.16	<ul style="list-style-type: none"><li>Removed references to ALTMEMPHY.</li><li>Removed references to SOPC Builder.</li><li>Removed ALTMEMPHY-related step from design checklist.</li></ul>
June 2012	2013.12.02	<ul style="list-style-type: none"><li>Removed overlapping information.</li><li>Added Feedback icon.</li></ul>
November 2011	2.1	Updated the design flow and the design checklist.
July 2010	2.0	Updated for 10.0 release.
January 2010	1.1	<ul style="list-style-type: none"><li>Improved description for Implementing Altera Memory Interface IP chapter.</li><li>Added timing simulation to flow chart and to design checklist.</li></ul>
November 2009	1.0	Initial release.

# Selecting Your Memory

3

2016.05.02

EMI\_GS



Subscribe



Send Feedback

System architects must consider architecture, algorithms, and features of the available components.

Typically, one of the fundamental problems in high-performance applications is memory, because the challenges and limitations of system performance often reside in memory architecture. As higher speeds become necessary for external memories, signal integrity becomes more challenging; newer devices include several features to address this challenge. Altera FPGAs include dedicated I/O circuitry, various I/O standard support, and specialized intellectual property (IP).

When you select an external memory device, consider the following factors:

- Bandwidth and speed
- Cost
- Data storage capacity
- Latency
- Power consumption

Because no single memory type can excel in every area, system architects must determine the right balance for their design. The following table lists the two common types of high-speed memories and their characteristics.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

**Table 3-1: Differences between DRAM and SRAM**

Memory Type	Description	Bandwidth and Speed	Cost	Data Storage Size and Capacity	Power consumption	Latency
DRAM	<p>A dynamic random access memory (DRAM) cell consisting of a capacitor and a single transistor. DRAM memory must be refreshed periodically to retain the data, resulting in lower overall efficiency and more complex controllers.</p> <p>Generally, designers select DRAM where cost per bit and capacity are important. DRAM is commonly used for main memory.</p>	Lower bandwidth resulting in slower speed	Lower cost	Higher data storage and capacity	Higher power consumption	Higher latency
SRAM	<p>A static random access memory (SRAM) cell that consists of six transistors. SRAM does not need to be refreshed because the transistors continue to hold the data as long as the power supply is not cut off.</p> <p>Generally, designers select SRAM where speed is more important than capacity. SRAM is commonly used for cache memory.</p>	Higher bandwidth resulting in faster speed	Higher cost	Lower data storage and capacity	Lower power consumption	Lower latency

**Note:** The Altera IP might or might not support all of the features supported by the memory.

To compare the performance of the supported external memory interfaces in Altera FPGA devices, refer to the *External Memory Interface Spec Estimator* page on the Altera website.

#### Related Information

[External Memory Interface Spec Estimator](#)

## DDR SDRAM Features

Double data rate (DDR) SDRAM is a 2n prefetch architecture with two data transfers per clock cycle. It uses a single-ended strobe, DQS, which is associated with a group of data pins, DQ, for read and write operations. Both DQS and DQ are bidirectional ports. Address ports are shared for read and write operations.

The desktop computing market has positioned DDR SDRAM as a mainstream commodity product, which means this memory is very low-cost. DDR SDRAM is also high-density and low-power. Relative to other high-speed memories, DDR SDRAM has higher latency—they have a multiplexed address bus, which reduces the pin count (minimizing cost) at the expense of a longer and more complex bus cycle.

## DDR2 SDRAM Features

DDR2 SDRAM is a 4n prefetch architecture (internally the memory operates at half the interface frequency) with two data transfers per clock cycle. DDR2 SDRAM can use a single-ended or differential strobe, DQS or DQSn, which is associated with a group of data pins, and DQ for read and write operations. The DQS, DQSn, and DQ are bidirectional ports. Address ports are shared for read and write operations.

DDR2 SDRAM includes additional features such as increased bandwidth due to higher clock speeds, improved signal integrity on DIMMs with on-die terminations, and lower supply voltages to reduce power.

## DDR3 SDRAM Features

DDR3 SDRAM is the third generation of SDRAM. DDR3 SDRAM is internally configured as an eight-bank DRAM and uses an 8n prefetch architecture to achieve high-speed operation. The 8n prefetch architecture is combined with an interface that transfers two data words per clock cycle at the I/O pins. A single read or write operation for DDR3 SDRAM consists of a single 8n-bit wide, one-clock-cycle data transfer at the internal DRAM core and eight corresponding n-bit wide, one-half clock cycle data transfers at the I/O pins. DDR3 SDRAMs are available as components and modules, such as DIMMs, SODIMMs, RDIMMs, and LRDIMMs.

DDR3 SDRAM can conserve system power, increase system performance, achieve better maximum throughput, and improve signal integrity with fly-by topology and dynamic on-die termination.

Read and write operations to the DDR3 SDRAM are burst oriented. Operation begins with the registration of an active command, which is followed by a read or write command. The address bits registered coincident with the active command select the bank and row to be activated (BA0 to BA2 select the bank; A0 to A15 select the row). The address bits registered coincident with the read or write command select the starting column location for the burst operation, determine if the auto precharge command is to be issued (via A10), and select burst chop (BC) of 4 or burst length (BL) of 8 mode at runtime (via A12), if enabled in the mode register. Before normal operation, the DDR3 SDRAM must be powered up and initialized in a predefined manner.

Differential strobes DQS and DQSn are mandated for DDR3 SDRAM and are associated with a group of data pins, as is DQ for read and write operations. DQS, DQSn, and DQ ports are bidirectional. Address ports are shared for read and write operations.

**Note:** The DDR3 SDRAM high-performance controller II supports local interfaces running at full-rate, half-rate, and quarter-rate.

For more information, refer to the respective DDR, DDR2, and DDR3 SDRAM data sheets.

For more information about parameterizing the DDR2 and DDR3 SDRAM IP, refer to the *Implementing and Parameterizing Memory IP* chapter.

### Related Information

#### [Implementing and Parameterizing Memory IP](#)

## QDR, QDR II, and QDR II+ SRAM Features

Quad Data Rate (QDR) SRAM has independent read and write ports that run concurrently at double data rate. QDR SRAM is true dual-port (although the address bus is still shared), which gives this memory a high bandwidth, allowing back-to-back transactions without the contention issues that can occur when using a single bidirectional data bus. Write and read operations share address ports.

The QDR II SRAM devices are available in  $\times 8$ ,  $\times 9$ ,  $\times 18$ , and  $\times 36$  data bus width configurations. The QDR II+ SRAM devices are available in  $\times 9$ ,  $\times 18$ , and  $\times 36$  data bus width configurations. Write and read operations are burst-oriented. All the data bus width configurations of QDR II SRAM support burst lengths of two and four. QDR II+ SRAM supports only a burst length of four. Burst-of-two and burst-of-four for QDR II and burst-of-four for QDR II+ SRAM devices provide the same overall bandwidth at a given clock speed.

For QDR II SRAM devices, the read latency is 1.5 clock cycles; for QDR II+ SRAM devices, it is 2 or 2.5 clock cycles depending on the memory device. For QDR II+ and burst-of-four QDR II SRAM devices, the write commands and addresses are clocked on the rising edge of the clock, and write latency is one clock cycle. For burst-of-two QDR II SRAM devices, the write command is clocked on the rising edge of the clock, and the write address is clocked on the falling edge of the clock. Therefore, the write latency is zero because the write data is presented at the same time as the write command.

QDR II+ and QDR II SRAM interfaces use a delay-locked loop (DLL) inside the device to edge-align the data with respect to the  $\kappa$  and  $\kappa\#$  or  $c$  and  $c\#$  pins. You can optionally turn off the DLL, but the performance of the QDR II+ and QDR II SRAM devices is degraded. All timing specifications listed in this document assume that the DLL is on. QDR II+ and QDR II SRAM devices also offer programmable impedance output buffers. You can set the buffers by terminating the  $z_Q$  pin to VSS through a resistor,  $R_Q$ . The value of  $R_Q$  should be five times the desired output impedance. The range for  $R_Q$  should be between 175 ohm and 350 ohm with a tolerance of 10%.

QDR II+ SRAM is best suited for applications where the required read/write ratio is near one-to-one. QDR II+ SRAM includes additional features such as increased bandwidth due to higher clock speeds, lower voltages to reduce power, and on-die termination to improve signal integrity. QDR II+ SDRAM is the latest and fastest generation. For QDR II+ and QDR II SRAM interfaces, Altera supports both 1.5-V and 1.8-V HSTL I/O standards.

For more information, refer to the respective QDRII and QDRII+ data sheets.

For more information about parameterizing the QDRII and QDRII+ SRAM IP, refer to the *Implementing and Parameterizing Memory IP* chapter.

### Related Information

[Implementing and Parameterizing Memory IP](#)

## RLDRAM II and RLDRAM 3 Features

Reduced latency DRAM (RLDRAM) provides DRAM-based point-to-point memory devices designed for communications, imaging, server systems, networking, and cache applications requiring high density, high memory bandwidth, and low latency. The fast random access speeds in RLDRAM devices make them a viable alternative to SRAM devices at a lower cost.

The high performance of RLDRAM is achieved by very low random access delay (tRC), low data-bus-turnaround delay, simple command protocol, and a large number of banks. RLDRAM is optimized to meet the needs of high-bandwidth networking applications.

Contrasting with the typical four banks in most memory devices, RLDRAM II is partitioned into eight banks and RLDRAM 3 is partitioned into sixteen banks. Partitioning reduces the parasitic capacitance of the address and data lines, allowing faster accesses and reducing the probability of random access conflicts. Each bank has a fixed number of rows and columns. Only one row per bank is accessed at a time. The memory (instead of the controller) controls the opening and closing of a row, which is similar to an SRAM interface.

Most DRAM memory types need both a row and column phase on a multiplexed address bus to support full random access, while RLDRAM supports a nonmultiplexed address, saving bus cycles at the expense of more pins. RLDRAM II and RLDRAM 3 use the High-Speed Transceiver Logic (HSTL) standard with double data rate (DDR) data transfer to provide a very high throughput.

There are two types of RLDRAM II or RLDRAM 3 devices—common I/O (CIO) and separate I/O (SIO). CIO devices share a single data I/O bus, which is similar to the double data rate (DDR) SDRAM interface. SIO devices, with separate data read and write buses, have an interface similar to SRAM. Altera UniPHY Memory IP only supports CIO RLDRAM.

RLDRAM II and RLDRAM 3 use a DDR scheme, performing two data transfers per clock cycle. RLDRAM II or RLDRAM 3 CIO devices use the bidirectional data pins ( $D_Q$ ) for both read and write data, while RLDRAM II or RLDRAM 3 SIO devices use  $D$  pins for write data (input to the memory) and  $Q$  pins for read data (output from the memory). Both types use two pairs of unidirectional free-running clocks. The memory uses  $DK$  and  $DK\#$  pins during write operations, and generates  $QK$  and  $QK\#$  pins during read operations. In addition, RLDRAM II and RLDRAM 3 use the system clocks ( $CK$  and  $CK\#$  pins) to sample commands and addresses, and to generate the  $QK$  and  $QK\#$  read clocks. Address ports are shared for write and read operations.

RLDRAM II CIO devices are available in  $\times 9$ ,  $\times 18$ ,  $\times 36$  data bus width configurations. RLDRAM II CIO interfaces may require an extra cycle for bus turnaround time for switching read and write operations. RLDRAM 3 devices are available in  $\times 18$  and  $\times 36$  data bus width configurations.

Write and read operations are burst oriented, and all the data bus width configurations of RLDRAM II and RLDRAM 3 support burst lengths of two and four. RLDRAM 3 also supports burst length of eight at bus width  $\times 18$ , and burst lengths of two and four at bus width  $\times 36$ . For detailed comparisons between RLDRAM II and RLDRAM 3 for these features, refer to the *Memory Selection Overview* table.

RLDRAM II and RLDRAM 3 also inherently include the additional memory bits used for parity or error correction code (ECC).

RLDRAM II and RLDRAM 3 also offer programmable impedance output buffers and on-die termination. The programmable impedance output buffers are for impedance matching and are guaranteed to produce 25- to 60-ohm output impedance. The on-die termination is dynamically switched on during read operations and switched off during write operations. Perform an IBIS simulation to observe the effects of this dynamic termination on your system. IBIS simulation can also show the effects of different drive strengths, termination resistors, and capacitive loads on your system.

RLDRAM 3 enables a faster, more efficient transfer of data by doubling performance and reduced latency compared to RLDRAM II. RLDRAM 3 memory is suitable for operation in which high bandwidth and deterministic performance is critical, and is optimized to meet the needs of high-bandwidth networking applications. For detailed comparisons between RLDRAM II and RLDRAM 3, refer to the following table.

For more information, refer to RLDRAM II and RLDRAM 3 data sheets available from the Micron website ([www.micron.com](http://www.micron.com)).

For more information about parameterizing the RLDRAM II and RLDRAM 3 IP, refer to the *Implementing and Parameterizing Memory IP* chapter.

#### Related Information

- [Implementing and Parameterizing Memory IP](#)

- [www.micron.com](http://www.micron.com)

## LPDDR2 Features

LPDDR2-S is a high-speed SDRAM device internally configured as a 4- or 8-bank memory. All LPDDR2 devices use double data rate architecture on the address and command bus to reduce the number of input pins in the system. The 10-bit address and command bus contains command, address, and bank/row buffer information. Each command uses one clock cycle, during which command information is transferred on both the positive and negative edges of the clock.

LPDDR2-S2 and LPDDR2-S4 devices use double data rate architecture on the DQ pins to achieve high speed operation. The double data rate architecture is essentially a 2n/4n prefetch architecture with an interface designed to transfer two data bits per DQ every clock cycle at the I/O pins. A single read or write access for the LPDDR2-S2/S4 consists of a single 2n-bit wide /4n-bit wide, one clock cycle data transfer at the internal SDRAM core, and two/four corresponding n-bit wide, with one-half clock cycle data transfers at the I/O pins.

## LPDDR3 Features

LPDDR3-SDRAM is a high-speed synchronous DRAM device internally configured as an 8-bank memory. All LPDDR3 devices use double data rate architecture on the address and command bus to reduce the number of input pins in the system. The 10-bit address and command bus contains command, address, and bank buffer information. Each command uses one clock cycle, during which command information is transferred on both the positive and negative edges of the clock.

LPDDR3 devices use double data rate architecture on the DQ pins to achieve high speed operation. The double data rate architecture is an interface that transfers two data bits per DQ every clock cycle at the I/O pins.

Read and write operations to the LPDDR3 SDRAMs are burst oriented. Operations begin at a selected location and continue for a programmed number of locations in a programmed sequence. The operations begin with the registration of an activate command, which is then followed by a read or write command. Use the address and BA bits registered and the activate command to select the row and the bank to be accessed. Use the address bits registered and the read or write command to select the bank and the starting column location for the burst access

For more information, refer to LPDDR3 SDRAM data sheets.

### Related Information

#### [Implementing and Parameterizing Memory IP](#)

## Memory Selection

One of the first considerations in choosing a high-speed memory is data bandwidth. Based on the system requirements, an approximate data rate to the external memory should be determined. You must also consider other memory attributes, including how much memory is required (density), how much latency can be tolerated, what is the power budget, and whether the system is cost sensitive.

The following table lists memory features and target markets of each technology.

**Table 3-2: Memory Selection Overview**

Parameter	LPDDR2	DDR3 SDRAM	DDR2 SDRAM	DDR SDRAM	RDRAM II	RDRAM 3	QDR II/+ SRAM
Bandwidth for 32 bit interface (1)	25.6	59.7	25.6	12.8	25.6	35.8	44.8
Bandwidth at % Efficiency (Gbps) (2)	17.9	23.9	17.9	9	17.9	—	38.1
Performance / Clock frequency	167–400 MHz <sup>(3)</sup>	300–933 MHz	167–400 MHz <sup>(3)</sup>	100–200 MHz	200–533 MHz	200–800 MHz	154–350 MHz
Altera-supported data rate	Up to 1,066 Mbps	Up to 2,133 Mbps	Up to 1,066 Mbps	Up to 400 Mbps	Up to 1066 Mbps	Up to 1600 Mbps	Up to 1400 Mbps
Density	64 MB –8 GB	512 MB–8 GB, 32 MB –8 GB (DIMM)	256 MB–1 GB, 32 MB –4 GB (DIMM)	128 MB–1 GB, 32 MB –2 GB (DIMM)	288 MB, 576 MB	576 MB –1.1 GB	18–144 MB
I/O standard	HSUL- 12.1.2V	SSTL-15 Class I, II	SSTL-18 Class I, II	SSTL-2 Class I, II	HSTL-1.8V/ 1.5V	HSTL-1.2V and SSTL-12	HSTL-1.8V/ 1.5V
Data group width	8, 16, 32	4, 8, 16	4, 8, 16	4, 8, 16, 32	9, 18, 36	18, 36	9, 18, 36
Burst length	4, 8, 16	8	4, 8	2, 4, 8	2, 4, 8	2, 4, 8	2, 4
Number of banks	4, 8	8	8 (>1 GB), 4	4	8	16	—

Parameter	LPDDR2	DDR3 SDRAM	DDR2 SDRAM	DDR SDRAM	RLDRAM II	RLDRAM 3	QDR II/+ SRAM
Row/column access	Row before column	Row before column	Row before column	Row before column	Row and column together or multiplexed option	Row and column together or multiplexed option	—
CAS latency (CL)	—	5, 6, 7, 8, 9, 10	3, 4, 5	2, 2.5, 3	—	—	—
Posted CAS additive latency (AL)	—	0, CL-1, CL-2	0, 1, 2, 3, 4	—	—	—	—
Read latency (RL)	3, 4, 5, 6, 7, 8	RL = CL + AL	RL = CL + AL	RL = CL	3, 4, 5, 6, 7, 8	3-16	1.5, 2, and 2.5 clock cycles
On-die termination	—	Yes	Yes	No	Yes	Yes	Yes
Data strobe	Differential bidirectional	Differential bidirectional strobe only	Differential or single-ended bidirectional strobe	Single-ended bidirectional strobe	Free-running differential read and write clocks	Free-running differential read and write clocks	Free-running read and write clocks
Refresh requirement	Yes	Yes	Yes	Yes	Yes	Yes	No
Relative cost comparison	Higher than DDR SDRAM	Presently lower than DDR2	Less than DDR SDRAM with market acceptance	Low	Higher than DDR SDRAM, less than SRAM	Higher than DDR SDRAM, less than SRAM	Highest

Parameter	LPDDR2	DDR3 SDRAM	DDR2 SDRAM	DDR SDRAM	RDRAM II	RDRAM 3	QDR II/+ SRAM
Target market	Mobile devices that target low operating power	Desktops, servers, storage, LCDs, displays, networking, and communication equipment	Desktops, servers, storage, LCDs, displays, networking, and communication equipment	Desktops, servers, storage, LCDs, displays, networking, and communication equipment	Main memory, cache memory, networking, packet processing, and traffic management	Main memory, cache memory, networking, packet processing, and traffic management	Cache memory, routers, ATM switches, packet memories, lookup, and classification memories

Notes to Table:

1. 32-bit data bus operating at the maximum supported frequency in a Stratix® V FPGA.
2. 70% efficiency for DDR memories, which takes into consideration the bus turnaround, refresh, infinite burst length and random access latency and assumes 85% efficiency for QDR memories.
3. The lower frequency limit depends on the higher of the DLL frequency and the minimum operating frequency of the given EMIF protocol. (Except for DDR2 interfaces running on Stratix V devices.)

Altera supports the memory interfaces, provides various IP for the physical interface and the controller, and offers many reference designs (refer to Altera's *Memory Solutions Center*).

For Altera support and the maximum performance for the various high-speed memory interfaces, refer to the *External Memory Interface Spec Estimator* page on the Altera website.

#### Related Information

- [Memory Solutions Center](#)
- [External Memory Interface Spec Estimator](#)

## Example of High-Speed Memory in Embedded Processor

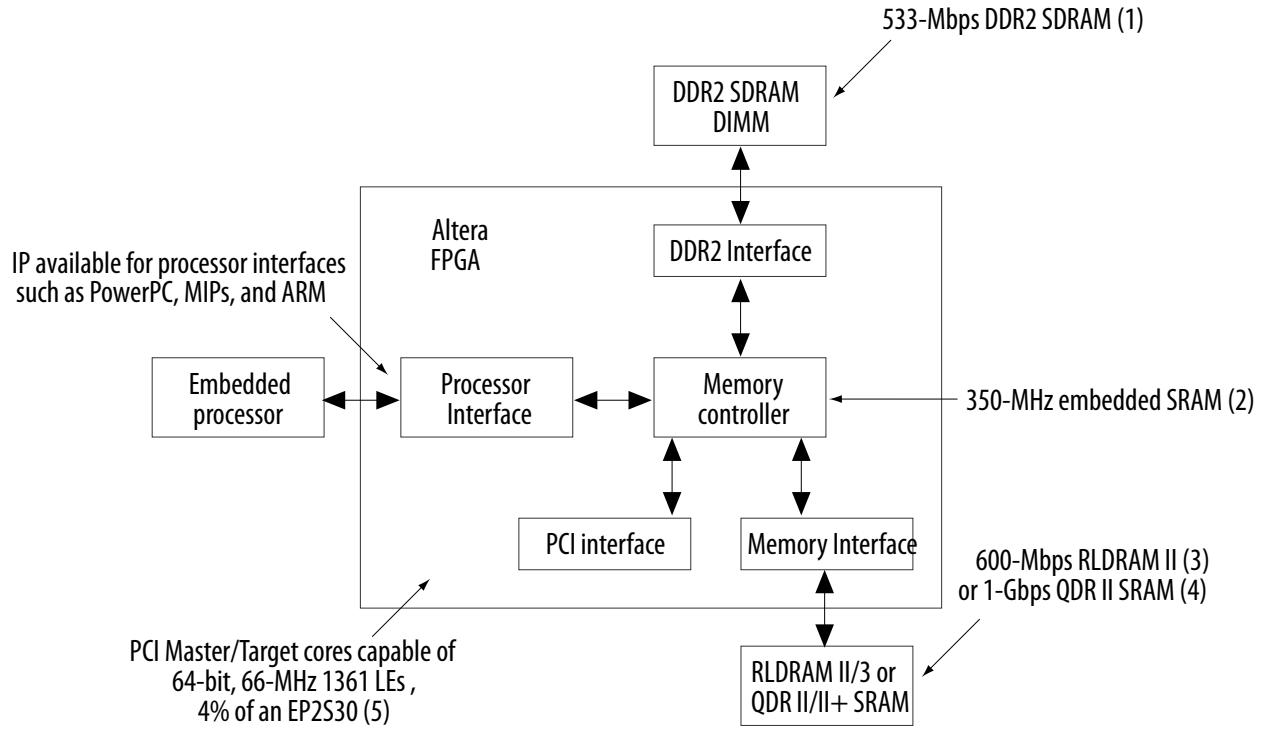
In embedded processor applications—any system that uses processors, excluding desktop processors—due to its very low cost, high density, and low power, DDR SDRAM is typically used for main memory.

Next-generation processors invest a large amount of die area to on-chip cache memory to prevent the execution pipelines from sitting idle. Unfortunately, these on-chip caches are limited in size, as a balance of performance, cost, and power must be taken into consideration. In many systems, external memories are used to add another level of cache. In high-performance systems, three levels of cache memory is common: level one (8 Kbytes is common) and level two (512 Kbytes) on chip, and level three off chip (2 Mbytes).

High-end servers, routers, and even video game systems are examples of high-performance embedded products that require memory architectures that are both high speed and low latency. Advanced memory controllers are required to manage transactions between embedded processors and their memories. Altera Arria® series and Stratix series FPGAs optimally implement advanced memory controllers by utilizing

their built-in DQS (strobe) phase shift circuitry. The following figure highlights some of the features available in an Altera FPGA in an embedded application, where DDR2 SDRAM is used as the main memory and QDR II/II+ SRAM or RLDRAM II/3 is an external cache level.

**Figure 3-1: Memory Controller Example Using FPGA**



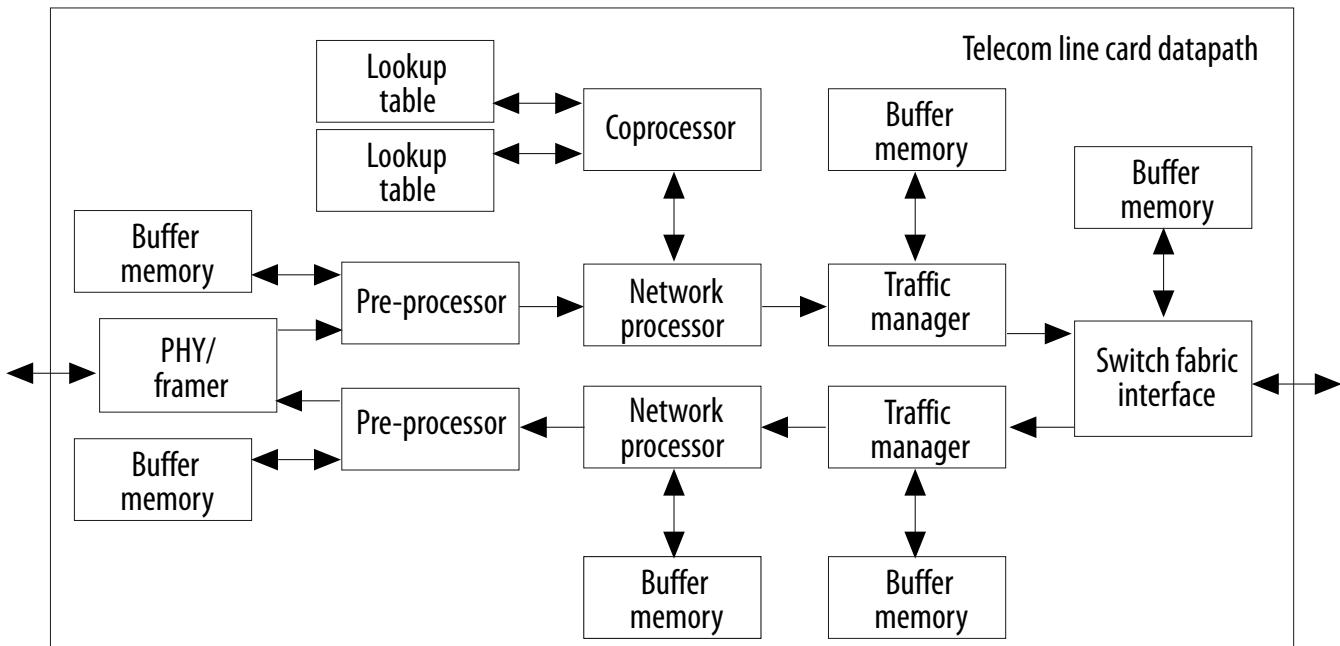
One of the target markets of RLDRAM II/3 and QDR/QDR II SRAM is external cache memory. RLDRAM II and RLDRAM 3 have a read latency close to synchronous SRAM, but with the density of SDRAM. A sixteen times increase in external cache density is achievable with one RLDRAM II/3 versus that of synchronous static RAM (SSRAM). In contrast, consider QDR and QDR II SRAM for systems that require high bandwidth and minimal latency. Architecturally, the dual-port nature of QDR and QDR II SRAM allows cache controllers to handle read data and instruction fetches completely independent of writes.

## Example of High-Speed Memory in Telecom

Because telecommunication network architectures are becoming more complex, high-end network systems are running multiple 10-Gbps line cards that connect to multi-shelf switch fabrics scaling to terabits per second.

The following figure shows an example of a typical system line interface card. These line cards offer interfaces ranging from a single-port OC-192 to multi-port Gbps Ethernet, and consist of a number of devices, including a PHY/framer, network processors, traffic managers, fabric interface devices, and high-speed memories.

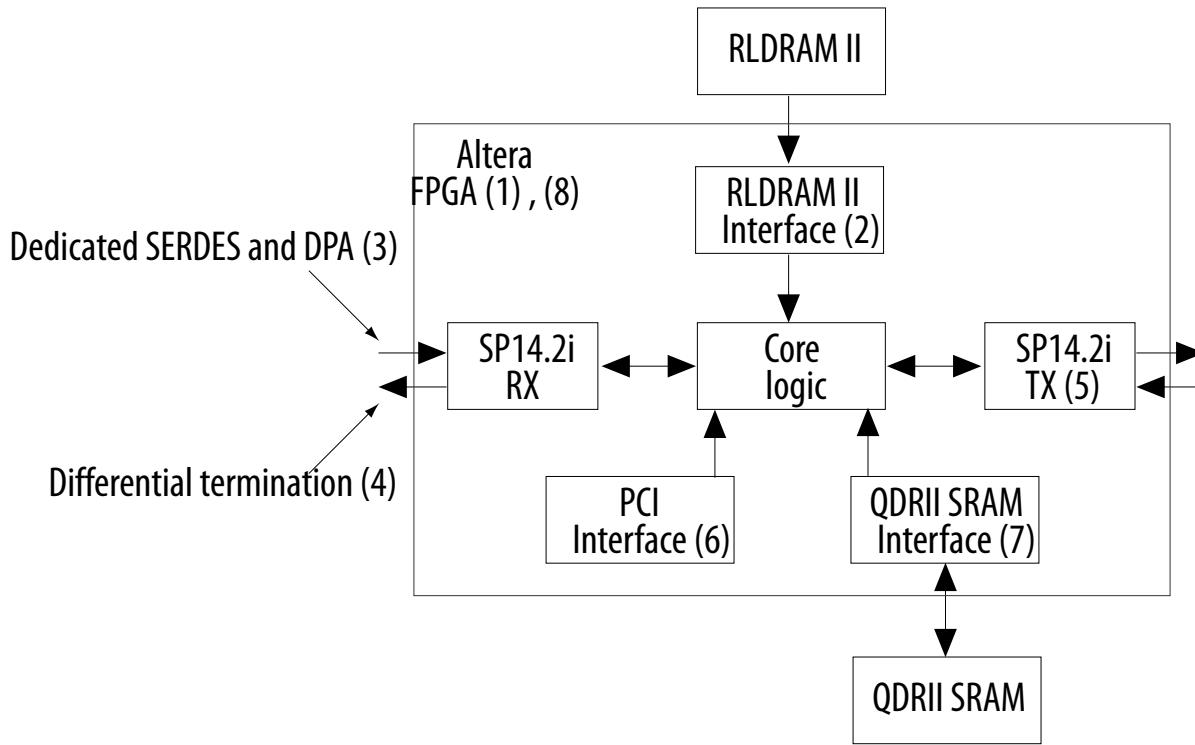
Figure 3-2: Typical Telecom System Line Interface Card



As packets traverse from the PHY/framer device to the switch fabric interface, they are buffered into memories, while the data path devices process headers (determining the destination, classifying packets, and storing statistics for billing) and control the flow of packets into the network to avoid congestion. Typically DDR/DDR2/DDR3 SDRAM and RLDRAM II/3 are used for large buffer memories off network processors, traffic managers, and fabric interfaces, while QDR and QDR II/II+ SRAMs are used for look-up tables (LUTs) off preprocessors and coprocessors.

In many designs, FPGAs connect devices together for interoperability and coprocessing, implement features that are not supported by ASIC devices, or implement a device function entirely. Altera Stratix series FPGAs implement traffic management, packet processing, switch fabric interfaces, and coprocessor functions, using features such as 1-Gbps LVDS I/O, high-speed memory interface support, multi-gigabit transceivers, and IP cores. The following figure highlights some of these features in a packet buffering application where RLDRAM II is used for packet buffer memory and QDR II SRAM is used for control memory.

Figure 3-3: FPGA Example in Packet Buffering Application



SDRAM is usually the best choice for buffering at high data rates due to the large amounts of memory required. Some system designers take a hybrid approach to the memory architecture, using SRAM to store the packet headers and DRAM to store the payload. The depth of the memories depends on the architecture and throughput of the system.

The buffer memory for the packet buffering application of an OC-192 line card (approximately 10 Gbps) must be able to sustain a minimum of one write and one read operation, which requires a memory bandwidth of 20 Gbps to operate at full line rate (more bandwidth is required if the headers are modified). The bandwidth requirement for memory is a key factor in memory selection. As an example, a simple first-order calculation using RLDRAM II as buffer memory requires a bus width of 48 bits to sustain 20 Gbps ( $300 \text{ MHz} \times 2 \text{ DDR} \times 0.70 \text{ efficiency} \times 48 \text{ bits} = 20.1 \text{ Gbps}$ ), which needs two RLDRAM II parts (one  $\times 18$  and one  $\times 36$ ). RLDRAM II and RLDRAM 3 also inherently include the additional memory bits used for parity or error correction code (ECC). QDR and QDR II SRAM have bandwidth and low random access latency advantages that make them useful for control memory in queue management and traffic management applications. Another typical implementation for this memory is billing and packet statistics, where each packet requires counters to be read from memory, incremented, and then rewritten to memory. The high bandwidth, low latency, and optimal one-to-one read/write ratio make QDR SRAM ideal for this feature.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Moved chapter from Volume 2 to Volume 1.
November 2015	2015.11.01	Added LPDDR3 features.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Modified note 3 on <i>Memory Selection Overview</i> table.
August 2014	2014.08.15	<ul style="list-style-type: none"><li>Changed some values in the <i>Bandwidth for 32 bits</i>, <i>Bandwidth at % Efficiency</i>, <i>Performance / Clock frequency</i>, and <i>Altera-supported data rate</i> rows of the <i>Memory Selection Overview</i> table.</li></ul>
December 2013	2013.12.16	Removed references to Stratix II devices.
November 2012	6.0	Added RLDRAM 3 support.
June 2012	5.0	<ul style="list-style-type: none"><li>Added LPDDR2 support.</li><li>Added Feedback icon.</li></ul>
November 2011	4.0	Moved and reorganized “Selecting your Memory” section to Volume 2: Design Guidelines.
June 2011	3.0	Added “Selecting Memory IP” chapter from Volume 2.
December 2010	2.1	<ul style="list-style-type: none"><li>Moved protocol-specific feature information to the memory interface user guides in Volume 3.</li><li>Updated maximum clock rate information for 10.1.</li></ul>
July 2010	2.0	<ul style="list-style-type: none"><li>Added specifications for DDR2 and DDR3 SDRAM Controllers with UniPHY.</li><li>Streamlined the specification tables.</li><li>Added reference to web-based Specification Estimator Tool.</li></ul>
January 2010	1.1	Updated DDR, DDR2, and DDR3 specifications.
November 2009	1.0	First published.

# Selecting Your FPGA Device

4

2016.05.02

EMI\_GS



Subscribe



Send Feedback

Altera external memory interfaces support three FPGA device families—Arria®<sup>®</sup>, Stratix®<sup>®</sup>, and Cyclone®<sup>®</sup> device families. These FPGA device families vary in terms of cost, memory standards, speed grade, and features.

**Note:** Use the Altera Product Selector to find and compare specifications and features of Altera devices.

Consult these topics together with the *Planning Pin and FPGA Resources* chapter, before you start implementing your external memory interface.

The following topics describe the factors that you should consider when selecting an FPGA device family.

## Related Information

- [Altera Product Selector](#)
- [Planning Pin and FPGA Resources](#)
- [Planning Pin and FPGA Resources](#)

## Memory Standards

Altera devices support two common types of high-speed memories—dynamic random access memory (DRAM) and static random access memory (SRAM). The commonly used DRAM devices include DDR, DDR2, DDR3, and DDR4 SDRAM, LPDDR2, LPDDR3, RLDRAM II, and RLDRAM 3, while SRAM devices include QDR II, QDR II+, and QDR II+ Xtreme SRAM.

For more information about these memory types, refer to the *Selecting Your Memory* chapter.

Different Altera FPGA devices support different memory types; not all Altera devices support all memory types and configurations. Before you start your design, you must select an Altera device, which supports the memory standard and configurations you plan to use.

In addition, Altera's FPGA devices support various data widths for different memory interfaces. The memory interface support between density and package combinations differs, so you must determine which FPGA device density and package combination suits your application.

For more information about the supported memory types and configurations, refer to the *External Memory Interface Spec Estimator* page on the Altera website.

## Related Information

- [Selecting Your Memory](#)
- [External Memory Interface Spec Estimator](#)

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

## I/O Interfaces

Ideally any interface should reside entirely in a single bank; however, interfaces that span across multiple adjacent banks or the entire side of a device are also fully supported.

Interfaces that span across sides (top and bottom, or left and right) and wraparound interfaces provide the same level of performance.

For information about the I/O interfaces supported for each device, and the locations of those I/O interfaces, refer to the *I/O Features* section in the appropriate device handbook.

## Wraparound Interfaces

For maximum performance, Altera recommends that data groups for external memory interfaces should always be within the same side of a device, ideally reside within a single bank.

High-speed memory interfaces using top or bottom I/O bank versus left or right I/O bank have different timing characteristics, so the timing margins are also different. However, Altera can support interfaces with wraparound data groups that wrap around a corner of the device between vertical and horizontal I/O banks at some speeds. Some devices support wraparound interfaces that run at the same speed as row or column interfaces.

Arria II GX devices can support wraparound interface across all sides of devices that are not used for transceivers. Other UniPHY-supported Altera devices support only interfaces with data groups that wrap around a corner of the device.

## Read and Write Leveling

The Arria V GZ, Arria 10, Stratix III, Stratix IV, and Stratix V I/O registers include read and write leveling circuitry to enable skew to be removed or applied to the interface on a DQS group basis. There is one leveling circuit located in each I/O subbank.

**Note:** UniPHY-based designs do not require read leveling circuitry during read leveling operation.

For more information about read and write leveling, refer to *Leveling Circuitry* section in the *Functional Description - UniPHY* chapter of the *External Memory Interface Handbook*.

### Related Information

[Functional Description - UniPHY](#)

## Dynamic OCT

The Arria II GZ, Arria V, Arria 10, Cyclone V, Stratix III, Stratix IV, and Stratix V devices support dynamic calibrated OCT.

Dynamic calibrated OCT allows the specified series termination to be enabled during writes, and parallel termination to be enabled during reads. These I/O features allow you to simplify PCB termination schemes.

## Device Settings Selection

After you have selected the appropriate FPGA device family for your memory interface, configure the device settings of your selected FPGA device family to meet your design needs.

Refer to the device ordering code and determine the appropriate device settings for your target device family.

For more information about the ordering code for your target device, refer to the “Ordering Information” section in volume 1 of the respective device handbooks.

The following sections describe the ordering code and how to select the appropriate device settings based on the ordering code to meet the requirements of your external memory interface.

### Device Speed Grade

The device speed grade affects the device timing performance, timing closure, and power utilization.

The device with the smallest number is the fastest device and vice-versa. Generally, the faster devices cost more.

### Device Operating Temperature

The operating temperature of the FPGA is divided into the following categories:

- Commercial grade—Used for all device families. The operating temperature range from 0 degrees C to 85 degrees C.
- Industrial grade—Used for all device families. The operating temperature range from -40 degrees C to 100 degrees C.
- Military grade—Used for Stratix IV device family only. The operating temperature range from -55 degrees C to 125 degrees C.
- Automotive grade—Used for Cyclone V device families only. The operating temperature range from -40 degrees C to 125 degrees C.

### Device Package Size

Each FPGA family has a range of package sizes.

Package size refers to the actual size of an FPGA device and corresponds to the number of pin counts. For example, the package size for the smallest FPGA device in the Stratix IV family is 29 mm x 29 mm, categorized under the F780 package option, where F780 refers to a device with 780 pin counts.

For more information about the package size available for your device, refer to the respective device handbooks.

### Device Density and I/O Pin Counts

An FPGA device of the same device family and package size also varies in terms of device density and I/O pin counts.

For example, after you have selected the Stratix IV device family with the F780 packaging option, you must determine the type of device models that ranges from EP4GX70 to EP4GX230. Each of these devices has similar speed grades that range from grade 2 to grade 4, but are different in density.

## Device Density

Device density refers to the number of logic elements (LEs). For example, PLLs, memory blocks, and so on. An FPGA device with higher density contains more logic elements in less area.

## I/O Pin Counts

To meet the growing demand for memory bandwidth and memory data rates, memory interface systems use parallel memory channels and multiple controller interfaces. However, the number of memory channels is limited by the package pin count of the Altera devices. Therefore, you must consider device pin count when you select a device; you must select a device with enough I/O pins for your memory interface requirement.

The number of device pins required depends on the memory standard, the number of memory interfaces, and the memory data width. For example, a  $\times 72$  DDR3 SDRAM single-rank interface requires 125 I/O pins:

- 72 DQ pins (including ECC)
- 9 DM pins
- 9 DQS, DQSn differential pin pairs
- 17 address pins (address and bank address)
- 7 command pins (CAS, RAS, WE, CKE, ODT, reset, and CS)
- 1 CK, CK# differential pin pair

**Note:** For more information about the number of embedded memory, PLLs and user I/O counts that are available for your device, refer to the respective device handbooks. For the number of DQS groups available for each FPGA device, refer to the respective device handbooks.

**Note:** For the maximum number of controllers that is supported by the FPGAs for different memory types, refer to the *Planning Pin and FPGA Resources* chapter.

Altera devices do not limit the interface widths beyond the following requirements:

- The DQS, DQ, clock, and address signals of the entire interface must reside within the same bank or side of the device if possible, to achieve better performance. Although wraparound interfaces are also supported at limited frequencies.
- The maximum possible interface width in any particular device is limited by the number of DQS and DQ groups available within that bank or side.
- Sufficient regional clock networks are available to the interface PLL to allow implementation within the required number of quadrants.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other clock, address, and command pin placement requirements.
- The greater the number of banks, the greater the skew. Altera recommends that you always compile a test project of your desired configuration and confirm that it meets timing requirement.

Your pin count calculation also determines which device side to use (top or bottom, left or right, and wraparound).

**Note:** When assigning DQS and DQ pins in Arria® II GX devices, you are allowed to use only twelve of the sixteen I/O pins in an I/O module as DQ pins. The remaining four pins can be used only as input pins.

**Note:** For DQS groups pin-out restriction format, refer to *Arria II GX Pin Connection Guidelines*.

**Note:** The Arria II GX and Stratix V devices do not support interfaces on the left side of the device. There are no user I/O pins, other than the transceiver pins available in these devices.

**Related Information**

- [Planning Pin and FPGA Resources](#)
- [Arria II GX Pin Connection Guidelines](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Moved chapter from Volume 2 to Volume 1.
November 2015	2015.11.02	Added LPDDR3 to memory standards.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Maintenance release.
December 2013	2013.12.16	Removed references to Cyclone III and Cyclone IV devices.
June 2012	5.0	<ul style="list-style-type: none"><li>• Added LPDDR2 support.</li><li>• Added Feedback icon.</li></ul>
November 2011	4.0	Moved and reorganized “Selecting your FPGA” section to Volume 2: Design Guidelines.
June 2011	3.0	Added “Selecting a Device” chapter from Volume 2.
December 2010	2.1	<ul style="list-style-type: none"><li>• Moved protocol-specific feature information to the memory interface user guides in Volume 3.</li><li>• Updated maximum clock rate information for 10.1.</li></ul>
July 2010	2.0	<ul style="list-style-type: none"><li>• Added specifications for DDR2 and DDR3 SDRAM Controllers with UniPHY.</li><li>• Streamlined the specification tables.</li><li>• Added reference to web-based Specification Estimator Tool.</li></ul>
January 2010	1.1	Updated DDR, DDR2, and DDR3 specifications.

Date	Version	Changes
November 2009	1.0	First published.

# External Memory Interface Handbook

## Volume 2: Design Guidelines

 [Subscribe](#)  
 [Send Feedback](#)

Last updated for Quartus Prime Design Suite: 16.0

**EMI\_DG**  
2016.05.02

101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)

**ALTERA**  
now part of Intel

# Contents

<b>Planning Pin and FPGA Resources.....</b>	<b>1-1</b>
Interface Pins.....	1-1
Estimating Pin Requirements.....	1-4
DDR, DDR2, DDR3, and DDR4 SDRAM Clock Signals.....	1-5
DDR, DDR2, DDR3, and DDR4 SDRAM Command and Address Signals.....	1-5
DDR, DDR2, DDR3, and DDR4 SDRAM Data, Data Strobes, DM/DBI, and Optional ECC Signals.....	1-6
DDR, DDR2, DDR3, and DDR4 SDRAM DIMM Options.....	1-8
QDR II, QDR II+, and QDR II+ Xtreme SRAM Clock Signals.....	1-11
QDR II, QDR II+ and QDR II+ Xtreme SRAM Command Signals.....	1-12
QDR II, QDR II+ and QDR II+ Xtreme SRAM Address Signals.....	1-12
QDR II, QDR II+ and QDR II+ Xtreme SRAM Data, BWS, and QVLD Signals.....	1-12
QDR IV SRAM Clock Signals.....	1-13
QDR IV SRAM Commands and Addresses, AP, and AINV Signals.....	1-14
QDR IV SRAM Data, DINV, and QVLD Signals.....	1-14
RLDRAM II and RLDRAM 3 Clock Signals.....	1-15
RLDRAM II and RLDRAM 3 Commands and Addresses.....	1-16
RLDRAM II and RLDRAM 3 Data, DM and QVLD Signals.....	1-16
LPDDR2 and LPDDR3 Clock Signal.....	1-18
LPDDR2 and LPDDR3 Command and Address Signal.....	1-18
LPDDR2 and LPDDR3 Data, Data Strobe, and DM Signals.....	1-18
Maximum Number of Interfaces.....	1-18
OCT Support .....	1-30
Guidelines for Arria 10 External Memory Interface IP.....	1-31
General Pin-Out Guidelines for Arria 10 EMIF IP.....	1-32
Resource Sharing Guidelines for Arria 10 EMIF IP.....	1-36
Guidelines for UniPHY-based External Memory Interface IP.....	1-37
General Pin-out Guidelines for UniPHY-based External Memory Interface IP.....	1-37
Pin-out Rule Exceptions for ×36 Emulated QDR II and QDR II+ SRAM Interfaces in Arria II, Stratix III and Stratix IV Devices.....	1-39
Pin-out Rule Exceptions for RLDRAM II and RLDRAM 3 Interfaces.....	1-44
Pin-out Rule Exceptions for QDR II and QDR II+ SRAM Burst-length-of-two Interfaces.....	1-46
Pin Connection Guidelines Tables.....	1-46
PLLs and Clock Networks.....	1-60
Using PLL Guidelines.....	1-66
PLL Cascading.....	1-67
DLL.....	1-67
Other FPGA Resources.....	1-68
Document Revision History.....	1-69
<b>DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines.....</b>	<b>2-1</b>

Leveling and Dynamic Termination.....	2-2
Read and Write Leveling.....	2-2
Dynamic ODT.....	2-4
Dynamic On-Chip Termination.....	2-4
Dynamic On-Chip Termination in Stratix III and Stratix IV Devices.....	2-5
Dynamic OCT in Stratix V Devices.....	2-6
Dynamic On-Chip Termination (OCT) in Arria 10 Devices.....	2-7
Termination for DDR2 SDRAM.....	2-8
External Parallel Termination.....	2-8
On-Chip Termination.....	2-10
Recommended Termination Schemes.....	2-11
Termination for DDR3 SDRAM.....	2-15
Terminations for Single-Rank DDR3 SDRAM Unbuffered DIMM.....	2-16
Terminations for Multi-Rank DDR3 SDRAM Unbuffered DIMM.....	2-17
Terminations for DDR3 SDRAM Registered DIMM.....	2-19
Terminations for DDR3 SDRAM Load-Reduced DIMM.....	2-19
Terminations for DDR3 SDRAM Components With Leveling.....	2-20
DDR3 and DDR4 on Arria 10 Devices.....	2-20
Dynamic On-Chip Termination (OCT) in Arria 10 Devices.....	2-20
Dynamic On-Die Termination (ODT) in DDR4.....	2-22
Choosing Terminations on Arria 10 Devices.....	2-22
On-Chip Termination Recommendations for DDR3 and DDR4 on Arria 10 Devices.....	2-22
Layout Approach.....	2-23
Design Layout Guidelines.....	2-24
General Layout Guidelines.....	2-24
Layout Guidelines for DDR2 SDRAM Interface.....	2-26
Layout Guidelines for DDR3 and DDR4 SDRAM Interfaces.....	2-30
Length Matching Rules.....	2-34
Spacing Guidelines.....	2-35
Layout Guidelines for DDR3 SDRAM Wide Interface (>72 bits).....	2-36
Package Deskew.....	2-39
Package Deskew Recommendation for Stratix V Devices.....	2-39
DQ/DQS/DM Deskew.....	2-40
Address and Command Deskew.....	2-40
Package Deskew Recommendations for Arria 10 Devices.....	2-40
Deskew Example.....	2-41
Package Migration.....	2-42
Package Deskew for RLDRAM II and RLDRAM 3.....	2-42
Document Revision History.....	2-43

## **Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines..... 3-1**

General Layout Guidelines.....	3-1
Dual-Slot Unbuffered DDR2 SDRAM.....	3-2
Overview of ODT Control.....	3-3
DIMM Configuration.....	3-5
Dual-DIMM Memory Interface with Slot 1 Populated.....	3-5
Dual-DIMM with Slot 2 Populated.....	3-6
Dual-DIMM Memory Interface with Both Slot 1 and Slot 2 Populated.....	3-8

Dual-DIMM DDR2 Clock, Address, and Command Termination and Topology.....	3-11
Control Group Signals.....	3-12
Clock Group Signals.....	3-12
Dual-Slot Unbuffered DDR3 SDRAM.....	3-12
Comparison of DDR3 and DDR2 DQ and DQS ODT Features and Topology.....	3-13
Dual-DIMM DDR3 Clock, Address, and Command Termination and Topology.....	3-13
FPGA OCT Features.....	3-14
Document Revision History.....	3-15

## **LPDDR2 and LPDDR3 SDRAM Board Design Guidelines.....4-1**

LPDDR2 Guidance.....	4-1
LPDDR2 SDRAM Configurations.....	4-1
OCT Signal Terminations for Arria V and Cyclone V Devices.....	4-4
General Layout Guidelines.....	4-7
LPDDR2 Layout Guidelines.....	4-8
LPDDR2 SDRAM Layout Approach.....	4-9
LPDDR3 Guidance.....	4-10
Signal Integrity, Board Skew, and Board Setting Parameters.....	4-11
LPDDR3 Layout Guidelines.....	4-11
Package Deskew.....	4-12
Document Revision History.....	4-15

## **RLDRAM II and RLDRAm 3 Board Design Guidelines.....5-1**

RLDRAM II Configurations.....	5-2
RLDRAM 3 Configurations.....	5-3
Signal Terminations.....	5-5
Input to the FPGA from the RLDRAm Components.....	5-7
Outputs from the FPGA to the RLDRAm II and RLDRAm 3 Components.....	5-7
RLDRAM II Termination Schemes.....	5-8
RLDRAM 3 Termination Schemes.....	5-9
PCB Layout Guidelines.....	5-10
General Layout Guidelines.....	5-10
RLDRAM II and RLDRAm 3 Layout Approach.....	5-12
RLDRAM II and RLDRAm 3 Layout Guidelines.....	5-12
Layout Approach.....	5-13
Arria V and Stratix V Board Setting Parameters.....	5-15
Arria 10 Board Setting Parameters.....	5-15
Package Deskew for RLDRAm II and RLDRAm 3.....	5-15
Document Revision History.....	5-15

## **QDR II and QDR-IV SRAM Board Design Guidelines.....6-1**

QDR II SRAM Configurations.....	6-2
Signal Terminations.....	6-4
Output from the FPGA to the QDR II SRAM Component.....	6-5
Input to the FPGA from the QDR II SRAM Component.....	6-5
Termination Schemes.....	6-5

General Layout Guidelines.....	6-8
QDR II Layout Guidelines.....	6-9
QDR II SRAM Layout Approach.....	6-10
Package Deskew for QDR II and QDR-IV.....	6-12
QDR-IV Layout Recommendations.....	6-12
QDR-IV Layout Approach.....	6-12
QDR-IV Layout Guidelines.....	6-14
Document Revision History.....	6-15

## **Implementing and Parameterizing Memory IP.....7-1**

Licensing IP Cores.....	7-1
Design Flow.....	7-2
IP Catalog Design Flow.....	7-3
Qsys System Integration Tool Design Flow.....	7-7
UniPHY-Based External Memory Interface IP.....	7-9
Qsys Interfaces.....	7-9
Generated Files for Memory Controllers with the UniPHY IP.....	7-41
Parameterizing Memory Controllers.....	7-45
Board Settings .....	7-67
Controller Settings for UniPHY IP.....	7-82
Diagnostics for UniPHY IP.....	7-88
Arria 10 External Memory Interface IP.....	7-90
Qsys Interfaces.....	7-90
Generated Files for Arria 10 External Memory Interface IP.....	7-106
Parameterizing Arria 10 External Memory Interface IP.....	7-110
Document Revision History.....	7-138

## **Simulating Memory IP.....8-1**

Simulation Options.....	8-1
Simulation Walkthrough with UniPHY IP.....	8-3
Simulation Scripts.....	8-4
Preparing the Vendor Memory Model.....	8-4
Functional Simulation with Verilog HDL.....	8-7
Functional Simulation with VHDL.....	8-7
Simulating the Example Design.....	8-8
UniPHY Abstract PHY Simulation.....	8-9
PHY-Only Simulation.....	8-11
Post-fit Functional Simulation.....	8-12
Simulation Issues.....	8-13
Simulation Walkthrough with Arria 10 EMIF IP.....	8-15
Skip Calibration Versus Full Calibration.....	8-15
Arria 10 Abstract PHY Simulation.....	8-16
Simulation Scripts.....	8-17
Functional Simulation with Verilog HDL.....	8-17
Functional Simulation with VHDL.....	8-18
Simulating the Example Design.....	8-19
Document Revision History.....	8-20

<b>Analyzing Timing of Memory IP.....</b>	<b>9-1</b>
Memory Interface Timing Components.....	9-2
Source-Synchronous Paths.....	9-2
Calibrated Paths.....	9-2
Internal FPGA Timing Paths.....	9-3
Other FPGA Timing Parameters.....	9-3
FPGA Timing Paths.....	9-3
Arria II Device PHY Timing Paths.....	9-3
Stratix III and Stratix IV PHY Timing Paths.....	9-6
Arria V, Arria V GZ, Arria 10, Cyclone V, and Stratix V Timing paths.....	9-8
Timing Constraint and Report Files for UniPHY IP.....	9-12
Timing Constraint and Report Files for Arria 10 EMIF IP.....	9-13
Timing Analysis Description .....	9-15
UniPHY IP Timing Analysis.....	9-15
Timing Analysis Description for Arria 10 EMIF IP.....	9-25
Timing Report DDR.....	9-28
Report SDC.....	9-30
Calibration Effect in Timing Analysis.....	9-31
Calibration Emulation for Calibrated Path.....	9-31
Calibration Error or Quantization Error.....	9-32
Calibration Uncertainties.....	9-32
Memory Calibration.....	9-32
Timing Model Assumptions and Design Rules.....	9-33
Memory Clock Output Assumptions.....	9-33
Write Data Assumptions.....	9-34
Read Data Assumptions.....	9-36
DLL Assumptions.....	9-37
PLL and Clock Network Assumptions for Stratix III Devices.....	9-37
Common Timing Closure Issues.....	9-38
Missing Timing Margin Report.....	9-38
Incomplete Timing Margin Report.....	9-38
Read Capture Timing.....	9-38
Write Timing.....	9-39
Address and Command Timing.....	9-39
PHY Reset Recovery and Removal.....	9-39
Clock-to-Strobe (for DDR and DDR2 SDRAM Only).....	9-40
Read Resynchronization and Write Leveling Timing (for SDRAM Only).....	9-40
Optimizing Timing.....	9-40
Timing Derivation Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs.....	9-41
Multiple Chip Select Configuration Effects.....	9-42
Timing Derivation using the Board Settings.....	9-43
Early I/O Timing Estimation for Arria 10 EMIF IP.....	9-46
Performing Early I/O Timing Analysis for Arria 10 EMIF IP.....	9-47
Performing I/O Timing Analysis.....	9-48
Performing I/O Timing Analysis with 3rd Party Simulation Tools.....	9-48
Performing Advanced I/O Timing Analysis with Board Trace Delay Model.....	9-48
Document Revision History.....	9-49

<b>Debugging Memory IP.....</b>	<b>10-1</b>
Resource and Planning Issues.....	10-1
Dedicated IOE DQS Group Resources and Pins.....	10-1
Dedicated DLL Resources.....	10-2
Specific PLL Resources.....	10-2
Specific Global, Regional and Dual-Regional Clock Net Resources.....	10-3
Planning Your Design.....	10-3
Optimizing Design Utilization.....	10-3
Interface Configuration Performance Issues.....	10-4
Interface Configuration Bottleneck and Efficiency Issues.....	10-4
Functional Issue Evaluation.....	10-5
Correct Combination of the Quartus Prime Software and ModelSim-Altera Device Models.....	10-5
Altera IP Memory Model.....	10-6
Vendor Memory Model.....	10-6
Insufficient Memory in Your PC.....	10-6
Transcript Window Messages.....	10-7
Passing Simulation.....	10-8
Modifying the Example Driver to Replicate the Failure.....	10-8
Timing Issue Characteristics.....	10-9
Evaluating FPGA Timing Issues.....	10-9
Evaluating External Memory Interface Timing Issues.....	10-10
Verifying Memory IP Using the SignalTap II Logic Analyzer.....	10-11
Signals to Monitor with the SignalTap II Logic Analyzer.....	10-12
Hardware Debugging Guidelines.....	10-13
Create a Simplified Design that Demonstrates the Same Issue.....	10-13
Measure Power Distribution Network.....	10-13
Measure Signal Integrity and Setup and Hold Margin.....	10-13
Vary Voltage.....	10-13
Use Freezer Spray and Heat Gun.....	10-14
Operate at a Lower Speed.....	10-14
Determine Whether the Issue Exists in Previous Versions of Software.....	10-14
Determine Whether the Issue Exists in the Current Version of Software.....	10-14
Try A Different PCB.....	10-15
Try Other Configurations.....	10-15
Debugging Checklist.....	10-15
Catagorizing Hardware Issues.....	10-16
Signal Integrity Issues.....	10-16
Hardware and Calibration Issues.....	10-18
EMIF Debug Toolkit Overview.....	10-22
Document Revision History.....	10-22
<b>Optimizing the Controller.....</b>	<b>11-1</b>
Factors Affecting Efficiency.....	11-1
Interface Standard.....	11-2
Bank Management Efficiency.....	11-2

Data Transfer.....	11-4
Ways to Improve Efficiency.....	11-5
DDR2 SDRAM Controller.....	11-5
Auto-Precharge Commands.....	11-6
Additive Latency.....	11-8
Bank Interleaving.....	11-9
Command Queue Look-Ahead Depth.....	11-11
Additive Latency and Bank Interleaving.....	11-12
User-Controlled Refresh.....	11-13
Frequency of Operation.....	11-13
Burst Length.....	11-14
Series of Reads or Writes.....	11-14
Data Reordering.....	11-15
Starvation Control.....	11-15
Command Reordering.....	11-16
Bandwidth.....	11-17
Efficiency Monitor.....	11-18
Document Revision History.....	11-19
<b>PHY Considerations.....</b>	<b>12-1</b>
Core Logic and User Interface Data Rate.....	12-1
Hard and Soft Memory PHY.....	12-2
Sequencer.....	12-2
PLL, DLL and OCT Resource Sharing.....	12-3
Pin Placement Consideration.....	12-4
Document Revision History.....	12-5
<b>Power Estimation Methods for External Memory Interfaces.....</b>	<b>13-1</b>
Performing Vector-Based Power Analysis with the PowerPlay Power Analyzer.....	13-2
Document Revision History.....	13-2

# Planning Pin and FPGA Resources

1

2016.05.02

EMI\_DG



Subscribe



Send Feedback

This information is for board designers who must determine FPGA pin usage, to create board layouts. The board design process sometimes occurs concurrently with the RTL design process.

Use this document with the External Memory Interfaces chapter of the relevant device family handbook.

Typically, all external memory interfaces require the following FPGA resources:

- Interface pins
- PLL and clock network
- DLL
- Other FPGA resources—for example, core fabric logic, and on-chip termination (OCT) calibration blocks

After you know the requirements for your external memory interface, you can start planning your system. The I/O pins and internal memory cannot be shared for other applications or external memory interfaces. However, if you do not have enough PLLs, DLLs, or clock networks for your application, you may share these resources among multiple external memory interfaces or modules in your system.

Ideally, any interface should reside entirely in a single bank; however, interfaces that span multiple adjacent banks or the entire side of a device are also fully supported. In addition, you may also have wraparound memory interfaces, where the design uses two adjacent sides of the device and the memory interface logic resides in a device quadrant. In some cases, top or bottom bank interfaces have higher supported clock rates than left or right or wraparound interfaces.

## Interface Pins

Any I/O banks that do not support transceiver operations in Arria® II, Arria V, Arria 10, Stratix® III, Stratix IV, and Stratix V devices support external memory interfaces. However, DQS (data strobe or data clock) and DQ (data) pins are listed in the device pin tables and fixed at specific locations in the device. You must adhere to these pin locations as these locations are optimized in routing to minimize skew and maximize margin. Always check the external memory interfaces chapters from the device handbooks for the number of DQS and DQ groups supported in a particular device and the pin table for the actual locations of the DQS and DQ pins.

The following table lists a summary of the number of pins required for various example memory interfaces. This table uses series OCT with calibration and parallel OCT with calibration, or dynamic calibrated OCT, when applicable, shown by the usage of R<sub>UP</sub> and R<sub>DN</sub> pins or RZQ pin.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

**Table 1-1: Pin Counts for Various Example External Memory Interfaces<sup>(1)(2)</sup>**

External Memory Interface	FPGA DQS Group Size	Number of DQ Pins	Number of DQS/CQ/QK Pins	Number of Control Pins <sup>(19)</sup>	Number of Address Pins <sup>(3)</sup>	Number of Command Pins	Number of Clock Pins	R <sub>UP</sub> /R <sub>DN</sub> Pins <sup>(4)</sup>	R <sub>ZQ</sub> Pins <sup>(11)</sup>	Total Pins (with R <sub>UP</sub> /R <sub>DN</sub> pins)	Total Pins (with R <sub>ZQ</sub> pin)
LPDDR2	x8	8	2	1	10	2	2	N/A	1	N/A	26
		16	4	2	10	2	2	N/A	1	N/A	37
		72	18	9	10	2	2	N/A	1	N/A	114
LPDDR3	x8	16	4	2	10	2	2	N/A	1	N/A	37
		72	18	9	10	2	2	N/A	1	N/A	114
DDR4 SDRAM <sup>(12)</sup>	x4	4	2	0 <sup>(7)</sup>	17	11	2	N/A	1	N/A	37
	x8	8	2	1	17	11	2	N/A	1	N/A	42
		16	4	2	17	10 <sup>(13)</sup>	2	N/A	1	N/A	52
DDR3 SDRAM <sup>(5)(6)</sup>	x4	4	2	0 <sup>(7)</sup>	14	10	2	2	1	34	33
	x8	8	2	1	14	10	2	2	1	39	38
		16	4	2	14	10	2	2	1	50	49
DDR2 SDRAM <sup>(8)</sup>	x4	4	1	1 <sup>(7)</sup>	15	9	2	2	1	34	33
	x8	8	1 <sup>(9)</sup>	1	15	9	2	2	1	38	37
		16	2 <sup>(9)</sup>	2	15	9	2	2	1	48	47
DDR SDRAM <sup>(6)</sup>	x4	4	1	1 <sup>(7)</sup>	14	7	2	2	1	29	28
	x8	8	1	1	14	7	2	2	1	33	35
		16	2	2	14	7	2	2	1	43	42
QDR II + / II+ Xtreme SRAM <sup>(18)</sup>	x18	36	2	2	19	3 <sup>(10)</sup>	2 <sup>(15)</sup>	2	1	66	65
	x36	72	2	4	18	3 <sup>(10)</sup>	2 <sup>(15)</sup>	2	1	10	102

External Memory Interface	FPGA DQS Group Size	Number of DQ Pins	Number of DQS/CQ/QK Pins	Number of Control Pins <sup>(19)</sup>	Number of Address Pins <sup>(3)</sup>	Number of Command Pins	Number of Clock Pins	R <sub>UP</sub> /R <sub>DN</sub> Pins <sup>(4)</sup>	R <sub>ZQ</sub> Pins <sup>(11)</sup>	Total Pins (with R <sub>UP</sub> /R <sub>DN</sub> pins)	Total Pins (with R <sub>ZQ</sub> pin)
QDR II SRAM	×9	18	2	1	19	2	4 <sup>(16)</sup>	2	1	48	47
	×18	36	2	2	18	2	4 <sup>(16)</sup>	2	1	66	65
	×36	72	2	4	17	2	4 <sup>(16)</sup>	2	1	103	102
QDR IV SRAM <sup>(20)</sup>	x18	36	8	5	22	7	10 <sup>(17)</sup>	N/A	1	N/A	89
	x36	72	8	5	21	7	10 <sup>(17)</sup>	N/A	1	N/A	124
RLDRA M 3 CIO <sup>(14)</sup>	x9	18	4	2	20	8 <sup>(10)</sup>	6 <sup>(17)</sup>	N/A	1	N/A	59
		36	8	2	19	8 <sup>(10)</sup>	6 <sup>(17)</sup>	N/A	1	N/A	80
RLDRA M II CIO	x9	9	2	1	22	7 <sup>(10)</sup>	4 <sup>(17)</sup>	2	1	47	46
		18	4	1	21	7 <sup>(10)</sup>	4 <sup>(17)</sup>	2	1	57	56
	×18	36	4	1	20	7 <sup>(10)</sup>	6 <sup>(17)</sup>	2	1	76	75

Notes to table:

1. These example pin counts are derived from memory vendor data sheets. Check the exact number of address and command pins of the memory devices in the configuration that you are using.
2. PLL and DLL input reference clock pins are not counted in this calculation.
3. The number of address pins depends on the memory device density.
4. Some DQS or DQ pins are dual purpose and can also be required as R<sub>UP</sub>, R<sub>DN</sub>, or configuration pins. A DQS group is lost if you use these pins for configuration or as R<sub>UP</sub> or R<sub>DN</sub> pins for calibrated OCT. Pick R<sub>UP</sub> and R<sub>DN</sub> pins in a DQS group that is not used for memory interface purposes. You may need to place the DQS and DQ pins manually if you place the R<sub>UP</sub> and R<sub>DN</sub> pins in the same DQS group pins.
5. The TDQS and TDQS# pins are not counted in this calculation, as these pins are not used in the memory controller.
6. Numbers are based on 1-GB memory devices.
7. Altera® FPGAs do not support DM pins in ×4 mode with differential DQS signaling.
8. Numbers are based on 2-GB memory devices without using differential DQS, RDQS, and RDQS# pin support.
9. Assumes single ended DQS mode. DDR2 SDRAM also supports differential DQS, which makes these DQS and DM numbers identical to DDR3 SDRAM.
10. The QVLD pin that indicates read data valid from the QDR II+ SRAM or RLDRAM II device, is included in this number.
11. R<sub>ZQ</sub> pins are supported by Arria V, Arria 10, Cyclone V, and Stratix V devices.

External Memory Interface	FPGA DQS Group Size	Number of DQ Pins	Number of DQS/CQ/QK Pins	Number of Control Pins (19)	Number of Address Pins (3)	Number of Command Pins	Number of Clock Pins	R <sub>UP</sub> /R <sub>DN</sub> Pins (4)	R <sub>ZQ</sub> Pins (11)	Total Pins (with R <sub>UP</sub> /R <sub>DN</sub> pins)	Total Pins (with R <sub>ZQ</sub> pin)
---------------------------	---------------------	-------------------	--------------------------	-----------------------------	----------------------------	------------------------	----------------------	---	---------------------------	---	---------------------------------------

12. Numbers are based on 2-GB discrete device with alert flag and address and command parity pins included.
13. DDR4 x16 devices support only a bank group of 1.
14. Numbers are based on a 576-MB device.
15. These numbers include K and K# clock pins. The CQ and CQ# clock pins are calculated in a separate column.
16. These numbers include K, K#, C, and C# clock pins. The CQ and CQ# clock pins are calculated in a separate column.
17. These numbers include CK, CK#, DK, and DK# clock pins. QK and QK# clock pins are calculated in a separate column.
18. This number is based on a 36864-kilobit device.
19. For DDR,DDR2,DDR3,LDDR2, LDDR3 SDRAM, RLDRAM 3, RLDRAM II, they are DM pins. For QDR II/II+/Extreme, they are BWS pins. For DDR4, they are DM/DBI pins. For QDR IV, they are DINVA[1:0], DINVB[1:0], and AINV.
20. This number is based on a 144-Mbit device with address bus inversion and data bus inversion bits included.

**Note:** Maximum interface width varies from device to device depending on the number of I/O pins and DQS or DQ groups available. Achievable interface width also depends on the number of address and command pins that the design requires. To ensure adequate PLL, clock, and device routing resources are available, you should always test fit any IP in the Quartus® Prime software before PCB sign-off.

Altera devices do not limit the width of external memory interfaces beyond the following requirements:

- Maximum possible interface width in any particular device is limited by the number of DQS groups available.
- Sufficient clock networks are available to the interface PLL as required by the IP.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other address and command, and clock pin placement requirements.
- The greater the number of banks, the greater the skew, hence Altera recommends that you always generate a test project of your desired configuration and confirm that it meets timing.

## Estimating Pin Requirements

You should use the Quartus Prime software for final pin fitting; however, you can estimate whether you have enough pins for your memory interface using the following steps:

1. Find out how many read data pins are associated per read data strobe or clock pair, to determine which column of the DQS and DQ group availability ( $\times 4$ ,  $\times 8/\times 9$ ,  $\times 16/\times 18$ , or  $\times 32/\times 36$ ) refer to the pin table.
2. Check the device density and package offering information to see if you can implement the interface in one I/O bank or on one side or on two adjacent sides.

**Note:** If you target Arria II GX devices and you do not have enough I/O pins to have the memory interface on one side of the device, you may place them on the other side of the device. Arria II GX devices allow a memory interface to span across the top and bottom, or left and right sides

of the device. For any interface that spans across two different sides, use the wraparound interface performance.

3. Calculate the number of other memory interface pins needed, including any other clocks (write clock or memory system clock), address, command, RUP, RDN, RZQ, and any other pins to be connected to the memory components. Ensure you have enough pins to implement the interface in one I/O bank or one side or on two adjacent sides.

**Note:**

1. The DQS groups in Arria II GX devices reside on I/O modules, each consisting of 16 I/O pins. You can only use a maximum of 12 pins per I/O modules when the pins are used as DQS or DQ pins or HSTL/SSTL output or HSTL/SSTL bidirectional pins. When counting the number of available pins for the rest of your memory interface, ensure you do not count the leftover four pins per I/O modules used for DQS, DQ, address and command pins. The leftover four pins can be used as input pins only.
2. Refer to the device pin-out tables and look for the blank space in the relevant DQS group column to identify the four pins that cannot be used in an I/O module for Arria II GX devices.
3. If you enable Ping Pong PHY, the IP core exposes two independent Avalon interfaces to user logic, and a single external memory interface of double the width for the data bus and the CS#, CKE, ODT, and CK/CK# signals. The rest remain as if in single interface configuration.

You should test the proposed pin-outs with the rest of your design in the Quartus Prime software (with the correct I/O standard and OCT connections) before finalizing the pin-outs. There can be interactions between modules that are illegal in the Quartus Prime software that you might not know about unless you compile the design and use the Quartus Prime Pin Planner.

## DDR, DDR2, DDR3, and DDR4 SDRAM Clock Signals

DDR, DDR2, DDR3, and DDR4 SDRAM devices use CK and CK# signals to clock the address and command signals into the memory. Furthermore, the memory uses these clock signals to generate the DQS signal during a read through the DLL inside the memory. The SDRAM data sheet specifies the following timings:

- $t_{DQSCK}$  is the skew between the CK or CK# signals and the SDRAM-generated DQS signal
- $t_{DSH}$  is the DQS falling edge from CK rising edge hold time
- $t_{DSS}$  is the DQS falling edge from CK rising edge setup time
- $t_{DQSS}$  is the positive DQS latching edge to CK rising edge

SDRAM have a write requirement ( $t_{DQSS}$ ) that states the positive edge of the DQS signal on writes must be within  $\pm 25\%$  ( $\pm 90^\circ$ ) of the positive edge of the SDRAM clock input. Therefore, you should generate the CK and CK# signals using the DDR registers in the IOE to match with the DQS signal and reduce any variations across process, voltage, and temperature. The positive edge of the SDRAM clock, CK, is aligned with the DQS write to satisfy  $t_{DQSS}$ .

DDR3 SDRAM can use a daisy-chained control address command (CAC) topology, in which the memory clock must arrive at each chip at a different time. To compensate for the flight-time skew between devices when using the CAC topology, you should employ write leveling.

## DDR, DDR2, DDR3, and DDR4 SDRAM Command and Address Signals

Command and address signals in SDRAM devices are clocked into the memory device using the CK or CK# signal. These pins operate at single data rate (SDR) using only one clock edge. The number of address pins depends on the SDRAM device capacity. The address pins are multiplexed, so two clock cycles are required to send the row, column, and bank address.

For DDR, DDR2, and DDR3, the CS#, RAS#, CAS#, WE#, CKE, and ODT pins are SDRAM command and control pins. For DDR3 SDRAM, certain topologies such as RDIMM and LRDIMM include RESET#, PAR\_IN (1.5V LVCMOS I/O standard), and ERR\_OUT# (SSTL-15 I/O standard).

The DDR2 SDRAM command and address inputs do not have a symmetrical setup and hold time requirement with respect to the SDRAM clocks, CK, and CK#.

Although DDR4 operates in fundamentally the same way as other SDRAM, there are no longer dedicated pins for RAS#, CAS#, and WE#, as those are now shared with higher-order address pins. DDR4 still has CS#, CKE, ODT, and RESET# pins, similar to DDR3. DDR4 introduces some additional pins, including the ACT# (activate) pin and BG (bank group) pins. Depending on the memory format and the functions enabled, the following pins might also exist in DDR4: PAR (address command parity) pin and the ALERT# pin.

For Altera SDRAM high-performance controllers in Stratix III and Stratix IV devices, the command and address clock is a dedicated PLL clock output whose phase can be adjusted to meet the setup and hold requirements of the memory clock. The command and address clock is also typically half-rate, although a full-rate implementation can also be created. The command and address pins use the DDIO output circuitry to launch commands from either the rising or falling edges of the clock. The chip select CS#, clock enable CKE, and ODT pins are only enabled for one memory clock cycle and can be launched from either the rising or falling edge of the command and address clock signal. The address and other command pins are enabled for two memory clock cycles and can also be launched from either the rising or falling edge of the command and address clock signal.

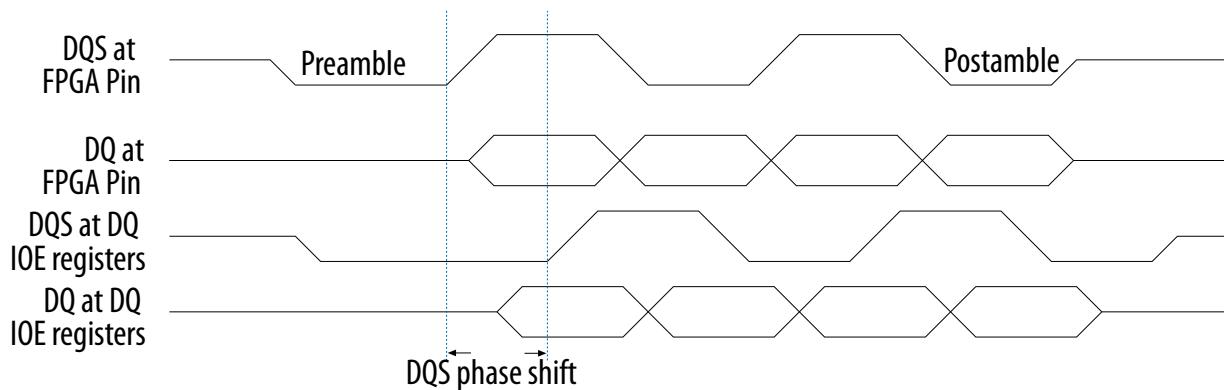
In Arria II GX devices, the command and address clock is either shared with the write\_clk\_2x or the mem\_clk\_2x clock.

## DDR, DDR2, DDR3, and DDR4 SDRAM Data, Data Strobes, DM/DBI, and Optional ECC Signals

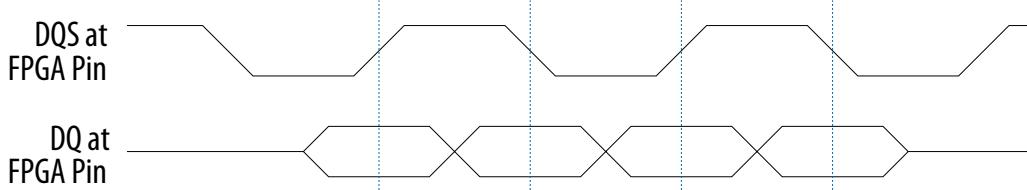
DDR SDRAM uses bidirectional single-ended data strobe (DQS); DDR3 and DDR4 SDRAM use bidirectional differential data strobes. The DQSn pins in DDR2 SDRAM devices are optional but recommended for DDR2 SDRAM designs operating at more than 333 MHz. Differential DQS operation enables improved system timing due to reduced crosstalk and less simultaneous switching noise on the strobe output drivers. The DQ pins are also bidirectional.

Regardless of interface width, DDR SDRAM always operates in  $\times 8$  mode DQS groups. DQ pins in DDR2, DDR3, and DDR4 SDRAM interfaces can operate in either  $\times 4$  or  $\times 8$  mode DQS groups, depending on your chosen memory device or DIMM, regardless of interface width. The  $\times 4$  and  $\times 8$  configurations use one pair of bidirectional data strobe signals, DQS and DQSn, to capture input data. However, two pairs of data strobes, UDQS and UDQS# (upper byte) and LDQS and LDQS# (lower byte), are required by the  $\times 16$  configuration devices. A group of DQ pins must remain associated with its respective DQS and DQSn pins.

The DQ signals are edge-aligned with the DQS signal during a read from the memory and are center-aligned with the DQS signal during a write to the memory. The memory controller shifts the DQ signals by -90 degrees during a write operation to center align the DQ and DQS signals. The PHY IP delays the DQS signal during a read, so that the DQ and DQS signals are center aligned at the capture register. Altera devices use a phase-locked loop (PLL) to center-align the DQS signal with respect to the DQ signals during writes and Altera devices use dedicated DQS phase-shift circuitry to shift the incoming DQS signal during reads. The following figure shows an example where the DQS signal is shifted by 90 degrees for a read from the DDR2 SDRAM.

**Figure 1-1: Edge-aligned DQ and DQS Relationship During a DDR2 SDRAM Read in Burst-of-Four Mode**

The following figure shows an example of the relationship between the data and data strobe during a burst-of-four write.

**Figure 1-2: DQ and DQS Relationship During a DDR2 SDRAM Write in Burst-of-Four Mode**

The memory device's setup ( $t_{DS}$ ) and hold times ( $t_{DH}$ ) for the DQ and DM pins during writes are relative to the edges of DQS write signals and not the CK or CK# clock. Setup and hold requirements are not necessarily balanced in DDR2 and DDR3 SDRAM, unlike in DDR SDRAM devices.

The DQS signal is generated on the positive edge of the system clock to meet the  $t_{DQSS}$  requirement. DQ and DM signals use a clock shifted -90 degrees from the system clock, so that the DQS edges are centered on the DQ or DM signals when they arrive at the DDR2 SDRAM. The DQS, DQ, and DM board trace lengths need to be tightly matched (within 20 ps).

The SDRAM uses the DM pins during a write operation. Driving the DM pins low shows that the write is valid. The memory masks the DQ signals if the DM pins are driven high. To generate the DM signal, Altera recommends that you use the spare DQ pin within the same DQS group as the respective data, to minimize skew.

The DM signal's timing requirements at the SDRAM input are identical to those for DQ data. The DDR registers, clocked by the -90 degree shifted clock, create the DM signals.

DDR4 supports DM similarly to other SDRAM, except that in DDR4 DM is active LOW and bidirectional, because it supports Data Bus Inversion (DBI) through the same pin. DM is multiplexed with DBI by a Mode Register setting whereby only one function can be enabled at a time. DBI is an input/output identifying whether to store/output the true or inverted data. When enabled, if DBI is LOW, during a write operation the data is inverted and stored inside the DDR4 SDRAM; during a read operation, the data is inverted and output. The data is not inverted if DBI is HIGH. For Arria 10, the DBI (for DDR4) and the DM (for DDR3) pins in each DQS group must be paired with a DQ pin for proper operation.

Some SDRAM modules support error correction coding (ECC) to allow the controller to detect and automatically correct error in data transmission. The 72-bit SDRAM modules contain eight extra data pins in addition to 64 data pins. The eight extra ECC pins should be connected to a single DQS or DQ group on the FPGA.

## DDR, DDR2, DDR3, and DDR4 SDRAM DIMM Options

Unbuffered DIMMs (UDIMMs) require one set of chip-select (CS#), on-die termination (ODT), clock-enable (CKE), and clock pair (CK/CKn) for every physical rank on the DIMM. Registered DIMMs use only one pair of clocks. DDR3 registered DIMMs require a minimum of two chip-select signals, while DDR4 requires only one.

Compared to the unbuffered DIMMs (UDIMM), registered and load-reduced DIMMs (RDIMMs and LRDIMMs, respectively) use at least two chip-select signals CS#[1:0] in DDR3 and DDR4. Both RDIMMs and LRDIMMs require an additional parity signal for address, RAS#, CAS#, and WE# signals. A parity error signal is asserted by the module whenever a parity error is detected.

LRDIMMs expand on the operation of RDIMMs by buffering the DQ/DQS bus. Only one electrical load is presented to the controller regardless of the number of ranks, therefore only one clock enable (CKE) and ODT signal are required for LRDIMMs, regardless of the number of physical ranks. Because the number of physical ranks may exceed the number of physical chip-select signals, DDR3 LRDIMMs provide a feature known as rank multiplication, which aggregates two or four physical ranks into one larger logical rank. Refer to LRDIMM buffer documentation for details on rank multiplication.

The following table shows UDIMM and RDIMM pin options for DDR, DDR2, and DDR3.

**Table 1-2: UDIMM and RDIMM Pin Options for DDR, DDR2, and DDR3**

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)
Data	72 bit DQ[71:0] = {CB[7:0], DQ[63:0]}			
Data Mask	DM[8:0]	DM[8:0]	DM[8:0]	DM[8:0]
Data Strobe <sup>(1)</sup>	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]
Address	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]
Clock	CK0/CK0#	CK0/CK0#, CK1/CK1#	CK0/CK0#	CK0/CK0#

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)
Command	ODT, CS#, CKE, RAS#, CAS#, WE#	ODT[1:0], CS#[1:0], CKE[1:0], RAS#, CAS#, WE#	ODT, CS#[1:0], CKE, RAS#, CAS#, WE# <sup>2</sup>	ODT[1:0], CS#[1:0], CKE[1:0], RAS#, CAS#, WE#
Parity	—	—	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT
Other Pins	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#

Note to Table:

1. DQS#[8:0] is optional in DDR2 SDRAM and is not supported in DDR SDRAM interfaces.
2. For single rank DDR2 RDIMM, ignore CS#[1] because it is not used.

The following table shows LRDIMM pin options for DDR3.

**Table 1-3: LRDIMM Pin Options for DDR3**

Pins	LRDIMM Pins (x4, 2R)	LRDIMM (x4, 4R, RMF=1) <sup>3</sup>	LRDIMM Pins (x4, 4R, RMF=2)	LRDIMM Pins (x4, 8R, RMF=2)	LRDIMM Pins (x4, 8R, RMF=4)	LRDIMM (x8, 4R, RMF=1) <sup>3</sup>	LRDIMM Pins (x8, 4R, RMF=2)
Data	72 bit DQ [71:0]= {CB [7:0], DQ [63:0]}	72 bit DQ [71:0]= {CB [7:0], DQ [63:0]}	72 bit DQ [71:0]= {CB [7:0], DQ [63:0]}	72 bit DQ [71:0]= {CB [7:0], DQ [63:0]}	72 bit DQ [71:0]= {CB [7:0], DQ [63:0]}	72 bit DQ [71:0]= {CB [7:0], DQ [63:0]}	72 bit DQ [71:0]= {CB [7:0], DQ [63:0]}
Data Mask	—	—	—	—	—	DM[8:0]	DM[8:0]
Data Strobe	DQS[17:0] and DQS#[17:0]	DQS[17:0] and DQS#[17:0]	DQS[17:0] and DQS#[17:0]	DQS[17:0] and DQS#[17:0]	DQS[17:0] and DQS#[17:0]	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]
Address	BA[2:0], A[15:0] — 2GB:A[13:0] 4GB:A[14:0] 8GB:A[15:0]	BA[2:0], A[15:0] — 2GB:A[13:0] 4GB:A[14:0] 8GB:A[15:0]	BA[2:0], A[16:0] — 4GB:A[14:0] 8GB:A[15:0] 16GB:A[16:0]	BA[2:0], A[16:0] — 4GB:A[14:0] 8GB:A[15:0] 16GB:A[16:0]	BA[2:0], A[17:0] — 16GB:A[15:0] 32GB:A[16:0] 64GB:A[17:0]	BA[2:0], A[15:0] — 4GB:A[14:0] 8GB:A[15:0] 16GB:A[16:0]	BA[2:0], A[16:0] —4GB:A[14:0] 8GB:A[15:0] 16GB:A[16:0]

Pins	LRDIMM Pins (x4, 2R)	LRDIMM (x4, 4R, RMF=1) <sup>3</sup>	LRDIMM Pins (x4, 4R, RMF=2)	LRDIMM Pins (x4, 8R, RMF=2)	LRDIMM Pins (x4, 8R, RMF=4)	LRDIMM (x8, 4R, RMF=1) <sup>3</sup>	LRDIMM Pins (x8, 4R, RMF=2)
Clock	CK0/CK0#	CK0/CK0#	CK0/CK0#	CK0/CK0#	CK0/CK0#	CK0/CK0#	CK0/CK0#
Command	ODT, CS[1:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[3:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[2:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[3:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[3:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[2:0]#, CKE, RAS#, CAS#, WE#	ODT, CS[2:0]#, CKE, RAS#, CAS#, WE#
Parity	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT	PAR_IN, ERR_OUT
Other Pins	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#					

Notes to Table:

1. DM pins are not used for LRDIMMs that are constructed using  $\times 4$  components.
2. S#[2] is treated as A[16] (whose corresponding pins are labeled as CS#[2] or RM[0]) and S#[3] is treated as A[17] (whose corresponding pins are labeled as CS#[3] or RM[1]) for certain rank multiplication configuration.
3. R = rank, RMF = rank multiplication factor.

The following table shows UDIMM, RDIMM, and LRDIMM pin options for DDR4.

Table 1-4: UDIMM, RDIMM, and LRDIMM Pin Options for DDR4

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)	LRDIMM Pins (Dual Rank)	LRDIMM Pins (Quad Rank)
Data	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}
Data Mask	DM#/ DBI#[8:0] <sup>(1)</sup>	DM#/ DBI#[8:0] <sup>(1)</sup>	DM#/ DBI#[8:0] <sup>(1)</sup>	DM#/ DBI#[8:0] <sup>(1)</sup>	—	—
Data Strobe	x8: DQS[8:0] and DQS#[8:0]	x8: DQS[8:0] and DQS#[8:0]	x8: DQS[8:0] and DQS#[8:0] x4: DQS[17:0] and DQS#[17:0]	x8: DQS[8:0] and DQS#[8:0] x4: DQS[17:0] and DQS#[17:0]	x4: DQS[17:0] and DQS#[17:0]	x4: DQS[17:0] and DQS#[17:0]

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)	LRDIMM Pins (Dual Rank)	LRDIMM Pins (Quad Rank)
Address	BA[1:0], BG[1:0], A[16:0] -  4GB: A[14:0]  8GB: A[15:0]  16GB: A[16:0] <sup>(2)</sup>	BA[1:0], BG[1:0], A[16:0] -  8GB: A[14:0]  16GB: A[15:0]  32GB: A[16:0] <sup>(2)</sup>	BA[1:0], BG[1:0], x8: A[16:0]  4GB: A[14:0]  8GB: A[15:0]  16GB: A[16:0] <sup>(2)</sup>  32GB: A[17:0] <sup>(3)</sup>	BA[1:0], BG[1:0], A[16:0] x4: A[17:0] -  8GB: A[14:0]  16GB: A[15:0]  32GB: A[16:0] <sup>(2)</sup>  64GB: A[17:0] <sup>(3)</sup>	BA[1:0], BG[1:0], A[17:0] -  16GB: A[15:0]  32GB: A[16:0] <sup>(2)</sup>  64GB: A[17:0] <sup>(3)</sup>	BA[1:0], BG[1:0], A[17:0] -  32GB: A[15:0]  64GB: A[16:0] <sup>(2)</sup>  128GB: A[17:0] <sup>(3)</sup>
Clock	CK0/CK0#	CK0/CK0#, CK1/CK1#	CK0/CK0#	CK0/CK0#	CK0/CK0#	CK0/CK0#
Command	ODT, CS#, CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT[1:0], CS#[1:0], CKE[1:0], RAS#/A16, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT, CS#, CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT[1:0], CS#[1:0], CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT, CS#[1:0], CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT, CS#[3:0], CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14
Parity	PAR, ALERT#	PAR, ALERT#	PAR, ALERT#	PAR, ALERT#	PAR, ALERT#	PAR, ALERT#
Other Pins	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#

Notes to Table:

1. DM/DBI pins are available only for DIMMs constructed using x8 or greater components.
2. This density requires 4Gb x4 or 2Gb x8 DRAM components.
3. This density requires 8Gb x4 DRAM components.
4. This table assumes a single slot configuration. The Arria 10 memory controller can support up to 4 ranks per channel. A single slot interface may have up to 4 ranks, and a dual slot interface may have up to 2 ranks per slot. In either case, the total number of ranks, calculated as the number of slots multiplied by the number of ranks per slot, must be less than or equal to 4.

## QDR II, QDR II+, and QDR II+ Xtreme SRAM Clock Signals

QDR II, QDR II+ and QDR II+ Xtreme SRAM devices have two pairs of clocks, listed below.

- Input clocks K and K#
- Echo clocks CQ and CQ#

In addition, QDR II devices have a third pair of input clocks, C and C#.

The positive input clock,  $\kappa$ , is the logical complement of the negative input clock,  $\kappa\#$ . Similarly,  $c$  and  $c_Q$  are complements of  $c\#$  and  $c_Q\#$ , respectively. With these complementary clocks, the rising edges of each clock leg latch the DDR data.

The QDR II SRAM devices use the  $\kappa$  and  $\kappa\#$  clocks for write access and the  $c$  and  $c\#$  clocks for read accesses only when interfacing more than one QDR II SRAM device. Because the number of loads that the  $\kappa$  and  $\kappa\#$  clocks drive affects the switching times of these outputs when a controller drives a single QDR II SRAM device,  $c$  and  $c\#$  are unnecessary. This is because the propagation delays from the controller to the QDR II SRAM device and back are the same. Therefore, to reduce the number of loads on the clock traces, QDR II SRAM devices have a single-clock mode, and the  $\kappa$  and  $\kappa\#$  clocks are used for both reads and writes. In this mode, the  $c$  and  $c\#$  clocks are tied to the supply voltage (VDD). Altera external memory IP supports only single-clock mode.

For QDR II, QDR II+, or QDR II+ Xtreme SRAM devices, the rising edge of  $\kappa$  is used to capture synchronous inputs to the device and to drive out data through  $Q[x:0]$ , in similar fashion to QDR II SRAM devices in single clock mode. All accesses are initiated on the rising edge of  $\kappa$ .

$c_Q$  and  $c_Q\#$  are the source-synchronous output clocks from the QDR II, QDR II+, or QDR II+ Xtreme SRAM device that accompanies the read data.

The Altera device outputs the  $\kappa$  and  $\kappa\#$  clocks, data, address, and command lines to the QDR II, QDR II+, or QDR II+ Xtreme SRAM device. For the controller to operate properly, the write data ( $D$ ), address ( $A$ ), and control signal trace lengths (and therefore the propagation times) should be equal to the  $\kappa$  and  $\kappa\#$  clock trace lengths.

You can generate  $\kappa$  and  $\kappa\#$  clocks using any of the PLL registers via the DDR registers. Because of strict skew requirements between  $\kappa$  and  $\kappa\#$  signals, use adjacent pins to generate the clock pair. The propagation delays for  $\kappa$  and  $\kappa\#$  from the FPGA to the QDR II, QDR II+, or QDR II+ Xtreme SRAM device are equal to the delays on the data and address ( $D, A$ ) signals. Therefore, the signal skew effect on the write and read request operations is minimized by using identical DDR output circuits to generate clock and data inputs to the memory.

## QDR II, QDR II+ and QDR II+ Xtreme SRAM Command Signals

QDR II, QDR II+ and QDR II+ Xtreme SRAM devices use the write port select ( $WPS\#$ ) signal to control write operations and the read port select ( $RPS\#$ ) signal to control read operations.

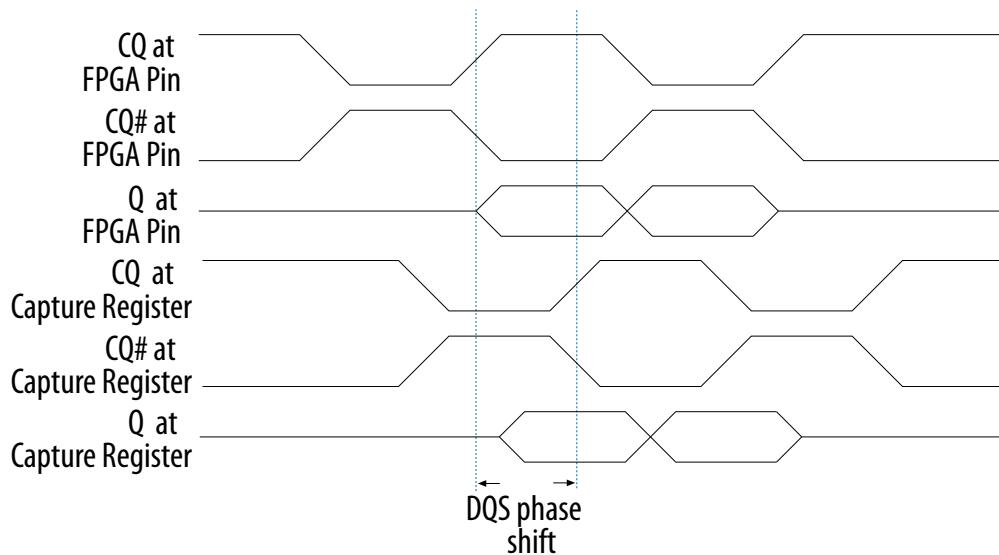
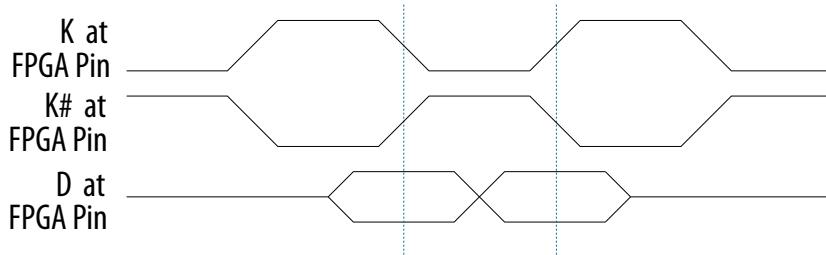
## QDR II, QDR II+ and QDR II+ Xtreme SRAM Address Signals

QDR II, QDR II+ and QDR II+ Xtreme SRAM devices use one address bus ( $A$ ) for both read and write accesses.

## QDR II, QDR II+ and QDR II+ Xtreme SRAM Data, BWS, and QVLD Signals

QDR II, QDR II+ and QDR II+ Xtreme SRAM devices use two unidirectional data buses: one for writes ( $D$ ) and one for reads ( $Q$ ).

At the pin, the read data is edge-aligned with the  $c_Q$  and  $c_Q\#$  clocks while the write data is center-aligned with the  $\kappa$  and  $\kappa\#$  clocks (see the following figures).

**Figure 1-3: Edge-aligned CQ and Q Relationship During QDR II+ SRAM Read****Figure 1-4: Center-aligned K and D Relationship During QDR II+ SRAM Write**

The byte write select signal (`bws#`) indicates which byte to write into the memory device.

QDR II+ and QDR II+ Xtreme SRAM devices also have a QVLD pin that indicates valid read data. The QVLD signal is edge-aligned with the echo clock and is asserted high for approximately half a clock cycle before data is output from memory.

**Note:** The Altera external memory interface IP does not use the QVLD signal.

## QDR IV SRAM Clock Signals

QDR IV SRAM devices have three pairs of differential clocks.

The three QDR IV differential clocks are as follows:

- Address and Command Input Clocks `CK` and `CK#`
- Data Input Clocks `DKx` and `DKx#`, where  $x$  can be A or B, referring to the respective ports
- Data Output Clocks, `QKx` and `QKx#`, where  $x$  can be A or B, referring to the respective ports

QDR IV SRAM devices have two independent bidirectional data ports, Port A and Port B, to support concurrent read/write transactions on both ports. These data ports are controlled by a common address port clocked by `CK` and `CK#` in double data rate. There is one pair of `CK` and `CK#` pins per QDR IV SRAM device.

$\text{DK}_x$  and  $\text{DK}_x\#$  samples the  $\text{DQ}_x$  inputs on both rising and falling edges. Similarly,  $\text{QK}_x$  and  $\text{QK}_x\#$  samples the  $\text{DQ}_x$  outputs on both rising and falling edges.

QDR IV SRAM devices employ two sets of free running differential clocks to accompany the data. The  $\text{DK}_x$  and  $\text{DK}_x\#$  clocks are the differential input data clocks used during writes. The  $\text{QK}_x$  and  $\text{QK}_x\#$  clocks are the output data clocks used during reads. Each pair of  $\text{DK}_x$  and  $\text{DK}_x\#$ , or  $\text{QK}_x$  and  $\text{QK}_x\#$  clocks are associated with either 9 or 18 data bits.

The polarity of the  $\text{QKB}$  and  $\text{QKB}\#$  pins in the Altera external memory interface IP was swapped with respect to the polarity of the differential input buffer on the FPGA. In other words, the  $\text{QKB}$  pins on the memory side must be connected to the negative pins of the input buffers on the FPGA side, and the  $\text{QKB}\#$  pins on the memory side must be connected to the positive pins of the input buffers on the FPGA side. Notice that the port names at the top-level of the IP already reflect this swap (that is, `mem_qkb` is assigned to the negative buffer leg, and `mem_qkb_n` is assigned to the positive buffer leg).

QDR IV SRAM devices are available in x18 and x36 bus width configurations. The exact clock-data relationships are as follows:

- For  $\times 18$  data bus width configuration, there are 9 data bits associated with each pair of write and read clocks. So, there are two pairs of  $\text{DK}_x$  and  $\text{DK}_x\#$  pins and two pairs of  $\text{QK}_x$  or  $\text{QK}_x\#$  pins.
- For  $\times 36$  data bus width configuration, there are 18 data bits associated with each pair of write and read clocks. So, there are two pairs of  $\text{DK}_x$  and  $\text{DK}_x\#$  pins and two pairs of  $\text{QK}_x$  or  $\text{QK}_x\#$  pins.

There are  $t_{CKDK}$  timing requirements for skew between  $\text{CK}$  and  $\text{DK}_x$  or  $\text{CK}\#$  and  $\text{DK}_x\#$ . Similarly, there are  $t_{CKQK}$  timing requirements for skew between  $\text{CK}$  and  $\text{QK}_x$  or  $\text{CK}\#$  and  $\text{QK}_x\#$ .

## QDR IV SRAM Commands and Addresses, AP, and AINV Signals

The  $\text{CK}$  and  $\text{CK}\#$  signals clock the commands and addresses into the memory devices. There is one pair of  $\text{CK}$  and  $\text{CK}\#$  pins per QDR IV SRAM device. These pins operate at double data rate using both rising and falling edge. The rising edge of  $\text{CK}$  latches the addresses for port A, while the falling edge of  $\text{CK}$  latches the addresses inputs for port B.

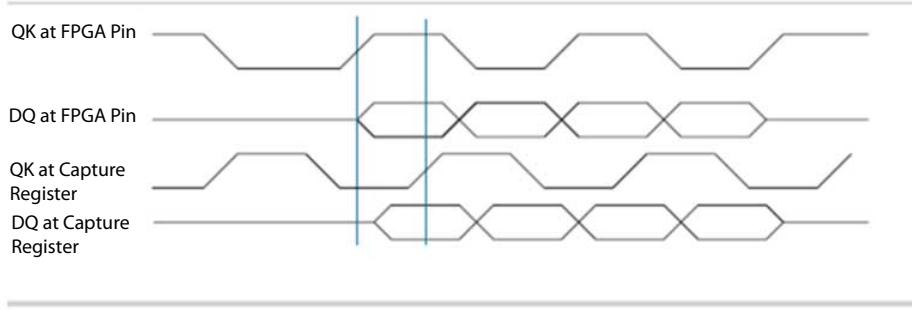
QDR IV SRAM devices have the ability to invert all address pins to reduce potential simultaneous switching noise. Such inversion is accomplished using the Address Inversion Pin for Address and Address Parity Inputs (AINV), which assumes an address parity of 0, and indicates whether the address bus and address parity are inverted.

The above features are available as **Option Control** under **Configuration Register Settings** in Arria 10 EMIF IP. The commands and addresses must meet the memory address and command setup ( $t_{AS}$ ,  $t_{CS}$ ) and hold ( $t_{AH}$ ,  $t_{CH}$ ) time requirements.

## QDR IV SRAM Data, DINV, and QVLD Signals

The read data is edge-aligned with the  $\text{QKA}$  or  $\text{QKB}\#$  clocks while the write data is center-aligned with the  $\text{DKA}$  and  $\text{DKB}\#$  clocks.

$\text{QK}$  is shifted by the DLL so that the clock edges can be used to clock in the  $\text{DQ}$  at the capture register.

**Figure 1-5: Edge-Aligned DQ and QK Relationship During Read****Figure 1-6: Center-Aligned DQ and DK Relationship During Write**

The polarity of the `QKB` and `QKB#` pins in the Altera external memory interface IP was swapped with respect to the polarity of the differential input buffer on the FPGA. In other words, the `QKB` pins on the memory side need to be connected to the negative pins of the input buffers on the FPGA side, and the `QKB#` pins on the memory side need to be connected to the positive pins of the input buffers on the FPGA side. Notice that the port names at the top-level of the IP already reflect this swap (that is, `mem_qkb` is assigned to the negative buffer leg, and `mem_qkb_n` is assigned to the positive buffer leg).

The synchronous read/write input, `RWx#`, is used in conjunction with the synchronous load input, `LDx#`, to indicate a Read or Write Operation. For port A, these signals are sampled on the rising edge of `CK` clock, for port B, these signals are sampled on the falling edge of `CK` clock.

QDR IV SRAM devices have the ability to invert all data pins to reduce potential simultaneous switching noise, using the Data Inversion Pin for DQ Data Bus, `DINVx`. This pin indicates whether `DQx` pins are inverted or not.

To enable the data pin inversion feature, click **Configuration Register Settings > Option Control** in the Arria 10 EMIF IP.

QDR IV SRAM devices also have a `QVLD` pin which indicates valid read data. The `QVLD` signal is edge-aligned with `QKx` or `QKx#` and is high approximately one-half clock cycle before data is output from the memory.

**Note:** The Altera external memory interface IP does not use the `QVLD` signal.

## RLDRAM II and RLDRAM 3 Clock Signals

RLDRAM II and RLDRAM 3 devices use `CK` and `CK#` signals to clock the command and address bus in single data rate (SDR). There is one pair of `CK` and `CK#` pins per RLDRAM II or RLDRAM 3 device.

Instead of a strobe, RLDRAM II and RLDRAM 3 devices use two sets of free-running differential clocks to accompany the data. The `DK` and `DK#` clocks are the differential input data clocks used during writes while

the QK or QK# clocks are the output data clocks used during reads. Even though QK and QK# signals are not differential signals according to the RLDRAM II and RLDRAM 3 data sheets, Micron treats these signals as such for their testing and characterization. Each pair of DK and DK#, or QK and QK# clocks are associated with either 9 or 18 data bits.

The exact clock-data relationships are as follows:

- RLDRAM II: For  $\times 36$  data bus width configuration, there are 18 data bits associated with each pair of write and read clocks. So, there are two pairs of DK and DK# pins and two pairs of QK or QK# pins.
- RLDRAM 3: For  $\times 36$  data bus width configuration, there are 18 data bits associated with each pair of write clocks. There are 9 data bits associated with each pair of read clocks. So, there are two pairs of DK and DK# pins and four pairs of QK and QK# pins.
- RLDRAM II: For  $\times 18$  data bus width configuration, there are 18 data bits per one pair of write clocks and nine data bits per one pair of read clocks. So, there is one pair of DK and DK# pins, but there are two pairs of QK and QK# pins.
- RLDRAM 3: For  $\times 18$  data bus width configuration, there are 9 data bits per one pair of write clocks and nine data bits per one pair of read clocks. So, there are two pairs of DK and DK# pins, and two pairs of QK and QK# pins.
- RLDRAM II: For  $\times 9$  data bus width configuration, there are nine data bits associated with each pair of write and read clocks. So, there is one pair of DK and DK# pins and one pair of QK and QK# pins each.
- RLDRAM 3: RLDRAM 3 does not have the  $\times 9$  data bus width configuration.

There are  $t_{CKDK}$  timing requirements for skew between CK and DK or CK# and DK#.

For both RLDRAM II and RLDRAM 3, because of the loads on these I/O pins, the maximum frequency you can achieve depends on the number of memory devices you are connecting to the Altera device. Perform SPICE or IBIS simulations to analyze the loading effects of the pin-pair on multiple RLDRAM II or RLDRAM 3 devices.

## RLDRAM II and RLDRAM 3 Commands and Addresses

The CK and CK# signals clock the commands and addresses into the memory devices.

These pins operate at single data rate using only one clock edge. RLDRAM II and RLDRAM 3 support both non-multiplexed and multiplexed addressing. Multiplexed addressing allows you to save a few user I/O pins while non-multiplexed addressing allows you to send the address signal within one clock cycle instead of two clock cycles. CS#, REF#, and WE# pins are input commands to the RLDRAM II or RLDRAM 3 device.

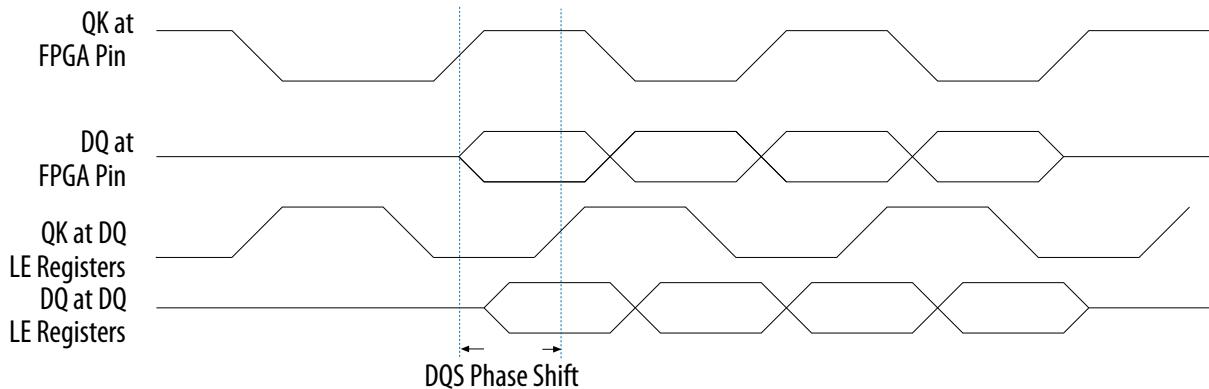
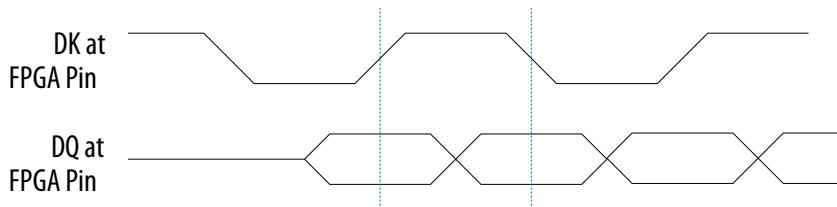
The commands and addresses must meet the memory address and command setup ( $t_{AS}$ ,  $t_{CS}$ ) and hold ( $t_{AH}$ ,  $t_{CH}$ ) time requirements.

**Note:** The Altera RLDRAM II and RLDRAM 3 external memory interface IP do not support multiplexed addressing.

## RLDRAM II and RLDRAM 3 Data, DM and QVLD Signals

The read data is edge-aligned with the QK or QK# clocks while the write data is center-aligned with the DK and DK# clocks (see the following figures). The memory controller shifts the DK and DK# signals to center align the DQ and DK or DK# signals during a write. It also shifts the QK signal during a read, so that the read data (DQ signals) and QK clock is center-aligned at the capture register.

Altera devices use dedicated DQS phase-shift circuitry to shift the incoming QK signal during reads and use a PLL to center-align the DK and DK# signals with respect to the DQ signals during writes.

**Figure 1-7: Edge-aligned DQ and QK Relationship During RLDRAM II or RLDRAM 3 Read****Figure 1-8: Center-aligned DQ and DK Relationship During RLDRAM II or RLDRAM 3 Write**

For RLDRAM II and RLDRAM 3, data mask (DM) pins are used only during a write. The memory controller drives the DM signal low when the write is valid and drives it high to mask the DQ signals.

For RLDRAM II, there is one DM pin per memory device. The DQ input signal is masked when the DM signal is high.

For RLDRAM 3, there are two DM pins per memory device. DM0 is used to mask the lower byte for the x18 device and (DQ[8:0], DQ[26:18]) for the x36 device. DM1 is used to mask the upper byte for the x18 device and (DQ[17:9], DQ[35:27]) for the x36 device.

The DM timing requirements at the input to the memory device are identical to those for DQ data. The DDR registers, clocked by the write clock, create the DM signals. This reduces any skew between the DQ and DM signals.

The RLDRAM II or RLDRAM 3 device's setup time ( $t_{DS}$ ) and hold ( $t_{DH}$ ) time for the write DQ and DM pins are relative to the edges of the DK or DK# clocks. The DK and DK# signals are generated on the positive edge of system clock, so that the positive edge of CK or CK# is aligned with the positive edge of DK or DK# respectively to meet the tCKDK requirement. The DQ and DM signals are clocked using a shifted clock so that the edges of DK or DK# are center-aligned with respect to the DQ and DM signals when they arrive at the RLDRAM II or RLDRAM 3 device.

The clocks, data, and DM board trace lengths should be tightly matched to minimize the skew in the arrival time of these signals.

RLDRAM II and RLDRAM 3 devices also have a QVLD pin indicating valid read data. The QVLD signal is edge-aligned with QK or QK# and is high approximately half a clock cycle before data is output from the memory.

**Note:** The Altera external memory interface IP does not use the QVLD signal.

## LPDDR2 and LPDDR3 Clock Signal

$\text{CK}$  and  $\text{CK}_n$  are differential clock inputs to the LPDDR2 and LPDDR3 interface. All the double data rate (DDR) inputs are sampled on both the positive and negative edges of the clock. Single data rate (SDR) inputs,  $\text{CS}_n$  and  $\text{CKE}$ , are sampled at the positive clock edge.

The clock is defined as the differential pair which consists of  $\text{CK}$  and  $\text{CK}_n$ . The positive clock edge is defined by the cross point of a rising  $\text{CK}$  and a falling  $\text{CK}_n$ . The negative clock edge is defined by the cross point of a falling  $\text{CK}$  and a rising  $\text{CK}_n$ .

The SDRAM data sheet specifies timing data for the following:

- $t_{\text{DSH}}$  is the  $\text{DQS}$  falling edge hold time from  $\text{CK}$ .
- $t_{\text{DSS}}$  is the  $\text{DQS}$  falling edge to the  $\text{CK}$  setup time.
- $t_{\text{DQSS}}$  is the Write command to the first  $\text{DQS}$  latching transition.
- $t_{\text{DQSCk}}$  is the  $\text{DQS}$  output access time from  $\text{CK}/\text{CK}_n$ .

## LPDDR2 and LPDDR3 Command and Address Signal

All LPDDR2 and LPDDR3 devices use double data rate architecture on the command/address bus to reduce the number of input pins in the system. The 10-bit command/address bus contains command, address, and bank/row buffer information. Each command uses one clock cycle, during which command information is transferred on both the positive and negative edges of the clock.

## LPDDR2 and LPDDR3 Data, Data Strobe, and DM Signals

LPDDR2 and LPDDR3 devices use bidirectional and differential data strobes.

Differential  $\text{DQS}$  operation enables improved system timing due to reduced crosstalk and less simultaneous switching noise on the strobe output drivers. The  $\text{DQ}$  pins are also bidirectional.  $\text{DQS}$  is edge-aligned with the read data and centered with the write data.

$\text{DM}$  is the input mask for the write data signal. Input data is masked when  $\text{DM}$  is sampled high coincident with that input data during a write access.

## Maximum Number of Interfaces

The maximum number of interfaces supported for a given memory protocol varies, depending on the FPGA in use.

Unless otherwise noted, the calculation for the maximum number of interfaces is based on independent interfaces where the address or command pins are not shared. The maximum number of independent interfaces is limited to the number of PLLs each FPGA device has.

**Note:** You must share DLLs if the total number of interfaces exceeds the number of DLLs available in a specific FPGA device. You may also need to share PLL clock outputs depending on your clock network usage, refer to *PLLs and Clock Networks*.

**Note:** For information about the number of  $\text{DQ}$  and  $\text{DQS}$  in other packages, refer to the  $\text{DQ}$  and  $\text{DQS}$  tables in the relevant device handbook.

Timing closure depends on device resource and routing utilization. For more information about timing closure, refer to the *Area and Timing Optimization Techniques* chapter in the *Quartus Prime Handbook*.

### Related Information

- [PLLs and Clock Networks](#) on page 1-60
- [Arria 10 Device Handbook](#)

- [Quartus Prime Handbook](#)

## Maximum Number of DDR SDRAM Interfaces Supported per FPGA

The following table describes the maximum number of  $\times 8$  DDR SDRAM components that can fit in the smallest and biggest devices and pin packages assuming the device is blank.

Each interface of size  $n$ , where  $n$  is a multiple of 8, consists of:

- $n$  DQ pins (including error correction coding (ECC))
- $n/8$  DM pins
- $n/8$  DQS pins
- 18 address pins
- 6 command pins (CAS#, RAS#, WE#, CKE, and CS#)
- 1 CK, CK# pin pair for up to every three  $\times 8$  DDR SDRAM components

**Table 1-5: Maximum Number of DDR SDRAM Interfaces Supported per FPGA**

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GX	EP2AGX190	1,152	Four $\times 8$ interfaces or one $\times 72$ interface on each side (no DQ pins on left side)
	EP2AGX260		
	EP2AGX45	358	<ul style="list-style-type: none"><li>• On top side, one <math>\times 16</math> interface</li></ul>
	EP2AGX65		<ul style="list-style-type: none"><li>• On bottom side, one <math>\times 16</math> interface</li><li>• On right side (no DQ pins on left side), one <math>\times 8</math> interface</li></ul>
Arria II GZ	EP2AGZ300	1,517	Four $\times 8$ interfaces or one $\times 72$ interface on each side
	EP2AGZ350		
	EP2AGZ225		
	EP2AGZ300	780	<ul style="list-style-type: none"><li>• On top side, three <math>\times 8</math> interfaces or one <math>\times 64</math> interface</li><li>• On bottom side, three <math>\times 8</math> interfaces or one <math>\times 64</math> interface</li><li>• No DQ pins on the left and right sides</li></ul>
Stratix III	EP3SL340	1,760	<ul style="list-style-type: none"><li>• Two <math>\times 72</math> interfaces on both top and bottom sides</li><li>• One <math>\times 72</math> interface on both right and left sides</li></ul>
	EP3SE50	484	<ul style="list-style-type: none"><li>• Two <math>\times 8</math> interfaces on both top and bottom sides</li><li>• Three <math>\times 8</math> interface on both right and left sides</li></ul>

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Stratix IV	EP4SGX290	1,932	<ul style="list-style-type: none"> <li>One <math>\times 72</math> interface on each side</li> </ul> <p>or</p>
	EP4SGX360		
	EP4SGX530		<ul style="list-style-type: none"> <li>One <math>\times 72</math> interface on each side and two additional <math>\times 72</math> wraparound interfaces, only if sharing DLL and PLL resources</li> </ul>
	EP4SE530	1,760	
	EP4SE820	<ul style="list-style-type: none"> <li>Three <math>\times 8</math> interfaces or one <math>\times 64</math> interface on both top and bottom sides</li> <li>On left side, one <math>\times 48</math> interface or two <math>\times 8</math> interfaces</li> <li>No DQ pins on the right side</li> </ul>	
	EP4SGX70		
	EP4SGX110		
	EP4SGX180		
	EP4SGX230		

### Maximum Number of DDR2 SDRAM Interfaces Supported per FPGA

The following table lists the maximum number of  $\times 8$  DDR2 SDRAM components that can be fitted in the smallest and biggest devices and pin packages assuming the device is blank.

Each interface of size  $n$ , where  $n$  is a multiple of 8, consists of:

- $n$  DQ pins (including ECC)
- $n/8$  DM pins
- $n/8$  DQS, DQSn pin pairs
- 18 address pins
- 7 command pins (CAS#, RAS#, WE#, CKE, ODT, and CS#)
- 1 CK, CK# pin pair up to every three  $\times 8$  DDR2 components

**Table 1-6: Maximum Number of DDR2 SDRAM Interfaces Supported per FPGA**

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GX	EP2AGX190	1,152	Four $\times 8$ interfaces or one $\times 72$ interface on each side (no DQ pins on left side)
	EP2AGX260		
	EP2AGX45	358	<ul style="list-style-type: none"> <li>One <math>\times 16</math> interface on both top and bottom sides</li> <li>On right side (no DQ pins on left side), one <math>\times 8</math> interface</li> </ul>
	EP2AGX65		

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GZ	EP2AGZ300	1,517	Four $\times 8$ interfaces or one $\times 72$ interface on each side
	EP2AGZ350		
	EP2AGZ225		
	EP2AGZ300	780	<ul style="list-style-type: none"> <li>Three <math>\times 8</math> interfaces or one <math>\times 64</math> interface on both top and bottom sides</li> <li>No DQ pins on the left and right sides</li> </ul>
Arria V	5AGXB1	1,517	<ul style="list-style-type: none"> <li>Two <math>\times 72</math> interfaces on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5AGXB3		
	5AGXB5		
	5AGXB7		
	5AGTD3		
	5AGTD7		
	5AGXA1	672	<ul style="list-style-type: none"> <li>One <math>\times 56</math> interface or two <math>\times 24</math> interfaces on both top and bottom sides</li> <li>One <math>\times 32</math> interface on the right side</li> <li>No DQ pins on the left side</li> </ul>
	5AGXA3		
	5AGXA5	672	<ul style="list-style-type: none"> <li>One <math>\times 56</math> interface or two <math>\times 24</math> interfaces on both top and bottom sides</li> <li>No DQ pins on the left side</li> </ul>
	5AGXA7		
Arria V GZ	5AGZE5	1,517	<ul style="list-style-type: none"> <li>Three <math>\times 72</math> interfaces on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5AGZE7		
	5AGZE1	780	<ul style="list-style-type: none"> <li>On top side, two <math>\times 8</math> interfaces</li> <li>On bottom side, four <math>\times 8</math> interfaces or one <math>\times 72</math> interface</li> <li>No DQ pins on left and right sides</li> </ul>
	5AGZE3		
Cyclone V	5CGTD9	1,152	<ul style="list-style-type: none"> <li>One <math>\times 72</math> interface or two <math>\times 32</math> interfaces on each of the top, bottom, and right sides</li> <li>No DQ pins on the left side</li> </ul>
	5CEA9		
	5CGXC9		
	5CEA7	484	<ul style="list-style-type: none"> <li>One <math>\times 48</math> interface or two <math>\times 16</math> interfaces on both top and bottom sides</li> <li>One <math>\times 8</math> interface on the right side</li> <li>No DQ pins on the left side</li> </ul>
	5CGTD7		
	5CGXC7		

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
MAX 10 FPGA	10M50D672	762	One x32 interface on the right side
	10M40D672		
	10M50D256	256	One x8 interface on the right side
	10M40D256		
Stratix III	10M25D256		
	10M16D256		
	EP3SL340	1,760	<ul style="list-style-type: none"> <li>Two <math>\times</math>72 interfaces on both top and bottom sides</li> <li>One <math>\times</math>72 interface on both right and left sides</li> </ul>
	EP3SE50	484	<ul style="list-style-type: none"> <li>Two <math>\times</math>8 interfaces on both top and bottom sides</li> <li>Three <math>\times</math>8 interfaces on both right and left sides</li> </ul>
Stratix IV	EP4SGX290	1,932	<ul style="list-style-type: none"> <li>One <math>\times</math>72 interface on each side</li> </ul>
	EP4SGX360		or
	EP4SGX530		
	EP4SE530	1,760	<ul style="list-style-type: none"> <li>One <math>\times</math>72 interface on each side and two additional <math>\times</math>72 wraparound interfaces only if sharing DLL and PLL resources</li> </ul>
Stratix V	EP4SE820		
	EP4SGX70	780	<ul style="list-style-type: none"> <li>Three <math>\times</math>8 interfaces or one <math>\times</math>64 interface on top and bottom sides</li> <li>On left side, one <math>\times</math>48 interface or two <math>\times</math>8 interfaces</li> <li>No DQ pins on the right side</li> </ul>
	EP4SGX110		
	EP4SGX180		
	EP4SGX230		
Stratix V	5SGXA5	1,932	<ul style="list-style-type: none"> <li>Three <math>\times</math>72 interfaces on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5SGXA7		
	5SGXA3	780	<ul style="list-style-type: none"> <li>On top side, two <math>\times</math>8 interfaces</li> </ul>
	5SGXA4		<ul style="list-style-type: none"> <li>On bottom side, four <math>\times</math>8 interfaces or one <math>\times</math>72 interface</li> <li>No DQ pins on left and right sides</li> </ul>

### Maximum Number of DDR3 SDRAM Interfaces Supported per FPGA

The following table lists the maximum number of  $\times$ 8 DDR3 SDRAM components that can be fitted in the smallest and biggest devices and pin packages assuming the device is blank.

Each interface of size  $n$ , where  $n$  is a multiple of 8, consists of:

- $n$  DQ pins (including ECC)
- $n/8$  DM pins
- $n/8$  DQS, DQSn pin pairs
- 17 address pins
- 7 command pins (CAS#, RAS#, WE#, CKE, ODT, reset, and CS#)
- 1 CK, CK# pin pair

**Table 1-7: Maximum Number of DDR3 SDRAM Interfaces Supported per FPGA**

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GX	EP2AGX190	1,152	<ul style="list-style-type: none"><li>• Four <math>\times 8</math> interfaces or one <math>\times 72</math> interface on each side</li><li>• No DQ pins on left side</li></ul>
	EP2AGX260		
	EP2AGX45	358	<ul style="list-style-type: none"><li>• One <math>\times 16</math> interface on both top and bottom sides</li><li>• On right side, one <math>\times 8</math> interface</li><li>• No DQ pins on left side</li></ul>
	EP2AGX65		
Arria II GZ	EP2AGZ300	1,517	Four $\times 8$ interfaces on each side
	EP2AGZ350		
	EP2AGZ225		
	EP2AGZ300	780	<ul style="list-style-type: none"><li>• Three <math>\times 8</math> interfaces on both top and bottom sides</li><li>• No DQ pins on left and right sides</li></ul>
Arria V	5AGXB1	1,517	<ul style="list-style-type: none"><li>• Two <math>\times 72</math> interfaces on both top and bottom sides</li><li>• No DQ pins on left and right sides</li></ul>
	5AGXB3		
	5AGXB5		
	5AGXB7		
	5AGTD3		
	5AGTD7		
	5AGXA1	672	<ul style="list-style-type: none"><li>• One <math>\times 56</math> interface or two <math>\times 24</math> interfaces on top and bottom sides</li><li>• One <math>\times 32</math> interface on the right side</li><li>• No DQ pins on the left side</li></ul>
	5AGXA3		
5AGXA5	5AGXA5	672	<ul style="list-style-type: none"><li>• One <math>\times 56</math> interface or two <math>\times 24</math> interfaces on both top and bottom sides</li><li>• No DQ pins on the left side</li></ul>
	5AGXA7		

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria V GZ	5AGZE5	1,517	<ul style="list-style-type: none"> <li>Two <math>\times 72</math> interfaces on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5AGZE7		
	5AGZE1	780	<ul style="list-style-type: none"> <li>On top side, four <math>\times 8</math> interfaces or one <math>\times 72</math> interface</li> <li>On bottom side, four <math>\times 8</math> interfaces or one <math>\times 72</math> interface</li> <li>No DQ pins on left and right sides</li> </ul>
	5AGZE3		
Cyclone V	5CGTD9	1,152	<ul style="list-style-type: none"> <li>One <math>\times 72</math> interface or two <math>\times 32</math> interfaces on each of the top, bottom, and right sides</li> <li>No DQ pins on the left side</li> </ul>
	5CEA9		
	5CGXC9		
	5CEA7	484	<ul style="list-style-type: none"> <li>One <math>\times 48</math> interface or two <math>\times 16</math> interfaces on both top and bottom sides</li> <li>One <math>\times 8</math> interface on the right side</li> <li>No DQ pins on the left side</li> </ul>
MAX 10 FPGA	5CGTD7		
	5CGXC7		
	10M50D672	762	One $\times 32$ interface on the right side
	10M40D672		
	10M50D256	256	One $\times 8$ interface on the right side
Stratix III	10M40D256		
	10M25D256		
	10M16D256		
	EP3SL340		<ul style="list-style-type: none"> <li>Two <math>\times 72</math> interfaces on both top and bottom sides</li> <li>One <math>\times 72</math> interface on both right and left sides</li> </ul>
Stratix III	EP3SE50	484	<ul style="list-style-type: none"> <li>Two <math>\times 8</math> interfaces on both top and bottom sides</li> <li>Three <math>\times 8</math> interfaces on both right and left sides</li> </ul>

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Stratix IV	EP4SGX290	1,932	<ul style="list-style-type: none"> <li>One <math>\times 72</math> interface on each side</li> </ul> <p>or</p>
	EP4SGX360		
	EP4SGX530		<ul style="list-style-type: none"> <li>One <math>\times 72</math> interface on each side and 2 additional <math>\times 72</math> wraparound interfaces only if sharing DLL and PLL resources</li> </ul>
	EP4SE530	1,760	
	EP4SE820	<ul style="list-style-type: none"> <li>Three <math>\times 8</math> interfaces or one <math>\times 64</math> interface on both top and bottom sides</li> <li>On left side, one <math>\times 48</math> interface or two <math>\times 8</math> interfaces</li> <li>No DQ pins on right side</li> </ul>	
	EP4SGX70		
	EP4SGX110		
	EP4SGX180		
Stratix V	EP4SGX230	780	<ul style="list-style-type: none"> <li>Two <math>\times 72</math> interfaces (800 MHz) on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5SGXA5	1,932	
	5SGXA7	<ul style="list-style-type: none"> <li>On top side, two <math>\times 8</math> interfaces</li> <li>On bottom side, four <math>\times 8</math> interfaces</li> <li>No DQ pins on left and right sides</li> </ul>	
	5SGXA3		780
	5SGXA4		

### Maximum Number of QDR II and QDR II+ SRAM Interfaces Supported per FPGA

The following table lists the maximum number of independent QDR II+ or QDR II SRAM interfaces that can be fitted in the smallest and biggest devices and pin packages assuming the device is blank.

One interface of  $\times 36$  consists of:

- 36 Q pins
- 36 D pins
- 1 K, K# pin pairs
- 1 CQ, CQ# pin pairs
- 19 address pins
- 4 BSWn pins
- WPSn, RPSn

One interface of  $\times 9$  consists of:

- 9 Q pins
- 9 D pins
- 1 K, K# pin pairs
- 1 CQ, CQ# pin pairs
- 21 address pins
- 1 BWSn pin
- WPSn, RPSn

**Table 1-8: Maximum Number of QDR II and QDR II+ SRAM Interfaces Supported per FPGA**

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Arria II GX	EP2AGX190	1,152	One $\times$ 36 interface and one $\times$ 9 interface one each side
	EP2AGX260		
	EP2AGX45	358	One $\times$ 9 interface on each side
	EP2AGX65		No DQ pins on left side
Arria II GZ	EP2AGZ300	1,517	<ul style="list-style-type: none"> <li>Two <math>\times</math>36 interfaces and one <math>\times</math>9 interface on both top and bottom sides</li> <li>Four <math>\times</math>9 interfaces on right and left sides</li> </ul>
	EP2AGZ350		
	EP2AGZ225		
	EP2AGZ300	780	<ul style="list-style-type: none"> <li>Three <math>\times</math>9 interfaces on both top and bottom sides</li> <li>No DQ pins on right and left sides</li> </ul>
	EP2AGZ350		
Arria V	5AGXB1	1,517	<ul style="list-style-type: none"> <li>Two <math>\times</math>36 interfaces on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5AGXB3		
	5AGXB5		
	5AGXB7		
	5AGTD3		
	5AGTD7		
	5AGXA1	672	<ul style="list-style-type: none"> <li>Two <math>\times</math>9 interfaces on both top and bottom sides</li> <li>One <math>\times</math>9 interface on the right side</li> <li>No DQ pins on the left side</li> </ul>
	5AGXA3		
	5AGXA5	672	<ul style="list-style-type: none"> <li>Two <math>\times</math>9 interfaces on both top and bottom sides</li> <li>No DQ pins on the left side</li> </ul>
	5AGXA7		
Arria V GZ	5AGZE5	1,517	<ul style="list-style-type: none"> <li>Two <math>\times</math>36 interfaces on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5AGZE7		
	5AGZE1	780	<ul style="list-style-type: none"> <li>On top side, one <math>\times</math>36 interface or three <math>\times</math>9 interfaces</li> <li>On bottom side, two <math>\times</math>9 interfaces</li> <li>No DQ pins on left and right sides</li> </ul>
	5AGZE3		

Device	Device Type	Package Pin Count	Maximum Number of Interfaces
Stratix III	EP3SL340	1,760	<ul style="list-style-type: none"> <li>Two <math>\times 36</math> interfaces and one <math>\times 9</math> interface on both top and bottom sides</li> <li>Five <math>\times 9</math> interfaces on both right and left sides</li> </ul>
	EP3SE50	484	<ul style="list-style-type: none"> <li>One <math>\times 9</math> interface on both top and bottom sides</li> <li>Two <math>\times 9</math> interfaces on both right and left sides</li> </ul>
	EP3SL50		
	EP3SL70		
Stratix IV	EP4SGX290	1,932	<ul style="list-style-type: none"> <li>Two <math>\times 36</math> interfaces on both top and bottom sides</li> <li>One <math>\times 36</math> interface on both right and left sides</li> </ul>
	EP4SGX360		
	EP4SGX530		
	EP4SE530	1,760	
	EP4SE820		
Stratix V	EP4SGX70	780	Two $\times 9$ interfaces on each side
	EP4SGX110		No DQ pins on right side
	EP4SGX180		
	EP4SGX230		
	5SGXA5	1,932	<ul style="list-style-type: none"> <li>Two <math>\times 36</math> interfaces on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5SGXA7		
	5SGXA3	780	<ul style="list-style-type: none"> <li>On top side, one <math>\times 36</math> interface or three <math>\times 9</math> interfaces</li> <li>On bottom side, two <math>\times 9</math> interfaces</li> <li>No DQ pins on left and right sides</li> </ul>
	5SGXA4		

### Maximum Number of RLDRAM II Interfaces Supported per FPGA

The following table lists the maximum number of independent RLDRAM II interfaces that can be fitted in the smallest and biggest devices and pin packages assuming the device is blank.

One common I/O  $\times 36$  interface consists of:

- 36 DQ
- 1 DM pin
- 2 DK, DK# pin pairs
- 2 QK, QK# pin pairs
- 1 CK, CK# pin pair
- 24 address pins
- 1 CS# pin
- 1 REF# pin
- 1 WE# pin

One common I/O  $\times 9$  interface consists of:

- 9 DQ
- 1 DM pins
- 1 DK, DK# pin pair
- 1 QK, QK# pin pair
- 1 CK, CK# pin pair
- 25 address pins
- 1 CS# pin
- 1 REF# pin
- 1 WE# pin

**Table 1-9: Maximum Number of RLDRAM II Interfaces Supported per FPGA**

Device	Device Type	Package Pin Count	Maximum Number of RLDRAM II CIO Interfaces
Arria II GZ	EP2AGZ300	1,517	Two $\times 36$ interfaces on each side
	EP2AGZ350		
	EP2AGZ225		
	EP2AGZ300	780	<ul style="list-style-type: none"> <li>• Three <math>\times 9</math> interfaces or one <math>\times 36</math> interface on both top and bottom sides</li> <li>• No DQ pins on the left and right sides</li> </ul>
	EP2AGZ350		
Arria V	5AGXB1	1,517	<ul style="list-style-type: none"> <li>• Two <math>\times 36</math> interfaces on both top and bottom sides</li> <li>• No DQ pins on left and right sides</li> </ul>
	5AGXB3		
	5AGXB5		
	5AGXB7		
	5AGTD3		
	5AGTD7		
	5AGXA1	672	<ul style="list-style-type: none"> <li>• One <math>\times 36</math> interface on both top and bottom sides</li> <li>• One <math>\times 18</math> interface on the right side</li> <li>• No DQ pins on the left side</li> </ul>
Arria V GZ	5AGXA3		
	5AGXA5	672	<ul style="list-style-type: none"> <li>• One <math>\times 36</math> interface on both top and bottom sides</li> <li>• No DQ pins on the left side</li> </ul>
	5AGXA7		
	5ZGZE5	1,517	<ul style="list-style-type: none"> <li>• Four <math>\times 36</math> interfaces on both top and bottom sides</li> <li>• No DQ pins on left and right sides</li> </ul>
	5ZGZE7		
	5AGZE1	780	<ul style="list-style-type: none"> <li>• On top side, three <math>\times 9</math> interfaces or two <math>\times 36</math> interfaces</li> <li>• On bottom side, two <math>\times 9</math> interfaces or one <math>\times 36</math> interfaces</li> <li>• No DQ pins on left and right sides</li> </ul>
	5AGZE3		

Device	Device Type	Package Pin Count	Maximum Number of RLDRAM II CIO Interfaces
Stratix III	EP3SL340	1,760	<ul style="list-style-type: none"> <li>Four <math>\times 36</math> components on both top and bottom sides</li> <li>Three <math>\times 36</math> interfaces on both right and left sides</li> </ul>
	EP3SE50	484	One $\times 9$ interface on both right and left sides
	EP3SL50		
	EP3SL70		
Stratix IV	EP4SGX290	1,932	<ul style="list-style-type: none"> <li>Three <math>\times 36</math> interfaces on both top and bottom sides</li> <li>Two <math>\times 36</math> interfaces on both right and left sides</li> </ul>
	EP4SGX360		
	EP4SGX530		
	EP4SE530	1,760	<ul style="list-style-type: none"> <li>Three <math>\times 36</math> interfaces on each side</li> </ul>
Stratix V	EP4SE820		
	EP4SGX70	780	One $\times 36$ interface on each side (no DQ pins on right side)
	EP4SGX110		
	EP4SGX180		
	EP4SGX230		
Stratix V	5SGXA5	1,932	<ul style="list-style-type: none"> <li>Four <math>\times 36</math> interfaces on both top and bottom sides</li> <li>No DQ pins on left and right sides</li> </ul>
	5SGXA7		
	5SGXA3	780	<ul style="list-style-type: none"> <li>On top side, two <math>\times 9</math> interfaces or one <math>\times 18</math> interfaces</li> <li>On bottom side, three <math>\times 9</math> interfaces or two <math>\times 36</math> interfaces</li> <li>No DQ pins on left and right sides</li> </ul>
	5SGXA4		

### Maximum Number of LPDDR2 SDRAM Interfaces Supported per FPGA

The following table lists the maximum number of x8 LPDDR2 SDRAM components that can fit in the smallest and largest devices and pin packages, assuming the device is blank.

Each interface of size  $n$ , where  $n$  is a multiple of 8, consists of:

- $n$  DQ pins (including ECC)
- $n/8$  DM pins
- $n/8$  DQS, DQSn pin pairs
- 10 address pins
- 2 command pins (CKE and CSn)
- 1 CK, CK# pin pair up to every three x8 LPDDR2 components

**Table 1-10: Maximum Number of LPDDR2 SDRAM Interfaces Supported per FPGA**

Device	Device Type	Package Pin Count	Maximum Number of LPDDR2 SDRAM Interfaces
Arria V	5AGXB1	1,517	<ul style="list-style-type: none"> <li>One <math>\times 72</math> interface on both top and bottom sides</li> <li>No DQ pins on the left and right sides</li> </ul>
	5AGXB3		
	5AGXB5		
	5AGXB7		
	5AGTD3		
	5AGTD7		
	5AGXA1	672	<ul style="list-style-type: none"> <li>One <math>\times 64</math> interface or two <math>\times 24</math> interfaces on both top and bottom sides</li> <li>One <math>\times 32</math> interface on the right side</li> </ul>
	5AGXA3		
	5AGXA5	672	<ul style="list-style-type: none"> <li>One <math>\times 64</math> interface or two <math>\times 24</math> interfaces on both the top and bottom sides</li> <li>No DQ pins on the left side</li> </ul>
	5AGXA7		
Cyclone V	5CGTD9	1,152	<ul style="list-style-type: none"> <li>One <math>\times 72</math> interface or two <math>\times 32</math> interfaces on each of the top, bottom, and right sides</li> <li>No DQ pins on the left side</li> </ul>
	5CEA9		
	5CGXC9		
	5CEA7	484	<ul style="list-style-type: none"> <li>One <math>\times 48</math> interface or two <math>\times 16</math> interfaces on both the top and bottom sides</li> <li>One <math>\times 8</math> interface on the right side</li> <li>No DQ pins on the left side</li> </ul>
	5CGTD7		
MAX 10 FPGA	10M50D672	762	One $\times 16$ interface on the right side
	10M40D672		
	10M50D256	256	One $\times 16$ interface on the right side
	10M40D256		
	10M25D256		
	10M16D256		

## OCT Support

If the memory interface uses any FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design, you need a calibration block for the OCT circuitry. This calibration block is not required to be within the same bank or side of the device as the memory interface pins. However, the block requires a pair of  $R_{UP}$  and  $R_{DN}$  or  $R_{ZQ}$  pins that must be placed within an I/O bank that has the same VCCIO voltage as the VCCIO voltage of the I/O pins that use the OCT calibration block.

The  $R_{ZQ}$  pin in Arria 10, Stratix V, Arria V, and Cyclone V devices can be used as a general purpose I/O pin when it is not used to support OCT, provided the signal conforms to the bank voltage requirements.

The  $R_{UP}$  and  $R_{DN}$  pins in Arria II GX, Arria II GZ, MAX 10, Stratix III, and Stratix IV devices are dual functional pins that can also be used as DQ and DQS pins in when they are not used to support OCT, giving the following impacts on your DQS groups:

- If the  $R_{UP}$  and  $R_{DN}$  pins are part of a  $\times 4$  DQS group, you cannot use that DQS group in  $\times 4$  mode.
- If the  $R_{UP}$  and  $R_{DN}$  pins are part of a  $\times 8$  DQS group, you can only use this group in  $\times 8$  mode if any of the following conditions apply:
  - You are not using DM or BWSn pins.
  - You are not using a  $\times 8$  or  $\times 9$  QDR II SRAM device, as the  $R_{UP}$  and  $R_{DN}$  pins may have dual purpose function as the CQn pins. In this case, pick different pin locations for  $R_{UP}$  and  $R_{DN}$  pins, to avoid conflict with memory interface pin placement. You have the choice of placing the  $R_{UP}$  and  $R_{DN}$  pins in the same bank as the write data pin group or address and command pin group.
  - You are not using complementary or differential DQS pins.

**Note:** The Altera external memory interface IP does not support  $\times 8$  QDR II SRAM devices in the Quartus Prime software.

A DQS/DQ  $\times 8/\times 9$  group in Arria II GZ, Stratix III, and Stratix IV devices comprises 12 pins. A typical  $\times 8$  memory interface consists of one DQS, one DM, and eight DQ pins which add up to 10 pins. If you choose your pin assignment carefully, you can use the two extra pins for  $R_{UP}$  and  $R_{DN}$ . However, if you are using differential DQS, you do not have enough pins for  $R_{UP}$  and  $R_{DN}$  as you only have one pin leftover. In this case, as you do not have to put the OCT calibration block with the DQS or DQ pins, you can pick different locations for the  $R_{UP}$  and  $R_{DN}$  pins. As an example, you can place it in the I/O bank that contains the address and command pins, as this I/O bank has the same VCCIO voltage as the I/O bank containing the DQS and DQ pins.

There is no restriction when using  $\times 16/\times 18$  or  $\times 32/\times 36$  DQS groups that include the  $\times 4$  groups when pin members are used as  $R_{UP}$  and  $R_{DN}$  pins, as there are enough extra pins that can be used as DQS or DQ pins.

You must pick your DQS and DQ pins manually for the  $\times 8$ ,  $\times 9$ ,  $\times 16$  and  $\times 18$ , or  $\times 32$  and  $\times 36$  groups, if they are using  $R_{UP}$  and  $R_{DN}$  pins within the group. The Quartus Prime software might not place these pins optimally and might be unable to fit the design.

## Guidelines for Arria 10 External Memory Interface IP

The Arria 10 device contains up to two I/O columns that can be used by external memory interfaces. The Arria 10 I/O subsystem resides in the I/O columns. Each column contains multiple I/O banks, each of which consists of four I/O lanes. An I/O lane is a group of twelve I/O ports.

The I/O column, I/O bank, I/O lane, adjacent I/O bank, and pairing pin for every physical I/O pin can be uniquely identified using the Bank Number and Index within I/O Bank values which are defined in each Arria 10 device pin-out file.

- The numeric component of the Bank Number value identifies the I/O column, while the letter represents the I/O bank.
- The Index within I/O Bank value falls within one of the following ranges: 0 to 11, 12 to 23, 24 to 35, or 36 to 47, and represents I/O lanes 1, 2, 3, and 4, respectively.
- The adjacent I/O bank is defined as the I/O bank with same column number but the letter is either before or after the respective I/O bank letter in the A-Z system.
- The pairing pin for an I/O pin is located in the same I/O bank. You can identify the pairing pin by adding one to its Index within I/O Bank number (if it is an even number), or by subtracting one from its Index within I/O Bank number (if it is an odd number).

For example, a physical pin with a Bank Number of 2K and Index within I/O Bank of 22, indicates that the pin resides in I/O lane 2, in I/O bank 2K, in column 2. The adjacent I/O banks are 2J and 2L. The pairing pin for this physical pin is the pin with an Index within I/O Bank of 23 and Bank Number of 2K.

#### Related Information

##### [Restrictions on I/O Bank Usage for Arria 10 EMIF IP with HPS](#)

## General Pin-Out Guidelines for Arria 10 EMIF IP

You should follow the recommended guidelines when performing pin placement for all external memory interface pins targeting Arria 10 devices, whether you are using the Altera hard memory controller or your own solution.

If you are using the Altera hard memory controller, you should employ the relative pin locations defined in the `<variation_name>/altera_emif_arch_nf_version_number/<synth|sim>/<variation_name>_altera_emif_arch_nf_version_number_<unique ID>.readme.txt` file, which is generated with your IP.

**Note:** 1. EMIF IP pin-out requirements for the Arria 10 Hard Processor Subsystem (HPS) are more restrictive than for a non-HPS memory interface. The HPS EMIF IP defines a fixed pin-out in the Quartus Prime IP file (`.qip`), based on the IP configuration. When targeting Arria 10 HPS, you do not need to make location assignments for external memory interface pins. To obtain the HPS-specific external memory interface pin-out, compile the interface in the Quartus Prime software. Alternatively, consult the device handbook or the device pin-out files. For information on how you can customize the HPS EMIF pin-out, refer to *Restrictions on I/O Bank Usage for Arria 10 EMIF IP with HPS*.

2. Ping Pong PHY, PHY only, RLDRAMx, QDRx and LPDDR3 are not supported with HPS.

Observe the following general guidelines for placing pins for your Arria 10 external memory interface:

1. Ensure that the pins of a single external memory interface reside within a single I/O column.
2. An external memory interface can occupy one or more banks in the same I/O column. When an interface must occupy multiple banks, ensure that those banks are adjacent to one another.
3. Be aware that any pin in the same bank that is not used by an external memory interface is available for use as a general purpose I/O of compatible voltage and termination settings.
4. All address and command pins and their associated clock pins (CK and CK#) must reside within a single bank. The bank containing the address and command pins is identified as the address and command bank.
5. To minimize latency, when the interface uses more than two banks, you must select the center bank of the interface as the address and command bank.
6. The address and command pins and their associated clock pins in the address and command bank must follow a fixed pin-out scheme, as defined in the *Arria 10 External Memory Interface Pin Information File*, which is available on the Altera website.

You do not have to place every address and command pin manually. If you assign the location for one address and command pin, the Fitter automatically places the remaining address and command pins.

**Note:** The pin-out scheme is a hardware requirement that you must follow, and can vary according to the topology of the memory device. Some schemes require three lanes to implement address and command pins, while others require four lanes. To determine which scheme to follow, refer to the messages window during parameterization of your IP, or to the `<variation_name>/altera_emif_arch_nf_<version>/<synth|sim>/<variation_name>_altera_emif_arch_nf_<version>_<unique ID>_readme.txt` file after you have generated your IP.

7. An unused I/O lane in the address and command bank can serve to implement a data group, such as a x8 DQS group. The data group must be from the same controller as the address and command signals.
8. An I/O lane must not be used by both address and command pins and data pins.
9. Place read data groups according to the DQS grouping in the pin table and pin planner. Read data strobes (such as DQS and DQS#) or read clocks (such as CQ and CQ# / QK and QK#) must reside at physical pins capable of functioning as DQS/CQ and DQSn/CQn for a specific read data group size. You must place the associated read data pins (such as DQ and Q), within the same group.

**Note:** 1. Unlike other device families, there is no need to swap CQ/CQ# pins in certain QDR II and QDR II+ latency configurations.

2. QDR-IV requires that the polarity of all QKB/QKB# pins be swapped with respect to the polarity of the differential buffer inputs on the FPGA to ensure correct data capture on port B. All QKB pins on the memory device must be connected to the negative pins of the input buffers on the FPGA side, and all QKB# pins on the memory device must be connected to the positive pins of the input buffers on the FPGA side. Notice that the port names at the top-level of the IP already reflect this swap (that is, `mem_qkb` is assigned to the negative buffer leg, and `mem_qkb_n` is assigned to the positive buffer leg).

10. You can use a single I/O lane to implement two x4 DQS groups. The pin table specifies which pins within an I/O lane can be used for the two pairs of DQS and DQS# signals. In addition, for x4 DQS groups you must observe the following rules:

- There must be an even number of x4 groups in an external memory interface.
- DQS group 0 and DQS group 1 must be placed in the same I/O lane. Similarly, DQS group 2 and group 3 must be in the same I/O lane. Generally, DQS group X and DQS group X+1 must be in the same I/O lane, where X is an even number.

11. You should place the write data groups according to the DQS grouping in the pin table and pin planner. Output-only data clocks for QDR II, QDR II+, and QDR II+ Extreme, and RLDRAM 3 protocols need not be placed on DQS/DQSn pins, but must be placed on a differential pin pair. They must be placed in the same I/O bank as the corresponding DQS group.

**Note:** For RLDRAM 3, x36 device, `DQ[8:0]` and `DQ[26:18]` are referenced to `DK0/DK0#`, and `DQ[17:9]` and `DQ[35:27]` are referenced to `DK1/DK1#`.

12. For protocols and topologies with bidirectional data pins where a write data group consists of multiple read data groups, you should place the data groups and their respective write and read clock in the same bank to improve I/O timing.

You do not need to specify the location of every data pin manually. If you assign the location for the read capture strobe/clock pin pairs, the Fitter will automatically place the remaining data pins.

13. Ensure that DM/BWS pins are paired with a write data pin by placing one in an I/O pin and another in the pairing pin for that I/O pin. It is recommended—though not required—that you follow the same rule for DBI pins, so that at a later date you have the freedom to repurpose the pin as DM.

**Note:** 1. x4 mode does not support DM/DBI, or Arria 10 EMIF IP for HPS.

2. If you are using an Arria 10 EMIF IP-based RLDRAM II or RLDRAM 3 external memory interface, you should ensure that all the pins in a DQS group (that is, `DQ`, `DM`, `DK`, and `QK`) are placed in the same I/O bank. This requirement facilitates timing closure and is necessary for successful compilation of your design.

## Multiple Interfaces in the Same I/O Column

To place multiple interfaces in the same I/O column, you must ensure that the global reset signals (`global_reset_n`) for each individual interface all come from the same input pin or signal.

## I/O Banks Selection

- For each memory interface, select consecutive I/O banks.
- A memory interface can only span across I/O banks in the same I/O column.
- Because I/O bank 2A is also employed for configuration-related operations, you can use it to construct external memory interfaces only when the following conditions are met:
  - The pins required for configuration related use (such as configuration bus for Fast Passive Parallel mode or control signals for Partial Reconfiguration) are never shared with pins selected for EMIF use, even after configuration is complete.
  - The I/O voltages are compatible.
  - The design has achieved a successful fit in the Quartus Prime software.

Refer to the Arria 10 Device Handbook and the *Configuration Function* column of the Pin-Out files for more information about pins and configuration modes.

- The number of I/O banks that you require depends on the memory interface width.
- The 3V I/O bank does not support dynamic OCT or calibrated OCT. To place a memory interface in a 3V I/O bank, ensure that calibrated OCT is disabled for the address/command signals, the memory clock signals, and the data bus signals, during IP generation.
- In some device packages, the number of I/O pins in some LVDS I/O banks is less than 48 pins.

## Address/Command Pins Location

- All address/command pins for a controller must be in a single I/O bank.
- If your interface uses multiple I/O banks, the address/command pins must use the middle bank. If the number of banks used by the interface is even, any of the two middle I/O banks can be used for address/command pins.
- Address/command pins and data pins cannot share an I/O lane but can share an I/O bank.
- The address/command pin locations for the soft and hard memory controllers are predefined. In the *External Memory Interface Pin Information for Devices* spreadsheet, each index in the "Index within I/O bank" column denotes a dedicated address/command pin function for a given protocol. The index number of the pin specifies to which I/O lane the pin belongs:
  - I/O lane 0—Pins with index 0 to 11
  - I/O lane 1—Pins with index 12 to 23
  - I/O lane 2—Pins with index 24 to 35
  - I/O lane 3—Pins with index 36 to 47
- For memory topologies and protocols that require only three I/O lanes for the address/command pins, use I/O lanes 0, 1, and 2.
- Unused address/command pins in an I/O lane can be used as general-purpose I/O pins.

## CK Pins Assignment

Assign the clock pin (CK pin) according to the number of I/O banks in an interface:

- The number of I/O banks is odd—assign one CK pin to the middle I/O bank.
- The number of I/O banks is even—assign the CK pin to any one of the middle two I/O banks.

Although the Fitter can automatically select the required I/O banks, Altera recommends that you make the selection manually to reduce the pre-fit run time.

## PLL Reference Clock Pin Placement

Place the PLL reference clock pin in the address/command bank. Other I/O banks may not have free pins that you can use as the PLL reference clock pin:

- If you are sharing the PLL reference clock pin between several interfaces, the I/O banks must be consecutive.

The Arria 10 External Memory Interface IP does not support PLL cascading.

## RZQ Pin Placement

You may place the RZQ pin in any I/O bank in an I/O column with the correct VCCIO and VCCPT for the memory interface I/O standard in use.

## DQ and DQS Pins Assignment

Altera recommends that you assign the DQS pins to the remaining I/O lanes in the I/O banks as required:

- Constrain the DQ and DQS signals of the same DQS group to the same I/O lane.
- DQ signals from two different DQS groups cannot be constrained to the same I/O lane.

If you do not specify the DQS pins assignment, the Fitter will automatically select the DQS pins.

## Sharing an I/O Bank Across Multiple Interfaces

If you are sharing an I/O bank across multiple external memory interfaces, follow these guidelines:

- The interfaces must use the same protocol, voltage, data rate, frequency, and PLL reference clock.
- You cannot use an I/O bank as the address/command bank for more than one interface. The memory controller and sequencer cannot be shared.
- You cannot share an I/O lane. There is only one DQS input per I/O lane, and an I/O lane can only connect to one memory controller.

## Ping Pong PHY Implementation

The Ping Pong PHY feature instantiates two hard memory controllers—one for the primary interface and one for the secondary interface. The hard memory controller I/O bank of the primary interface is used for address and command and is always adjacent and above the hard memory controller I/O bank of the secondary interface. All four lanes of the primary hard memory controller I/O bank are used for address and command.

When you use Ping Pong PHY, the EMIF IP exposes two independent Avalon-MM interfaces to user logic; these interfaces correspond to the two hard memory controllers inside the interface. Each Avalon-MM interface has its own set of clock and reset signals. Refer to *Qsys Interfaces* for more information on the additional signals exposed by Ping Pong PHY interfaces.

For more information on Ping Pong PHY in Arria 10, refer to *Functional Description—Arria 10 EMIF*, in this handbook. For pin allocation information for Arria 10 devices, refer to *External Memory Interface Pin Information for Arria 10 Devices* on the Altera web site.

## Additional Requirements for DDR3 and DDR4 Ping-Pong PHY Interfaces

If you are using Ping Pong PHY with a DDR3 or DDR4 external memory interface on an Arria 10 device, follow these guidelines:

- The address and command I/O bank must not contain any DQS group.
- I/O banks that are above the address and command I/O bank must contain only data pins of the primary interface—that is, the interface with the lower DQS group indices.
- The I/O bank immediately below the address and command I/O bank must contain at least one DQS group of the secondary interface—that is, the interface with the higher DQS group indices. This I/O bank can, but is not required to, contain DQS groups of the primary interface.
- I/O banks that are two or more banks below the address and command I/O bank must contain only data pins of the secondary interface.

### Related Information

- [Pin-Out Files for Altera Devices](#)
- [Functional Description—Arria 10 EMIF](#)
- [External Memory Interface Pin Information for Arria 10 Devices](#)
- [Restrictions on I/O Bank Usage for Arria 10 EMIF IP with HPS](#)

## Resource Sharing Guidelines for Arria 10 EMIF IP

In Arria 10, different external memory interfaces can share PLL reference clock pins, core clock networks, I/O banks, and hard Nios processors. Each I/O bank has DLL and PLL resources, therefore these do not need to be shared. The Fitter automatically merges DLL and PLL resources when a bank is shared by different external memory interfaces, and duplicates them for a multi-I/O-bank external memory interface.

### PLL Reference Clock Pin

To conserve pin usage and enable core clock network and I/O bank sharing, you can share a PLL reference clock pin between multiple external memory interfaces. Sharing of a PLL reference clock pin also implies sharing of the reference clock network.

Observe the following guidelines for sharing the PLL reference clock pin:

1. To share a PLL reference clock pin, connect the same signal to the `pll_ref_clk` port of multiple external memory interfaces in the RTL code.
2. Place related external memory interfaces in the same I/O column.
3. Place related external memory interfaces in adjacent I/O banks. If you leave an unused I/O bank between the I/O banks used by the external memory interfaces, that I/O bank cannot be used by any other external memory interface with a different PLL reference clock signal.

**Note:** The `pll_ref_clk` pin can be placed in the address and command I/O bank or in a data I/O bank, there is no impact on timing.

### Core Clock Network

To access all external memory interfaces synchronously and to reduce global clock network usage, you may share the same core clock network with other external memory interfaces.

Observe the following guidelines for sharing the core clock network:

1. To share a core clock network, connect the `clks_sharing_master_out` of the master to the `clks_sharing_slave_in` of all slaves in the RTL code.
2. Place related external memory interfaces in the same I/O column.
3. Related external memory interface must have the same rate, memory clock frequency, and PLL reference clock.

## I/O Bank

To reduce I/O bank utilization, you may share an I/O Bank with other external memory interfaces.

Observe the following guidelines for sharing an I/O Bank:

1. Related external memory interfaces must have the same protocol, rate, memory clock frequency, and PLL reference clock.
2. You cannot use a given I/O bank as the address and command bank for more than one external memory interface.
3. You cannot share an I/O lane between external memory interfaces, but an unused pin can serve as a general purpose I/O pin, of compatible voltage and termination standards.

## Hard Nios Processor

All external memory interfaces residing in the same I/O column will share the same hard Nios processor. The shared hard Nios processor calibrates the external memory interfaces serially.

## Reset Signal

When multiple external memory interfaces occupy the same I/O column, they must share the same IP reset signal.

# Guidelines for UniPHY-based External Memory Interface IP

Altera recommends that you place all the pins for one memory interface (attached to one controller) on the same side of the device. For projects where I/O availability is limited and you must spread the interface on two sides of the device, place all the input pins on one side and the output pins on an adjacent side of the device, along with their corresponding source-synchronous clock.

## General Pin-out Guidelines for UniPHY-based External Memory Interface IP

For best results in laying out your UniPHY-based external memory interface, you should observe the following guidelines.

**Note:** For a unidirectional data bus as in QDR II and QDR II+ SRAM interfaces, do not split a read data pin group or a write data pin group onto two sides. You should also not split the address and command group onto two sides either, especially when you are interfacing with QDR II and QDR II+ SRAM burst-length-of-two devices, where the address signals are double data rate. Failure to adhere to these rules might result in timing failure.

In addition, there are some exceptions for the following interfaces:

- $\times 36$  emulated QDR II and QDR II+ SRAM in Arria II, Stratix III, and Stratix IV devices.
- RLDRAM II and RLDRAM 3 CIO devices.
- QDR II/+ SDRAM burst-length-of-two devices.
- You must compile the design in the Quartus Prime software to ensure that you are not violating signal integrity and Quartus Prime placement rules, which is critical when you have transceivers in the same design.

The following are general guidelines for placing pins optimally for your memory interfaces:

1. For Arria II GZ, Arria V, Cyclone V, Stratix III, Stratix IV, and Stratix V designs, if you are using OCT, the RUP and RDN, or RZQ pins must be in any bank with the same I/O voltage as your memory interface signals and often use two DQS or DQ pins from a group. If you decide to place the RUP and RDN, or RZQ pins in a bank where the DQS and DQ groups are used, place these pins first and then determine how many DQ pins you have left, to find out if your data pins can fit in the remaining pins. Refer to *OCT Support for Arria II GX, Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices*.
2. Use the PLL that is on the same side of the memory interface. If the interface is spread out on two adjacent sides, you may use the PLL that is located on either adjacent side. You must use the dedicated input clock pin to that particular PLL as the reference clock for the PLL. The input of the memory interface PLL cannot come from the FPGA clock network.
3. The Altera IP uses the output of the memory interface PLL as the DLL input reference clock. Therefore, ensure you select a PLL that can directly feed a suitable DLL.

**Note:** Alternatively, you can use an external pin to feed into the DLL input reference clock. The available pins are also listed in the External Memory Interfaces chapter of the relevant device family handbook. You can also activate an unused PLL clock output, set it at the desired DLL frequency, and route it to a PLL dedicated output pin. Connect a trace on the PCB from this output pin to the DLL reference clock pin, but be sure to include any signal integrity requirements such as terminations.

4. Read data pins require the usage of DQS and DQ group pins to have access to the DLL control signals.

**Note:** In addition, QVLD pins in RLDRAM II and RLDRAM 3 DRAM, and QDR II+ SRAM must use DQS group pins, when the design uses the QVLD signal. None of the Altera IP uses QVLD pins as part of read capture, so theoretically you do not need to connect the QVLD pins if you are using the Altera solution. It is good to connect it anyway in case the Altera solution gets updated to use QVLD pins.

5. In differential clocking (DDR3/DDR2 SDRAM, RLDRAM II, and RLDRAM 3 interfaces), connect the positive leg of the read strobe or clock to a DQS pin, and the negative leg of the read strobe or clock to a DQSn pin. For QDR II or QDR II+ SRAM devices with 2.5 or 1.5 cycles of read latency, connect the CQ pin to a DQS pin, and the CQn pin to a CQn pin (and not the DQSn pin). For QDR II or QDR II+ SRAM devices with 2.0 cycles of read latency, connect the CQ pin to a CQn pin, and the CQn pin to a DQS pin.
6. Write data (if unidirectional) and data mask pins (DM or BWSn) pins must use DQS groups. While the DLL phase shift is not used, using DQS groups for write data minimizes skew, and must use the SW and TCCS timing analysis methodology.
7. Assign the write data strobe or write data clock (if unidirectional) in the corresponding DQS/DQSn pin with the write data groups that place in DQ pins (except in RLDRAM II and RLDRAM 3 CIO devices). Refer to the *Pin-out Rule Exceptions* for your memory interface protocol.

- Note:** When interfacing with a DDR, or DDR2, or DDR3 SDRAM without leveling, put the CK and CK# pairs in a single  $\times 4$  DQS group to minimize skew between clocks and maximize margin for the  $t_{DQSS}$ ,  $t_{DSS}$ , and  $t_{DSH}$  specifications from the memory devices.
8. Assign any address pins to any user I/O pin. To minimize skew within the address pin group, you should assign the address pins in the same bank or side of the device.
  9. Assign the command pins to any I/O pins and assign the pins in the same bank or device side as the other memory interface pins, especially address and memory clock pins. The memory device usually uses the same clock to register address and command signals.
    - In QDR II and QDR II+ SRAM interfaces where the memory clock also registers the write data, assign the address and command pins in the same I/O bank or same side as the write data pins, to minimize skew.
    - For more information about assigning memory clock pins for different device families and memory standards, refer to *Pin Connection Guidelines Tables*.

#### Related Information

- [Pin Connection Guidelines Tables](#) on page 1-46
- [Additional Guidelines for Arria V GZ and Stratix V Devices](#) on page 1-55
- [OCT Support](#) on page 1-30
- [Pin-out Rule Exceptions for  \$\times 36\$  Emulated QDR II and QDR II+ SRAM Interfaces in Arria II, Stratix III and Stratix IV Devices](#) on page 1-39
- [Pin-out Rule Exceptions for QDR II and QDR II+ SRAM Burst-length-of-two Interfaces](#) on page 1-46
- [Pin-out Rule Exceptions for RLDRAM II and RLDRAM 3 Interfaces](#) on page 1-44

## Pin-out Rule Exceptions for $\times 36$ Emulated QDR II and QDR II+ SRAM Interfaces in Arria II, Stratix III and Stratix IV Devices

A few packages in the Arria II, Arria V GZ, Stratix III, Stratix IV, and Stratix V device families do not offer any  $\times 32/\times 36$  DQS groups where one read clock or strobe is associated with 32 or 36 read data pins. This limitation exists in the following I/O banks:

- All I/O banks in U358- and F572-pin packages for all Arria II GX devices
- All I/O banks in F484-pin packages for all Stratix III devices
- All I/O banks in F780-pin packages for all Arria II GZ, Stratix III, and Stratix IV devices; top and side I/O banks in F780-pin packages for all Stratix V and Arria V GZ devices
- All I/O banks in F1152-pin packages for all Arria II GZ, Stratix III, and Stratix IV devices, except EP4SGX290, EP4SGX360, EP4SGX530, EPAGZ300, and EPAGZ350 devices
- Side I/O banks in F1517- and F1760-pin packages for all Stratix III devices
- All I/O banks in F1517-pin for EP4SGX180, EP4SGX230, EP4S40G2, EP4S40G5, EP4S100G2, EP4S100G5, and EPAGZ225 devices
- Side I/O banks in F1517-, F1760-, and F1932-pin packages for all Arria II GZ and Stratix IV devices

This limitation limits support for  $\times 36$  QDR II and QDR II+ SRAM devices. To support these memory devices, this following section describes how you can emulate the  $\times 32/\times 36$  DQS groups for these devices.

- The maximum frequency supported in  $\times 36$  QDR II and QDR II+ SRAM interfaces using  $\times 36$  emulation is lower than the maximum frequency when using a native  $\times 36$  DQS group.

**Note:** The F484-pin package in Stratix III devices cannot support  $\times 32/\times 36$  DQS group emulation, as it does not support  $\times 16/\times 18$  DQS groups.

To emulate a  $\times 32/\times 36$  DQS group, combine two  $\times 16/\times 18$  DQS groups together. For  $\times 36$  QDR II and QDR II+ SRAM interfaces, the 36-bit wide read data bus uses two  $\times 16/\times 18$  groups; the 36-bit wide write data uses another two  $\times 16/\times 18$  groups or four  $\times 8/\times 9$  groups. The CQ and CQn signals from the QDR II and QDR II+ SRAM device traces are then split on the board to connect to two pairs of CQ/CQn pins in the FPGA. You might then need to split the QVLD pins also (if you are connecting them). These connections are the only connections on the board that you need to change for this implementation. There is still only one pair of K and Kn connections on the board from the FPGA to the memory (see the following figure). Use an external termination for the CQ/CQn signals at the FPGA end. You can use the FPGA OCT features on the other QDR II interface signals with  $\times 36$  emulation. In addition, there may be extra assignments to be added with  $\times 36$  emulation.

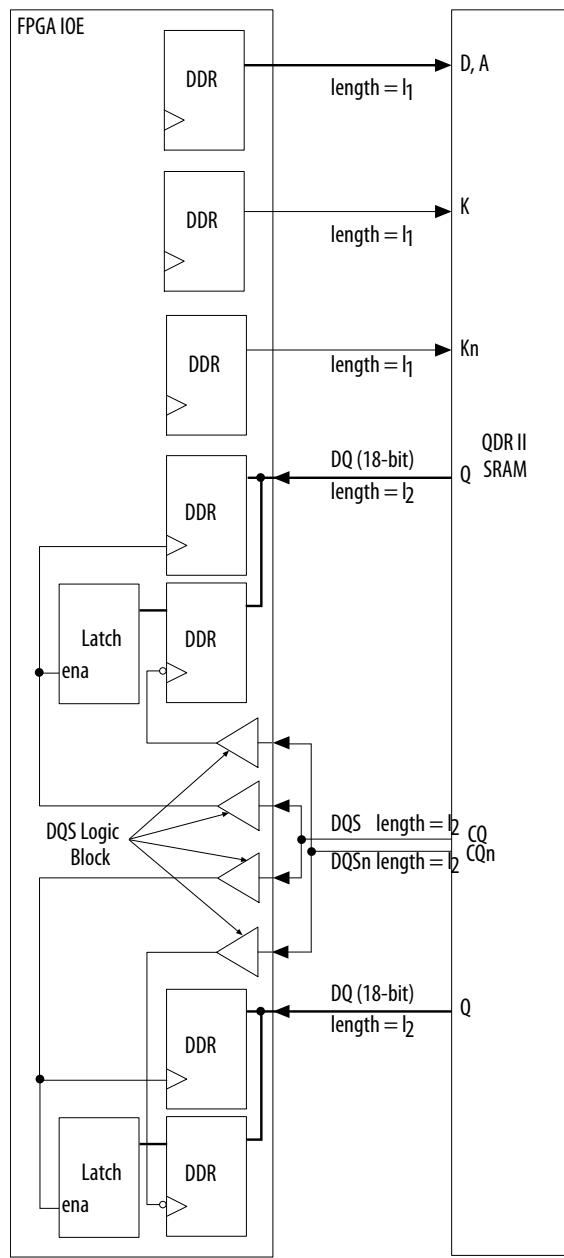
**Note:** Other QDR II and QDR II+ SRAM interface rules also apply for this implementation.

You may also combine four  $\times 9$  DQS groups (or two  $\times 9$  DQS groups and one  $\times 18$  group) on the same side of the device, if not the same I/O bank, to emulate a  $\times 36$  write data group, if you need to fit the QDR II interface in a particular side of the device that does not have enough  $\times 18$  DQS groups available for write data pins. Altera does not recommend using  $\times 4$  groups as the skew may be too large, as you need eight  $\times 4$  groups to emulate the  $\times 36$  write data bits.

You cannot combine four  $\times 9$  groups to create a  $\times 36$  read data group as the loading on the CQ pin is too large and hence the signal is degraded too much.

When splitting the CQ and CQn signals, the two trace lengths that go to the FPGA pins must be as short as possible to reduce reflection. These traces must also have the same trace delay from the FPGA pin to the Y or T junction on the board. The total trace delay from the memory device to each pin on the FPGA should match the Q trace delay (I2).

**Note:** You must match the trace delays. However, matching trace length is only an approximation to matching actual delay.

**Figure 1-9: Board Trace Connection for Emulated x36 QDR II and QDR II+ SRAM Interface**

### Timing Impact on x36 Emulation

With  $\times 36$  emulation, the CQ/CQn signals are split on the board, so these signals see two loads (to the two FPGA pins)—the DQ signals still only have one load. The difference in loading gives some slew rate degradation, and a later CQ/CQn arrival time at the FPGA pin.

The slew rate degradation factor is taken into account during timing analysis when you indicate in the UniPHY Preset Editor that you are using  $\times 36$  emulation mode. However, you must determine the difference in CQ/CQn arrival time as it is highly dependent on your board topology.

The slew rate degradation factor for  $\times 36$  emulation assumes that CQ/CQn has a slower slew rate than a regular  $\times 36$  interface. The slew rate degradation is assumed not to be more than 500 ps (from 10% to 90%

$V_{CCIO}$  swing). You may also modify your board termination resistor to improve the slew rate of the  $\times 36$ -emulated CQ/CQn signals. If your modified board does not have any slew rate degradation, you do not need to enable the  $\times 36$  emulation timing in the UniPHY-based controller parameter editor.

For more information about how to determine the CQ/CQn arrival time skew, refer to *Determining the CQ/CQn Arrival Time Skew*.

Because of this effect, the maximum frequency supported using  $\times 36$  emulation is lower than the maximum frequency supported using a native  $\times 36$  DQS group.

#### Related Information

[Determining the CQ/CQn Arrival Time Skew](#) on page 1-42

### Rules to Combine Groups

For devices that do not have four  $\times 16/\times 18$  groups in a single side of the device to form two  $\times 36$  groups for read and write data, you can form one  $\times 36$  group on one side of the device, and another  $\times 36$  group on the other side of the device. All the read groups have to be on the same edge (column I/O or row I/O) and all write groups have to be on the same type of edge (column I/O or row I/O), so you can have an interface with the read group in column I/O and the write group in row I/O. The only restriction is that you cannot combine an  $\times 18$  group from column I/O with an  $\times 18$  group from row IO to form a  $\times 36$ -emulated group.

For vertical migration with the  $\times 36$  emulation implementation, check if migration is possible and enable device migration in the Quartus Prime software.

**Note:** I/O bank 1C in both Stratix III and Stratix IV devices has dual-function configuration pins. Some of the DQS pins may not be available for memory interfaces if these are used for device configuration purposes.

Each side of the device in these packages has four remaining  $\times 8/\times 9$  groups. You can combine four of the remaining for the write side (only) if you want to keep the  $\times 36$  QDR II and QDR II+ SRAM interface on one side of the device, by changing the **Memory Interface Data Group** default assignment, from the default **18** to **9**.

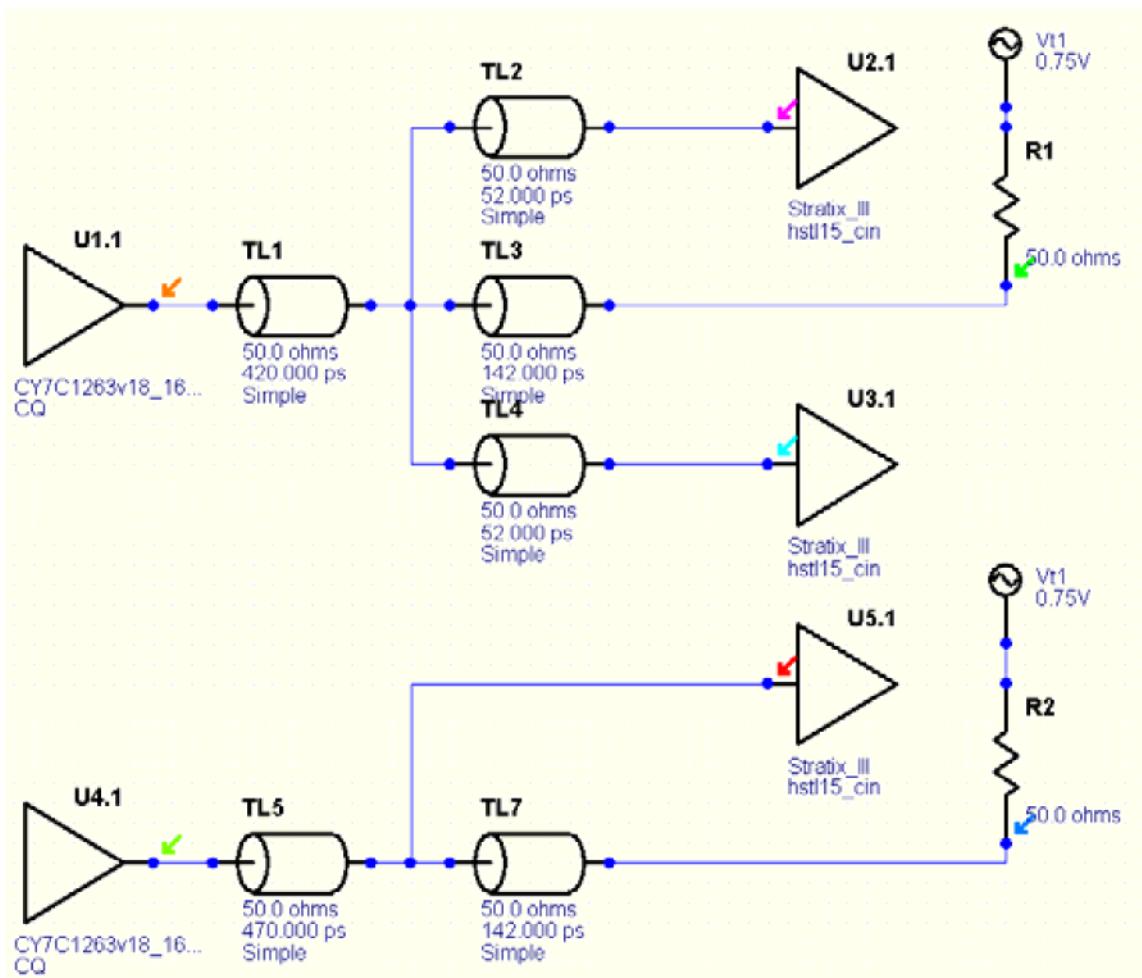
For more information about rules to combine groups for your target device, refer to the External Memory Interfaces chapter in the respective device handbooks.

### Determining the CQ/CQn Arrival Time Skew

Before compiling a design in the Quartus Prime software, you need to determine the CQ/CQn arrival time skew based on your board simulation. You then need to apply this skew in the **report\_timing.tcl** file of your QDR II and QDR II+ SRAM interface in the Quartus Prime software.

The following figure shows an example of a board topology comparing an emulated case where CQ is double-loaded and a non-emulated case where CQ only has a single load.

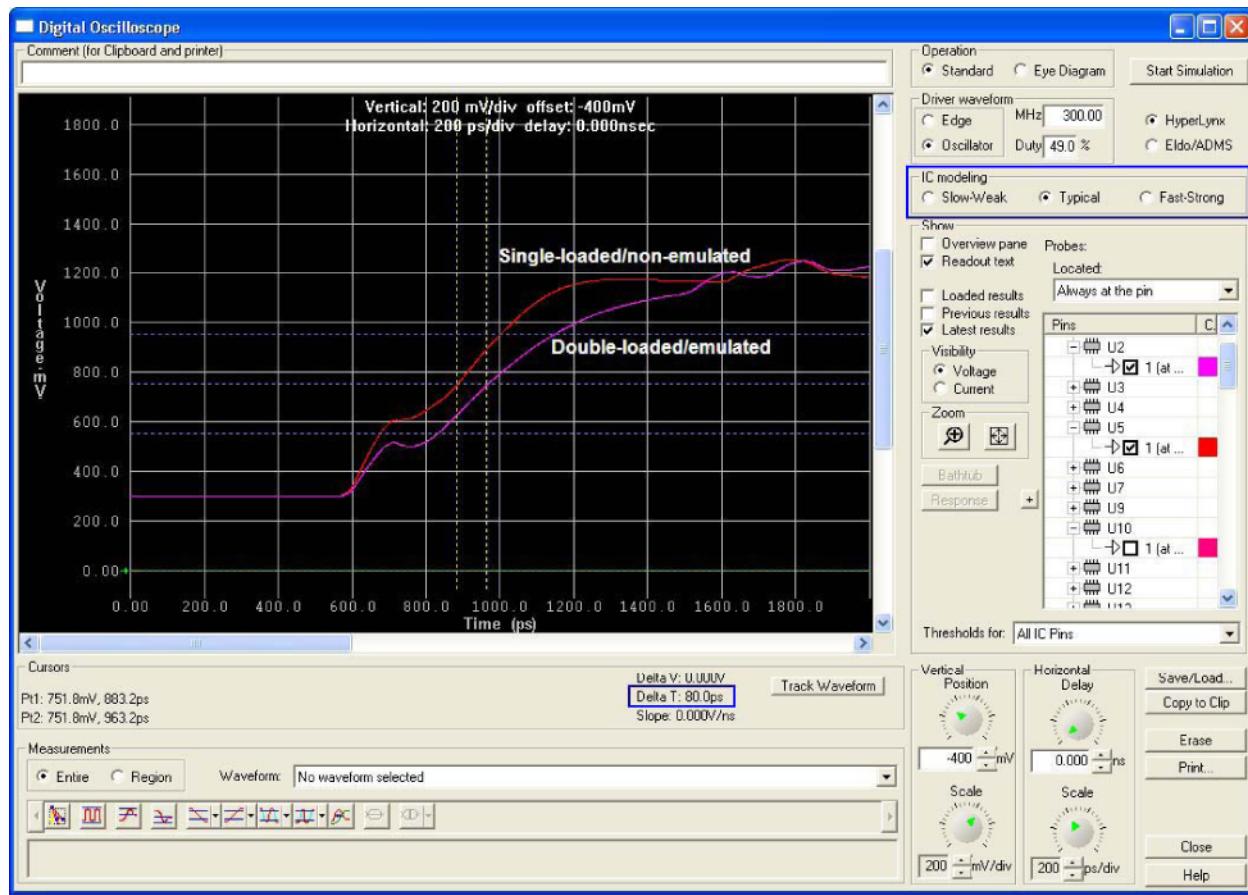
Figure 1-10: Board Simulation Topology Example



Run the simulation and look at the signal at the FPGA pin. The following figure shows an example of the simulation results from the preceding figure. As expected, the double-loaded emulated signal, in pink, arrives at the FPGA pin later than the single-loaded signal, in red. You then need to calculate the difference of this arrival time at VREF level (0.75 V in this case). Record the skew and rerun the simulation in the other two cases (slow-weak and fast-strong). To pick the largest and smallest skew to be included in Quartus Prime timing analysis, follow these steps:

1. Open the `<variation_name>_report_timing.tcl` and search for `tmin_additional_dqs_variation`.
2. Set the minimum skew value from your board simulation to `tmin_additional_dqs_variation`.
3. Set the maximum skew value from your board simulation to `tmax_additional_dqs_variation`.
4. Save the `.tcl` file.

Figure 1-11: Board Simulation Results



## Pin-out Rule Exceptions for RLDRAM II and RLDRAM 3 Interfaces

RLDRAM II and RLDRAM 3 CIO devices have one bidirectional bus for the data, but there are two different sets of clocks: one for read and one for write. As the QK and QK# already occupies the DQS and DQSn pins needed for read, placement of DK and DK# pins are restricted due to the limited number of pins in the FPGA. This limitations causes the exceptions to the previous rules, which are discussed below.

The address or command pins of RLDRAM II must be placed in a DQ-group because these pins are driven by the PHY clock. Half-rate RLDRAM II interfaces and full-rate RLDRAM 3 interfaces use the PHY clock for both the DQ pins and the address or command pins.

### Interfacing with x9 RLDRAM II CIO Devices

RLDRAM 3 devices do not have the x9 configuration.

RLDRAM II devices have the following pins:

- 2 pins for QK and QK# signals
- 9 DQ pins (in a  $\times 8/\times 9$  DQS group)
- 2 pins for DK and DK# signals
- 1 DM pin
- 14 pins total (15 if you have a QVLD)

In the FPGA, the  $\times 8/\times 9$  DQS group consists of 12 pins: 2 for the read clocks and 10 for the data. In this case, move the QVLD (if you want to keep this connected even though this is not used in the Altera memory interface solution) and the DK and DK# pins to the adjacent DQS group. If that group is in use, move to any available user I/O pins in the same I/O bank.

## Interfacing with $\times 18$ RLDRAM II and RLDRAM 3 CIO Devices

This topic describes interfacing with  $\times 18$  RLDRAM II and RLDRAM 3 devices.

RLDRAM II devices have the following pins:

- 4 pins for QK/QK# signals
- 18 DQ pins (in  $\times 8/\times 9$  DQS group)
- 2 pins for DK/DK# signals
- 1 DM pin
- 25 pins total (26 if you have a QVLD)

In the FPGA, you use two  $\times 8/\times 9$  DQS group totaling 24 pins: 4 for the read clocks and 18 for the read data.

Each  $\times 8/\times 9$  group has one DQ pin left over that can either use QVLD or DM, so one  $\times 8/\times 9$  group has the DM pin associated with that group and one  $\times 8/\times 9$  group has the QVLD pin associated with that group.

RLDRAM 3 devices have the following pins:

- 4 pins for QK/QK# signals
- 18 DQ pins (in  $\times 8/\times 9$  DQS group)
- 4 pins for DK/DK# signals
- 2 DM pins
- 28 pins total (29 if you have a QVLD)

In the FPGA, you use two  $\times 8/\times 9$  DQS group totaling 24 pins: 4 for the read clocks and 18 for the read data.

Each  $\times 8/\times 9$  group has one DQ pin left over that can either use QVLD or DM, so one  $\times 8/\times 9$  group has the DM pin associated with that group and one  $\times 8/\times 9$  group has the QVLD pin associated with that group.

## Interfacing with RLDRAM II and RLDRAM 3 $\times 36$ CIO Devices

This topic describes interfacing with RLDRAM II and RLDRAM 3  $\times 36$  CIO devices.

RLDRAM II devices have the following pins:

- 4 pins for QK/QK# signals
- 36 DQ pins (in  $\times 16/\times 18$  DQS group)
- 4 pins for DK/DK# signals
- 1 DM pins
- 46 pins total (47 if you have a QVLD)

In the FPGA, you use two  $\times 16/\times 18$  DQS groups totaling 48 pins: 4 for the read clocks and 36 for the read data. Configure each  $\times 16/\times 18$  DQS group to have:

- Two QK/QK# pins occupying the DQS/DQSn pins
- Pick two DQ pins in the  $\times 16/\times 18$  DQS groups that are DQS and DQSn pins in the  $\times 4$  or  $\times 8/\times 9$  DQS groups for the DK and DK# pins
- 18 DQ pins occupying the DQ pins
- There are two DQ pins leftover that you can use for QVLD or DM pins. Put the DM pin in the group associated with  $\text{DK}[1]$  and the QVLD pin in the group associated with  $\text{DK}[0]$ .
- Check that DM is associated with  $\text{DK}[1]$  for your chosen memory component.

RLDRAM 3 devices have the following pins:

- 8 pins for QK/QK# signals
- 36 DQ pins (in  $x8/x9$  DQS group)
- 4 pins for DK/DK# signals
- 2 DM pins
- 48 pins total (49 if you have a QVLD)

In the FPGA, you use four  $\times 8/\times 9$  DQS groups.

In addition, observe the following placement rules for RDRAM 3 interfaces:

For  $\times 18$  devices:

- Use two  $\times 8/\times 9$  DQS groups. Assign the QK/QK# pins and the DQ pins of the same read group to the same DQS group.
- DQ, DM, and DK/DK# pins belonging to the same write group should be assigned to the same I/O sub-bank, for timing closure.
- Whenever possible, assign CK/CK# pins to the same I/O sub-bank as the DK/DK# pins, to improve tCKDK timing.

For  $\times 36$  devices:

- Use four  $\times 8/\times 9$  DQS groups. Assign the QK/QK# pins and the DQ pins of the same read group to the same DQS group.
- DQ, DM, and DK/DK# pins belonging to the same write group should be assigned to the same I/O sub-bank, for timing closure.
- Whenever possible, assign CK/CK# pins to the same I/O sub-bank as the DK/DK# pins, to improve tCKDK timing.

## Pin-out Rule Exceptions for QDR II and QDR II+ SRAM Burst-length-of-two Interfaces

If you are using the QDR II and QDR II+ SRAM burst-length-of-two devices, you may want to place the address pins in a DQS group to minimize skew, because these pins are now double data rate too.

The address pins typically do not exceed 22 bits, so you may use one  $\times 18$  DQS groups or two  $\times 9$  DQS groups on the same side of the device, if not the same I/O bank. In Arria V GZ, Stratix III, Stratix IV, and Stratix V devices, one  $\times 18$  group typically has 22 DQ bits and 2 pins for DQS/DQSn pins, while one  $\times 9$  group typically has 10 DQ bits with 2 pins for DQS/DQSn pins. Using  $\times 4$  DQS groups should be a last resort.

## Pin Connection Guidelines Tables

The following table lists the FPGA pin utilization for DDR, DDR2, and DDR3 SDRAM without leveling interfaces.

**Table 1-11: FPGA Pin Utilization for DDR, DDR2, and DDR3 SDRAM without Leveling Interfaces**

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization			
		Arria II GX	Arria II GZ, Stratix III, and Stratix IV	Arria V, Cyclone V, and Stratix V	MAX 10 FPGA
Memory System Clock	CK and CK# <sup>(1)</sup> <sub>(2)</sub>	If you are using single-ended DQS signaling, place any unused DQ or DQS pins with DIFFOUT capability located in the same bank or on the same side as the data pins.  If you are using differential DQS signaling in UniPHY IP, place on DIFFOUT in the same single DQ group of adequate width to minimize skew.	If you are using single-ended DQS signaling, place any DIFFOUT pins in the same bank or on the same side as the data pins  If you are using differential DQS signaling in UniPHY IP, place any DIFFOUT pins in the same bank or on the same side as the data pins. If there are multiple CK/CK# pairs, place them on DIFFOUT in the same single DQ group of adequate width.  For example, DIMMs requiring three memory clock pin-pairs must use a $\times 4$ DQS group.	If you are using single-ended DQS signaling, place any unused DQ or DQS pins with DIFFOUT capability in the same bank or on the same side as the data pins.  If you are using differential DQS signaling, place any unused DQ or DQS pins with DIFFOUT capability for the mem_clk[n:0] and mem_clk_n[n:0] signals (where n >= 0). CK and CK# pins must use a pin pair that has DIFFOUT capability.  CK and CK# pins can be in the same group as other DQ or DQS pins. CK and CK# pins can be placed such that one signal of the differential pair is in a DQ group and the	Place any differential I/O pin pair (DIFFIO) in the same bank or on the same side as the data pins.

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization			
		Arria II GX	Arria II GZ, Stratix III, and Stratix IV	Arria V, Cyclone V, and Stratix V	MAX 10 FPGA
				other signal is not.  If there are multiple CK and CK# pin pairs, place them on DIFFOUT in the same single DQ group of adequate width.	
Clock Source	—	Dedicated PLL clock input pin with direct connection to the PLL (not using the global clock network).  For Arria II GX, Arria II GZ, Arria V GZ, Stratix III, Stratix IV and Stratix V Devices, also ensure that the PLL can supply the input reference clock to the DLL. Otherwise, refer to alternative DLL input reference clocks (see <i>General Pin-out Guidelines</i> ).			
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal.			
Data	DQ	DQ in the pin table, marked as Q in the Quartus Prime Pin Planner. Each DQ group has a common background color for all of the DQ and DM pins, associated with DQS (and DQSn) pins.			
Data mask	DM				
Data strobe	DQS or DQS and DQSn (DDR2 and DDR2 SDRAM only)	DQS (S in the Quartus Prime Pin Planner) for single-ended DQS signaling or DQS and DQSn (S and Sbar in the Quartus Prime Pin Planner) for differential DQS signaling. DDR2 supports either single-ended or differential DQS signaling. DDR3 SDRAM mandates differential DQS signaling.			

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization			
		Arria II GX	Arria II GZ, Stratix III, and Stratix IV	Arria V, Cyclone V, and Stratix V	MAX 10 FPGA
Address and command	A[], BA[], CAS#, CKE, CS#, ODT, RAS#, WE#, RESET#	Any user I/O pin. To minimize skew, you must place the address and command pins in the same bank or side of the device as the CK/CK# pins, DQ, DQS, or DM pins. The reset# signal is only available in DDR3 SDRAM interfaces. Altera devices use the SSTL-15 I/O standard on the RESET# signal to meet the voltage requirements of 1.5 V CMOS at the memory device. Altera recommends that you do not terminate the RESET# signal to VTT.	For Arria V, Cyclone V and Stratix V devices, you must place address and command pins in fully populated DQ/DQS groups with 12 available pins in the group.		
Notes to Table:					
<ol style="list-style-type: none"> <li>1. The first CK/CK# pair refers to <code>mem_clk[0]</code> or <code>mem_clk_n[0]</code> in the IP core.</li> <li>2. The restriction on the placement for the first CK/CK# pair is required because this placement allows the mimic path that the IP VT tracking uses to go through differential I/O buffers to mimic the differential DQS signals.</li> </ol>					

**Related Information**

[General Pin-out Guidelines for UniPHY-based External Memory Interface IP](#) on page 1-37

**DDR3 SDRAM With Leveling Interface Pin Utilization Applicable for Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices**

The following table lists the FPGA pin utilization for DDR3 SDRAM with leveling interfaces.

**Table 1-12: DDR3 SDRAM With Leveling Interface Pin Utilization Applicable for Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices**

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Data	DQ	DQ in the pin table, marked as Q in the Quartus Prime Pin Planner. Each DQ group has a common background color for all of the DQ and DM pins, associated with DQS (and DQSn) pins. The $\times 4$ DIMM has the following mapping between DQS and DQ pins: <ul style="list-style-type: none"> <li>• DQS[0] maps to DQ[3:0]</li> <li>• DQS[9] maps to DQ[7:4]</li> <li>• DQS[1] maps to DQ[11:8]</li> <li>• DQS[10] maps to DQ[15:12]</li> </ul> The DQS pin index in other DIMM configurations typically increases sequentially with the DQ pin index (DQS[0]: DQ[3:0]; DQS[1]: DQ[7:4]; DQS[2]: DQ[11:8]). In this DIMM configuration, the DQS pins are indicated this way to ensure pin out is compatible with both $\times 4$ and $\times 8$ DIMMs.
Data Strobe	DQS and DQSn	DQS and DQSn (S and Sbar in the Quartus Prime Pin Planner)
Address and Command	A[], BA[], CAS#, CKE, CS#, ODT, RAS#, WE#,	Any user I/O pin. To minimize skew, you should place address and command pins in the same bank or side of the device as the following pins: CK/CK# pins, DQ, DQS, or DM pins.  For Arria V, Cyclone V and Stratix V devices, you must place address and command pins in fully populated DQ/DQS groups with 12 available pins in the group.
	RESET#	Altera recommends that you use the 1.5V CMOS I/O standard on the RESET# signal. If your board is already using the SSTL-15 I/O standard, you do not terminate the RESET# signal to VTT.

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Memory system clock	CK and CK#	<p>For controllers with UniPHY IP, you can assign the memory clock to any unused DIFF_OUT pins in the same bank or on the same side as the data pins. However, for Arria V GZ and Stratix V devices, place the memory clock pins to any unused DQ or DQS pins. Do not place the memory clock pins in the same DQ group as any other DQ or DQS pins.</p> <p>If there are multiple CK/CK# pin pairs using Arria V GZ or Stratix V devices, you must place them on DIFFOUT in the same single DQ groups of adequate width. For example, DIMMs requiring three memory clock pin-pairs must use a <math>\times 4</math> DQS group.</p> <p>Placing the multiple CK/CK# pin pairs on DIFFOUT in the same single DQ groups for Stratix III and Stratix IV devices improves timing.</p>
Clock Source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal.

## QDR II and QDR II+ SRAM Pin Utilization for Arria II, Arria V, Stratix III, Stratix IV, and Stratix V Devices

The following table lists the FPGA pin utilization for QDR II and QDR II+ SRAM interfaces.

**Table 1-13: QDR II and QDR II+ SRAM Pin Utilization for Arria II, Arria V, Stratix III, Stratix IV, and Stratix V Devices**

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Read Clock	CQ and CQ# <sup>(1)</sup>	<p>For QDR II SRAM devices with 1.5 or 2.5 cycles of read latency or QDR II+ SRAM devices with 2.5 cycles of read latency, connect CQ to DQS pin (S in the Quartus Prime Pin Planner), and CQn to CQn pin (Qbar in the Quartus Prime Pin Planner).</p> <p>For QDR II or QDR II+ SRAM devices with 2.0 cycles of read latency, connect CQ to CQn pin (Qbar in the Quartus Prime Pin Planner), and CQn to DQS pin (S in the Quartus Prime Pin Planner).</p> <p>Arria V devices do not use CQn. The CQ rising and falling edges are used to clock the read data, instead of separate CQ and CQn signals.</p>

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Read Data	Q	DQ pins (Q in the Quartus Prime Pin Planner). Ensure that you are using the DQ pins associated with the chosen read clock pins (DQS and CQn pins). QVLD pins are only available for QDR II+ SRAM devices and note that Altera IP does not use the QVLD pin.
Data Valid	QVLD	
Memory and Write Data Clock	K and K#	Differential or pseudo-differential DQ, DQS, or DQSn pins in or near the write data group.
Write Data	D	DQ pins. Ensure that you are using the DQ pins associated with the chosen memory and write data clock pins (DQS and DQS pins).
Byte Write Select	BWS#, NWS#	
Address and Command	A, WPS#, RPS#	<p>Any user I/O pin. To minimize skew, you should place address and command pins in the same bank or side of the device as the following pins: K and K# pins, DQ, DQS, BWS#, and NWS# pins. If you are using burst-length-of-two devices, place the address signals in a DQS group pin as these signals are now double data rate.</p> <p>For Arria V, Cyclone V and Stratix V devices, you must place address and command pins in fully populated DQ/DQS groups with 12 available pins in the group.</p>
Clock source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal

Note to table:

1. For Arria V designs with integer latency, connect the CQ# signal to the CQ/CQ# pins from the pin table and ignore the polarity in the Pin Planner. For Arria V designs with fractional latency, connect the CQ signal to the CQ/CQ# pins from the pin table.

## RLDRAM II CIO Pin Utilization for Arria II GZ, Arria V, Stratix III, Stratix IV, and Stratix V Devices

The following table lists the FPGA pin utilization for RLDRAm II CIO and RLDRAm 3 interfaces.

**Table 1-14: RLDRAm II CIO Pin Utilization for Arria II GZ, Arria V, Stratix III, Stratix IV, and Stratix V Devices and RLDRAm 3 Pin Utilization for Arria V GZ and Stratix V Devices**

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Read Clock	QK and QK# <sup>(1)</sup>	DQS and DQSn pins (S and Sbar in the Quartus Prime Pin Planner)

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Data	Q	DQ pins (Q in the Quartus Prime Pin Planner). Ensure that you are using the DQ pins associated with the chosen read clock pins (DQS and DQSn pins). Altera IP does not use the QVLD pin. You may leave this pin unconnected on your board. You may not be able to fit these pins in a DQS group. For more information about how to place these pins, refer to “Exceptions for RLDRAM II and RLDRAM 3 Interfaces” on page 3–34.
Write Data Clock	DK and DK#	DQ pins in the same DQS group as the read data (Q) pins or in adjacent DQS group or in the same bank as the address and command pins. For more information, refer to <i>Exceptions for RLDRAM II and RLDRAM 3 Interfaces</i> . DK/DK# must use differential output-capable pins.  For Nios-based configuration, the DK pins must be in a DQ group but the DK pins do not have to be in the same group as the data or QK pins.
Memory Clock	CK and CK#	Any differential output-capable pins.  For Arria V GZ and Stratix V devices, place any unused DQ or DQS pins with DIFFOUT capability. Place the memory clock pins either in the same bank as the DK or DK# pins to improve DK versus CK timing, or in the same bank as the address and command pins to improve address command timing. Do not place CK and CK# pins in the same DQ group as any other DQ or DQS pins.
Address and Command	A, BA, CS#, REF#, WE#	Any user I/O pins. To minimize skew, you should place address and command pins in the same bank or side of the device as the following pins: CK/CK# pins, DQ, DQS, and DM pins.  For Arria V, Cyclone V and Stratix V devices, you must place address and command pins in fully populated DQ/DQS groups with 12 available pins in the group.
Clock source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal

Note to Table:

1. For Arria V devices, refer to the pin table for the QK and QK# pins. Connect QK and QK# signals to the QK and QK# pins from the pin table and ignore the polarity in the Pin Planner.

**Related Information**

[Pin-out Rule Exceptions for RLDRAM II and RLDRAM 3 Interfaces](#) on page 1-44

**LPDDR2 Pin Utilization for Arria V, Cyclone V, and MAX 10 FPGA Devices**

The following table lists the FPGA pin utilization for LPDDR2 SDRAM.

**Table 1-15: LPDDR2 Pin Utilization for Arria V, Cyclone V, and MAX 10 FPGA Devices**

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Memory Clock	CK, CKn	Differential clock inputs. All double data rate (DDR) inputs are sampled on both positive and negative edges of the CK signal. Single data rate (SDR) inputs are sampled at the positive clock edge. Place any unused DQ or DQS pins with DIFFOUT capability for the mem_clk[n:0] and mem_clk_n[n:0] signals (where n>=0). Do not place CK and CK# pins in the same group as any other DQ or DQS pins. If there are multiple CK and CK# pin pairs, place them on DIFFOUT in the same single DQ group of adequate width.
Address and Command	CA0-CA9 CSn CKE	Unidirectional DDR command and address bus inputs. Chip Select: CSn is considered to be part of the command code. Clock Enable: CKE HIGH activates and CKE LOW deactivates internal clock signals and therefore device input buffers and output drivers. Place address and command pins in any DDR-capable I/O pin. To minimize skew, Altera recommends using address and command pins in the same bank or side of the device as the CK/CK#, DQ, DQS, or DM pins..  For Arria V, Cyclone V and Stratix V devices, you must place address and command pins in fully populated DQ/DQS groups with 12 available pins in the group.
Data	DQ0-DQ7 (x8) DQ0-DQ15 (x16) DQ0-DQ31 (x32)	Bidirectional data bus. Pins are used as data inputs and outputs. DQ in the pin table is marked as Q in the Pin Planner. Each DQ group has a common background color for all of the DQ and DM pins associated with DQS (and DQSn) pins. Place on DQ group pin marked Q in the Pin Planner.
Data Strobe	DQS, DQSn	Data Strobe. The data strobe is bidirectional (used for read and write data) and differential (DQS and DQSn). It is output with read data and input with write data. Place on DQS and DQSn (S and Sbar in the Pin Planner) for differential DQS signaling.

Interface Pin Description	Memory Device Pin Name	FPGA Pin Utilization
Data Mask	DM0 ( $\times 8$ ) DM0-DM1 ( $\times 16$ ) DM0-DM3 ( $\times 32$ )	Input Data Mask. DM is the input mask signal for write data. Input data is masked when DM is sampled HIGH coincident with that input data during a write access. DM is sampled on both edges of DQS. DQ in the pin table is marked as Q in the Pin Planner. Each DQ group has a common background color for all of the DQ and DM pins, associated with DQS (and DQSn) pins. Place on DQ group pin marked Q in the Pin Planner.
Clock Source	—	Dedicated PLL clock input pin with direct (not using a global clock net) connection to the PLL and optional DLL required by the interface.
Reset	—	Dedicated clock input pin to accommodate the high fan-out signal.

## Additional Guidelines for Arria V GZ and Stratix V Devices

This section provides guidelines for improving timing for Arria V GZ and Stratix V devices and the rules that you must follow to overcome timing failures.

### Performing Manual Pin Placement

The following table lists rules that you can follow to perform proper manual pin placement and avoid timing failures.

The rules are categorized as follows:

- **Mandatory**—This rule is mandatory and cannot be violated as it would result in a no-fit error.
- **Recommended**—This rule is recommended and if violated the implementation is legal but the timing is degraded.
- **Highly Recommended**—This rule is not mandatory but is highly recommended because disregarding this rule might result in timing violations.

**Table 1-16: Manual Pin Placement Rules**

Rules	Frequency	Device	Reason
<b>Mandatory</b>			
Must place all CK, CK#, address, control, and command pins of an interface in the same I/O sub-bank.	> 800 MHz	All	For optimum timing, clock and data output paths must share as much hardware as possible. For write data pins (for example, DQ/DQS), the best timing is achieved through the DQS Groups.

Rules	Frequency	Device	Reason
Must not split interface between top and bottom sides	Any	All	Because PLLs and DLLs on the top edge cannot access the bottom edge of a device and vice-versa.
Must not place pins from separate interfaces in the same I/O sub-banks unless the interfaces share PLL or DLL resources.	Any	All	All pins require access to the same leveling block.
Must not share the same PLL input reference clock unless the interfaces share PLL or DLL resources.	Any	All	Because sharing the same PLL input reference clock forces the same ff-PLL to be used. Each ff-PLL can drive only one PHY clock tree and interfaces not sharing a PLL cannot share a PHY clock tree.

### Recommended

Place all CK, CK#, address, control, and command pins of an interface in the same I/O sub-bank.	<800 MHz	All	Place all CK/CK#, address, control, and command pins in the same I/O sub-bank when address and command timing is critical. For optimum timing, clock and data output paths should share as much hardware as possible. For write data pins (for example, DQ/DQS), the best timing is achieved through the DQS Groups.
Avoid using I/Os at the device corners (for example, sub-bank "A").	Any	A7 <sup>(1)</sup>	The delay from the FPGA core fabric to the I/O periphery is higher toward the sub-banks in the corners. By not using I/Os at the device corners, you can improve core timing closure.
	$\geq 800$ MHz	All	Corner I/O pins use longer delays, therefore avoiding corner I/O pins is recommended for better memory clock performance.
Avoid straddling an interface across the center PLL.	Any	All	Straddling the center PLL causes timing degradation, because it increases the length of the PHY clock tree and increases jitter. By not straddling the center PLL, you can improve core timing closure.

Rules	Frequency	Device	Reason
Use the center PLL(f-PLL1) for a wide interface that must straddle across center PLL.	$\geq 800$ MHz	All	Using a non-center PLL results in driving a sub-bank in the opposite quadrant due to long PHY clock tree delay.
Place the DQS/DQS# pins such that all DQ groups of the same interface are next to each other and do not span across the center PLL.	Any	All	To ease core timing closure. If the pins are too far apart then the core logic is also placed apart which results in difficult timing closure.
Place CK, CK#, address, control, and command pins in the same quadrant as DQ groups for improved timing in general.	Any	All	

**Highly Recommended**

Place all CK, CK#, address, control, and command pins of an interface in the same I/O sub-bank.	$\geq 800$ MHz	All	For optimum timing, clock and data output paths should share as much hardware as possible. For write data pins (for example, DQ/ DQS), the best timing is achieved through the DQS Groups.
Use center PLL and ensure that the PLL input reference clock pin is placed at a location that can drive the center PLL.	$\geq 800$ MHz	All	Using a non-center PLL results in driving a sub-bank in the opposite quadrant due to long PHY clock tree delay.
If center PLL is not accessible, place pins in the same quadrant as the PLL.	$\geq 800$ MHz	All	

Note to Table:

1. This rule is currently applicable to A7 devices only. This rule might be applied to other devices in the future if they show the same failure.

**Additional Guidelines for Arria V ( Except Arria V GZ) Devices**

This section provides guidelines on how to improve timing for Arria V devices and the rules that you must follow to overcome timing failures.

**Performing Manual Pin Placement**

The following table lists rules you can follow to perform proper manual pin placement and avoid timing failures.

The rules are categorized as follows:

- **Mandatory**—This rule is mandatory and cannot be violated as it would result in a no-fit error.
- **Recommended**—This rule is recommended and if violated the implementation is legal but the timing is degraded.

**Table 1-17: Manual Pin Placement Rules for Arria V (Except Arria V GZ) Devices**

Rules	Frequency	Device	Reason
<b>Mandatory</b>			
Must place all CK, CK#, address, control, and command pins of an interface on the same device edge as the DQ groups.	All	All	For optimum timing, clock and data output ports must share as much hardware as possible.
Must not place pins from separate interfaces in the same I/O sub-banks unless the interfaces share PLL or DLL resources. To share resources, the interfaces must use the same memory protocol, frequency, controller rate, and phase requirements.	All	All	All pins require access to the same PLL/DLL block.
Must not split interface between top, bottom, and right sides.	All	All	PHYCLK network support interfaces at the same side of the I/O banks only. PHYCLK networks do not support split interface.
<b>Recommended</b>			
Place the DQS/DQS# pins such that all DQ groups of the same interface are next to each other and do not span across the center PLL.	All	All	To ease core timing closure. If the pins are too far apart then the core logic is also placed apart which results in difficult timing closure.
Place all pins for a memory interface in an I/O bank and use the nearest PLL to that I/O bank for the memory interface.	All	All	Improve timing performance by reducing the PHY clock tree delay.

**Note:** Not all hard memory controllers on a given device package necessarily have the same address widths; some hard memory controllers have 16-bit address capability, while others have only 15-bit addresses.

## Additional Guidelines for MAX 10 Devices

The following additional guidelines apply when you implement an external memory interface for a MAX 10 device.

### I/O Pins Not Available for DDR3 or LPDDR2 External Memory Interfaces (Preliminary)

The I/O pins named in the following table are not available for use when implementing a DDR3 or LPDDR2 external memory interface for a MAX 10 device.

	F256	U324	F484	F672
10M16	N16	R15	U21	—
	P16	P15	U22	—
	—	R18	M21	—
	—	P18	L22	—
	—	—	F21	—
	—	—	F20	—
	—	E16	E19	—
	—	D16	F18	—
10M25	N16	—	U21	—
	P16	—	U22	—
	—	—	M21	—
	—	—	L22	—
	—	—	F21	—
	—	—	F20	—
	—	—	E19	—
	—	—	F18	—
	—	—	F17	—
	—	—	E17	—
10M50	—	—	—	W23
	—	—	—	W24
	—	—	—	U25
	—	—	—	U24
	N16	—	U21	T24
	P16	—	U22	R25
	—	—	M21	R24
	—	—	L22	P25
	—	—	F21	K23
	—	—	F20	K24

	F256	U324	F484	F672
	—	—	E19	J23
	—	—	F18	H23
	—	—	F17	G23
	—	—	E17	F23
	—	—	—	G21
	—	—	—	G22

### Additional Restrictions on I/O Pin Availability

The following restrictions are in addition to those represented in the above table.

- When implementing a DDR3 or LPDDR2 external memory interface, you can use only 75 percent of the remaining I/O pins in banks 5 and 6 for normal I/O operations.
- When implementing a DDR2 external memory interface, 25 percent of the remaining I/O pins in banks 5 and 6 can be assigned only as input pins.

### MAX 10 Board Design Considerations

- For DDR2, DDR3, and LPDDR2 interfaces, the maximum board skew between pins must be lower than 40 ps. This guideline applies to all pins (address, command, clock, and data).
- To minimize unwanted inductance from the board via, Altera recommends that you keep the PCB via depth for  $V_{CCIO}$  banks below 49.5 mil.
- For devices with DDR3 interface implementation, onboard termination is required for the DQ, DQS, and address signals. Altera recommends that you use termination resistor value of  $80\ \Omega$  to  $V_{TT}$ .
- For the DQ, address, and command pins, keep the PCB trace routing length less than six inches for DDR3, or less than three inches for LPDDR2.

### Power Supply Variation for LPDDR2 Interfaces

For an LPDDR2 interface that targets 200 MHz, constrain the memory device I/O and core power supply variation to within  $\pm 3\%$ .

### Additional Guidelines for Cyclone V Devices

This topic provides guidelines for improving performance for Cyclone V devices.

### I/O Pins Connect to Ground for Hard Memory Interface Operation

According to the Cyclone V pin-out file, there are some general I/O pins that are connected to ground for hard memory interface operation. These I/O pins should be grounded to reduce crosstalk from neighboring I/O pins and to ensure the performance of the hard memory interface.

The grounded user I/O pins can also be used as regular I/O pins if you run short of available I/O pins; however, the hard memory interface performance will be reduced if these pins are not connected to ground.

### PLLs and Clock Networks

The exact number of clocks and PLLs required in your design depends greatly on the memory interface frequency, and on the IP that your design uses.

For example, you can build simple DDR slow-speed interfaces that typically require only two clocks: system and write. You can then use the rising and falling edges of these two clocks to derive four phases ( $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ , and  $270^\circ$ ). However, as clock speeds increase, the timing margin decreases and additional clocks are required, to optimize setup and hold and meet timing. Typically, at higher clock speeds, you need to have dedicated clocks for resynchronization, and address and command paths.

Altera memory interface IP uses one PLL, which generates the various clocks needed in the memory interface data path and controller, and provides the required phase shifts for the write clock and address and command clock. The PLL is instantiated when you generate the Altera memory IPs.

By default, the memory interface IP uses the PLL to generate the input reference clock for the DLL, available in all supported device families. This method eliminates the need of an extra pin for the DLL input reference clock.

The input reference clock to the DLL can come from certain input clock pins or clock output from certain PLLs.

**Note:** Altera recommends using integer PLLs for memory interfaces; handbook specifications are based on integer PLL implementations.

For the actual pins and PLLs connected to the DLLs, refer to the *External Memory Interfaces* chapter of the relevant device family handbook.

You must use the PLL located in the same device quadrant or side as the memory interface and the corresponding dedicated clock input pin for that PLL, to ensure optimal performance and accurate timing results from the Quartus Prime software.

The input clock to the PLL can fan out to logic other than the PHY, so long as the clock input pin to the PLL is a dedicated input clock path, and you ensure that the clock domain transfer between UniPHY and the core logic is clocked by the reference clock going into a global clock.

## Number of PLLs Available in Altera Device Families

The following table lists the number of PLLs available in Altera device families.

**Table 1-18: Number of PLLs Available in Altera Device Families**

Device Family	Enhanced PLLs Available
Arria II GX	4-6
Arria II GZ	3-8
Arria V	16-24
Arria V GZ (fPLL)	22-28
Cyclone V	4-8
MAX 10 FPGA	1-4
Stratix III	4-12
Stratix IV	3-12

Device Family	Enhanced PLLs Available
Stratix V (fPLL)	22-28
Note to Table:	
1. For more details, refer to the <i>Clock Networks and PLL</i> chapter of the respective device family handbook.	

## Number of Enhanced PLL Clock Outputs and Dedicated Clock Outputs Available in Altera Device Families

The following table lists the number of enhanced PLL clock outputs and dedicated clock outputs available in Altera device families.

**Table 1-19: Number of Enhanced PLL Clock Outputs and Dedicated Clock Outputs Available in Altera Device Families<sup>(1)</sup>**

Device Family	Number of Enhanced PLL Clock Outputs	Number Dedicated Clock Outputs
Arria II GX <sup>(2)</sup>	7 clock outputs each	1 single-ended or 1 differential pair  3 single-ended or 3 differential pair total <sup>(3)</sup>
Arria V	18 clock outputs each	4 single-ended or 2 single-ended and 1 differential pair
Stratix III	Left/right: 7 clock outputs Top/bottom: 10 clock outputs	Left/right: 2 single-ended or 1 differential pair  Top/bottom: 6 single-ended or 4 single-ended and 1 differential pair
Arria II GZ and Stratix IV	Left/right: 7 clock outputs Top/bottom: 10 clock outputs	Left/right: 2 single-ended or 1 differential pair  Top/bottom: 6 single-ended or 4 single-ended and 1 differential pair
Arria V GZ and Stratix V	18 clock outputs each	4 single-ended or 2 single-ended and 1 differential pair

Device Family	Number of Enhanced PLL Clock Outputs	Number Dedicated Clock Outputs
Notes to Table:		
1.	For more details, refer to the <i>Clock Networks and PLL</i> chapter of the respective device family handbook.	
2.	PLL_5 and PLL_6 of Arria II GX devices do not have dedicated clock outputs.	
3.	The same PLL clock outputs drives three single-ended or three differential I/O pairs, which are only supported in PLL_1 and PLL_3 of the EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices.	

## Number of Clock Networks Available in Altera Device Families

The following table lists the number of clock networks available in Altera device families.

**Table 1-20: Number of Clock Networks Available in Altera Device Families <sup>(1)</sup>**

Device Family	Global Clock Network	Regional Clock Network
Arria II GX	16	48
Arria II GZ	16	64–88
Arria V	16	88
Arria V GZ	16	92
Cyclone V	16	N/A
MAX 10 FPGA	10	
Stratix III	16	64–88
Stratix IV	16	64–88
Stratix V	16	92

Note to Table:

1. For more information on the number of available clock network resources per device quadrant to better understand the number of clock networks available for your interface, refer to the *Clock Networks and PLL* chapter of the respective device family handbook.

**Note:** You must decide whether you need to share clock networks, PLL clock outputs, or PLLs if you are implementing multiple memory interfaces.

## Clock Network Usage in UniPHY-based Memory Interfaces—DDR2 and DDR3 SDRAM <sup>(1)/(2)</sup>

The following table lists clock network usage in UniPHY-based memory interfaces for DDR2 and DDR3 protocols.

**Table 1-21: Clock Network Usage in UniPHY-based Memory Interfaces—DDR2 and DDR3 SDRAM**

Device	DDR3 SDRAM		DDR2 SDRAM	
	Half-Rate		Half-Rate	
	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock	Number of half-rate clock
Stratix III	3 global	1 global 1 regional	1 global 2 global	1 global 1 regional
Arria II GZ and Stratix IV	3 global	1 global 1 regional	1 regional 2 regional	1 global 1 regional
Arria V GZ and Stratix V	1 global 2 regional	2 global	1 regional 2 regional	2 global

Notes to Table:

1. There are two additional regional clocks, `p11_avl_clk` and `p11_config_clk` for DDR2 and DDR3 SDRAM with UniPHY memory interfaces.
2. In multiple interface designs with other IP, the clock network might need to be modified to get a design to fit. For more information, refer to the *Clock Networks and PLLs* chapter in the respective device handbooks.

## Clock Network Usage in UniPHY-based Memory Interfaces—RLDRAM II, and QDR II and QDR II+ SRAM

The following table lists clock network usage in UniPHY-based memory interfaces for RLDRAM II, QDR II, and QDR II+ protocols.

**Table 1-22: Clock Network Usage in UniPHY-based Memory Interfaces—RLDRAM II, and QDR II and QDR II+ SRAM**

Device	RLDRAM II			QDR II/QDR II+ SRAM		
	Half-Rate		Full-Rate	Half-Rate		Full-Rate
	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock
Arria II GX	—	—	—	2 global	2 global	4 global
Stratix III	2 regional	1 global 1 regional	1 global 2 regional	1 global 1 regional	2 regional	1 global 2 regional

Device	RLDRAM II			QDR II/QDR II+ SRAM		
	Half-Rate		Full-Rate	Half-Rate		Full-Rate
	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock	Number of full-rate clock	Number of half-rate clock	Number of full-rate clock
Arria II GZ and Stratix IV	2 regional	1 global 1 regional	1 global 2 regional	1 global 1 regional	2 regional	1 global 2 regional

**Note:** For more information about the clocks used in UniPHY-based memory standards, refer to the *Functional Description—UniPHY* chapter in volume 3 of the External Memory Interface Handbook.

#### Related Information

##### [Functional Description—UniPHY](#)

### PLL Usage for DDR, DDR2, and DDR3 SDRAM Without Leveling Interfaces

The following table lists PLL usage for DDR, DDR2, and DDR3 protocols without leveling interfaces.

**Table 1-23: PLL Usage for DDR, DDR2, and DDR3 SDRAM Without Leveling Interfaces**

Clock	Arria II GX Devices	Stratix III and Stratix IV Devices
C0	<ul style="list-style-type: none"> <li>phy_clk_1x in half-rate designs</li> <li>aux_half_rate_clk</li> <li>PLL scan_clk</li> </ul>	<ul style="list-style-type: none"> <li>phy_clk_1x in half-rate designs</li> <li>aux_half_rate_clk</li> <li>PLL scan_clk</li> </ul>
C1	<ul style="list-style-type: none"> <li>phy_clk_1x in full-rate designs</li> <li>aux_full_rate_clk</li> <li>mem_clk_2x to generate DQS and CK/CK# signals</li> <li>ac_clk_2x</li> <li>cs_n_clk_2x</li> </ul>	<ul style="list-style-type: none"> <li>mem_clk_2x</li> </ul>
C2	<ul style="list-style-type: none"> <li>Unused</li> </ul>	<ul style="list-style-type: none"> <li>phy_clk_1x in full-rate designs</li> <li>aux_full_rate_clk</li> </ul>
C3	<ul style="list-style-type: none"> <li>write_clk_2x (for DQ)</li> <li>ac_clk_2x</li> <li>cs_n_clk_2x</li> </ul>	<ul style="list-style-type: none"> <li>write_clk_2x</li> </ul>
C4	<ul style="list-style-type: none"> <li>resync_clk_2x</li> </ul>	<ul style="list-style-type: none"> <li>resync_clk_2x</li> </ul>
C5	<ul style="list-style-type: none"> <li>measure_clk_2x</li> </ul>	<ul style="list-style-type: none"> <li>measure_clk_1x</li> </ul>
C6	—	<ul style="list-style-type: none"> <li>ac_clk_1x</li> </ul>

## PLL Usage for DDR3 SDRAM With Leveling Interfaces

The following table lists PLL usage for DDR3 protocols with leveling interfaces.

**Table 1-24: PLL Usage for DDR3 SDRAM With Leveling Interfaces**

Clock	Stratix III and Stratix IV Devices
C0	<ul style="list-style-type: none"> <li>• phy_clk_1x in half-rate designs</li> <li>• aux_half_rate_clk</li> <li>• PLL scan_clk</li> </ul>
C1	<ul style="list-style-type: none"> <li>• mem_clk_2x</li> </ul>
C2	<ul style="list-style-type: none"> <li>• aux_full_rate_clk</li> </ul>
C3	<ul style="list-style-type: none"> <li>• write_clk_2x</li> </ul>
C4	<ul style="list-style-type: none"> <li>• resync_clk_2x</li> </ul>
C5	<ul style="list-style-type: none"> <li>• measure_clk_1x</li> </ul>
C6	<ul style="list-style-type: none"> <li>• ac_clk_1x</li> </ul>

## Using PLL Guidelines

When using PLL for external memory interfaces, you must consider the following guidelines:

- For the clock source, use the clock input pin specifically dedicated to the PLL that you want to use with your external memory interface. The input and output pins are only fully compensated when you use the dedicated PLL clock input pin. If the clock source for the PLL is not a dedicated clock input pin for the dedicated PLL, you would need an additional clock network to connect the clock source to the PLL block. Using additional clock network may increase clock jitter and degrade the timing margin.
- Pick a PLL and PLL input clock pin that are located on the same side of the device as the memory interface pins.
- Share the DLL and PLL static clocks for multiple memory interfaces provided the controllers are on the same or adjacent side of the device and run at the same memory clock frequency.
- If your design uses a dedicated PLL to only generate a DLL input reference clock, you must set the PLL mode to **No Compensation** in the Quartus Prime software to minimize the jitter, or the software forces this setting automatically. The PLL does not generate other output, so it does not need to compensate for any clock path.
- If your design cascades PLL, the source (upstream) PLL must have a low-bandwidth setting, while the destination (downstream) PLL must have a high-bandwidth setting to minimize jitter. Altera does not recommend using cascaded PLLs for external memory interfaces because your design gets accumulated jitters. The memory output clock may violate the memory device jitter specification.

- Use cascading PLLs at your own risk. For more information, refer to “PLL Cascading”.
- If you are using Arria II GX devices, for a single memory instance that spans two right-side quadrants, use a middle-side PLL as the source for that interface.
- If you are using Arria II GZ, Arria V GZ, Stratix III, Stratix IV, or Stratix V devices, for a single memory instance that spans two top or bottom quadrants, use a middle top or bottom PLL as the source for that interface. The ten dual regional clocks that the single interface requires must not block the design using the adjacent PLL (if available) for a second interface.

#### Related Information

[PLL Cascading](#) on page 1-67

## PLL Cascading

Arria II GZ PLLs, Stratix III PLLs, Stratix IV PLLs, Stratix V and Arria V GZ fractional PLLs (fPLLs), and the two middle PLLs in Arria II GX EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260 devices can be cascaded using either the global or regional clock trees, or the cascade path between two adjacent PLLs.

**Note:** Use cascading PLLs at your own risk. You should use faster memory devices to maximize timing margins.

The UniPHY IP supports PLL cascading using the cascade path without any additional timing derating when the bandwidth and compensation rules are followed. The timing constraints and analysis assume that there is no additional jitter due to PLL cascading when the upstream PLL uses no compensation and low bandwidth, and the downstream PLL uses no compensation and high bandwidth.

The UniPHY IP does not support PLL cascading using the global and regional clock networks. You can implement PLL cascading at your own risk without any additional guidance and specifications from Altera. The Quartus Prime software does issue a critical warning suggesting use of the cascade path to minimize jitter, but does not explicitly state that Altera does not support cascading using global and regional clock networks.

Some Arria II GX devices (EP2AGX95, EP2AGX125, EP2AGX190, and EP2AGX260) have direct cascade path for two middle right PLLs. Arria II GX PLLs have the same bandwidth options as Stratix IV GX left and right PLLs.

The Arria 10 External Memory Interface IP does not support PLL cascading.

## DLL

The Altera memory interface IP uses one DLL. The DLL is located at the corner of the device and can send the control signals to shift the DQS pins on its adjacent sides for Stratix-series devices, or DQS pins in any I/O banks in Arria II GX devices.

For example, the top-left DLL can shift DQS pins on the top side and left side of the device. The DLL generates the same phase shift resolution for both sides, but can generate different phase offset to the two different sides, if needed. Each DQS pin can be configured to use or ignore the phase offset generated by the DLL.

The DLL cannot generate two different phase offsets to the same side of the device. However, you can use two different DLLs to for this functionality.

DLL reference clocks must come from either dedicated clock input pins located on either side of the DLL or from specific PLL output clocks. Any clock running at the memory frequency is valid for the DLLs.

To minimize the number of clocks routed directly on the PCB, typically this reference clock is sourced from the memory controllers PLL. In general, DLLs can use the PLLs directly adjacent to them (corner PLLs when available) or the closest PLL located in the two sides adjacent to its location.

**Note:** By default, the DLL reference clock in Altera external memory IP is from a PLL output.

When designing for 780-pin packages with EP3SE80, EP3SE110, EP3SL150, EP4SE230, EP4SE360, EP4SGX180, and EP4SGX230 devices, the PLL to DLL reference clock connection is limited. DLL2 is isolated from a direct PLL connection and can only receive a reference clock externally from pins  $\text{CLK}[11:4]_P$  in EP3SE80, EP3SE110, EP3SL150, EP4SE230, and EP4SE360 devices. In EP4SGX180 and EP4SGX230 devices, DLL2 and DLL3 are not directly connected to PLL. DLL2 and DLL3 receive a reference clock externally from pins  $\text{CLK}[7:4]_P$  and  $\text{CLK}[15:12]_P$  respectively.

For more DLL information, refer to the respective device handbooks.

The DLL reference clock should be the same frequency as the memory interface, but the phase is not important.

The required DQS capture phase is optimally chosen based on operating frequency and external memory interface type (DDR, DDR2, DDR3 SDRAM, and QDR II SRAM, or RLDRAM II). As each DLL supports two possible phase offsets, two different memory interface types operating at the same frequency can easily share a single DLL. More may be possible, depending on the phase shift required.

Altera memory IP always specifies a default optimal phase setting, to override this setting, refer to *Implementing and Parameterizing Memory IP*.

When sharing DLLs, your memory interfaces must be of the same frequency. If the required phase shift is different amongst the multiple memory interfaces, you can use a different delay chain in the DQS logic block or use the DLL phase offset feature.

To simplify the interface to IP connections, multiple memory interfaces operating at the same frequency usually share the same system and static clocks as each other where possible. This sharing minimizes the number of dedicated clock nets required and reduces the number of different clock domains found within the same design.

As each DLL can directly drive four banks, but each PLL only has complete C (output) counter coverage of two banks (using dual regional networks), situations can occur where a second PLL operating at the same frequency is required. As cascaded PLLs increase jitter and reduce timing margin, you are advised to first ascertain if an alternative second DLL and PLL combination is not available and more optimal.

Select a DLL that is available for the side of the device where the memory interface resides. If you select a PLL or a PLL input clock reference pin that can also serve as the DLL input reference clock, you do not need an extra input pin for the DLL input reference clock.

#### Related Information

[Implementing and Parameterizing Memory IP](#) on page 7-1

## Other FPGA Resources

The Altera memory interface IP uses FPGA fabric, including registers and the Memory Block to implement the memory interface.

For resource utilization examples to ensure that you can fit your other modules in the device, refer to the “Resource Utilization” section in the *Introduction to UniPHY IP* chapter of the External Memory Interface Handbook.

One OCT calibration block is used if you are using the FPGA OCT feature in the memory interface. The OCT calibration block uses two pins (RUP and RDN), or single pin (RZQ) (“OCT Support for

Arria II GX, Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices”). You can select any of the available OCT calibration block as you do not need to place this block in the same bank or device side of your memory interface. The only requirement is that the I/O bank where you place the OCT calibration block uses the same VCCIO voltage as the memory interface. You can share multiple memory interfaces with the same OCT calibration block if the VCCIO voltage is the same.

#### Related Information

- [OCT Support](#) on page 1-30
- [Introduction to UniPHY IP](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"><li>• Modified <i>Data Strobe</i> and <i>Address</i> data in <i>UDIMM, RDIMM, and LRDIMM Pin Options</i> for <i>DDR4</i> table in <i>DDR, DDR2, DDR3, and DDR4 SDRAM DIMM Options</i>. Added notes to table.</li></ul>
November 2015	2015.11.02	<ul style="list-style-type: none"><li>• Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li><li>• Modified <i>I/O Banks Selection, PLL Reference Clock and RZQ Pins Placement</i>, and <i>Ping Pong PHY Implementation</i> sections in <i>General Pin-Out Guidelines for Arria 10 EMIF IP</i>.</li><li>• Added <i>Additional Requirements for DDR3 and DDR4 Ping-Pong PHY Interfaces</i> in <i>General Pin-Out Guidelines for Arria 10 EMIF IP</i>.</li><li>• Removed references to OCT Blocks from <i>Resource Sharing Guidelines for Arria 10 EMIF IP</i> section.</li><li>• Added LPDDR3.</li></ul>
May 2015	2015.05.04	<ul style="list-style-type: none"><li>• Removed the F672 package of the 10M25 device.</li><li>• Updated the additional guidelines for MAX 10 devices to improve clarity.</li><li>• Added related information link to the <i>MAX 10 FPGA Signal Integrity Design Guidelines</i> for the <i>Additional Guidelines for MAX 10 Devices</i> topic.</li></ul>

Date	Version	Changes
December 2014	2014.12.15	<ul style="list-style-type: none"> <li>• <i>General Pin-Out Guidelines for Arria 10 EMIF IP</i> section:           <ul style="list-style-type: none"> <li>• Added note to step 10.</li> <li>• Removed steps 13 and 14.</li> <li>• Added a bullet point to <i>Address/Command Pins Location</i>.</li> <li>• Added <i>Ping Pong PHY Implementation</i></li> <li>• Added parenthetical comment to fifth bullet point in <i>I/O Banks Selection</i></li> <li>• Added note following the procedure, advising that all pins in a DQS group should reside in the same I/O bank, for RLDRAM II and RLDRAM 3 interfaces.</li> </ul> </li> <li>• Added <i>QDR IV SRAM Clock Signals, QDR IV SRAM Commands and Addresses, AP, and AINV Signals</i>, and <i>QDR IV SRAM Data, DINV, and QVLD Signals</i> topics.</li> <li>• Added note to <i>Estimating Pin Requirements</i> section.</li> <li>• <i>DDR, DDR2, DDR3, and DDR4 SDRAM DIMM Options</i> section:           <ul style="list-style-type: none"> <li>• Added <i>UDIMM, RDIMM, and LRDIMM Pin Options for DDR4</i> table.</li> <li>• Changed notes to <i>LRDIMM Pin Options for DDR, DDR2, and DDR3</i> table.</li> <li>• Removed reference to Chip ID pin.</li> </ul> </li> </ul>

Date	Version	Changes
August 2014	2014.08.15	<ul style="list-style-type: none"><li>• Made several changes to <i>Pin Counts for Various Example Memory Interfaces</i> table:<ul style="list-style-type: none"><li>• Added DDR4 SDRAM and RLDRAM 3 CIO.</li><li>• Removed x72 rows from table entries for DDR, DDR2, and DDR3.</li><li>• Added Arria 10 to note 11.</li><li>• Added notes 12-18.</li></ul></li><li>• Added DDR4 to descriptions of:<ul style="list-style-type: none"><li>• Clock signals</li><li>• Command and address signals</li><li>• Data, data strobe, DM/DBI, and optional ECC signals</li><li>• SDRAM DIMM options</li></ul></li><li>• Added QDR II+ Xtreme to descriptions of:<ul style="list-style-type: none"><li>• SRAM clock signals</li><li>• SRAM command signals</li><li>• SRAM address signals</li><li>• SRAM data, BWS, and QVLD signals</li></ul></li><li>• Changed title of section <i>OCT Support for Arria II GX, Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices</i> to <i>OCT Support</i>.</li><li>• Reorganized chapter to have separate sections for <i>Guidelines for Arria 10 External Memory Interface IP</i> and <i>Guidelines for UniPHY-based External Memory Interface IP</i>.</li><li>• Revised Arria 10-specific guidelines.</li></ul>
December 2013	2013.12.16	<ul style="list-style-type: none"><li>• Removed references to ALTMEMPHY and HardCopy.</li><li>• Removed references to Cyclone III and Cyclone IV devices.</li></ul>
November 2012	6.0	<ul style="list-style-type: none"><li>• Added Arria V GZ information.</li><li>• Added RLDRAM 3 information.</li><li>• Added LRDIMM information.</li></ul>
June 2012	5.0	<ul style="list-style-type: none"><li>• Added LPDDR2 information.</li><li>• Added Cyclone V information.</li><li>• Added Feedback icon.</li></ul>

Date	Version	Changes
November 2011	4.0	<ul style="list-style-type: none"><li>• Moved and reorganized <i>Planning Pin and Resource</i> section to Volume 2:Design Guidelines.</li><li>• Added <i>Additional Guidelines for Arria V GZ and Stratix V Devices</i> section.</li><li>• Added Arria V and Cyclone V information.</li></ul>
June 2011	3.0	<ul style="list-style-type: none"><li>• Moved <i>Select a Device</i> and <i>Memory IP Planning</i> chapters to Volume 1.</li><li>• Added information about interface pins.</li><li>• Added guidelines for using PLL.</li></ul>
December 2010	2.1	<ul style="list-style-type: none"><li>• Added a new section on controller efficiency.</li><li>• Added Arria II GX and Stratix V information.</li></ul>
July 2010	2.0	Updated information about UniPHY-based interfaces and Stratix V devices.
April 2010	1.0	Initial release.

# DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines

2

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The following topics provide guidelines for improving the signal integrity of your system and for successfully implementing a DDR2, DDR3, or DDR4 SDRAM interface on your system.

The following areas are discussed:

- comparison of various types of termination schemes, and their effects on the signal quality on the receiver
- proper drive strength setting on the FPGA to optimize the signal integrity at the receiver
- effects of different loading types, such as components versus DIMM configuration, on signal quality

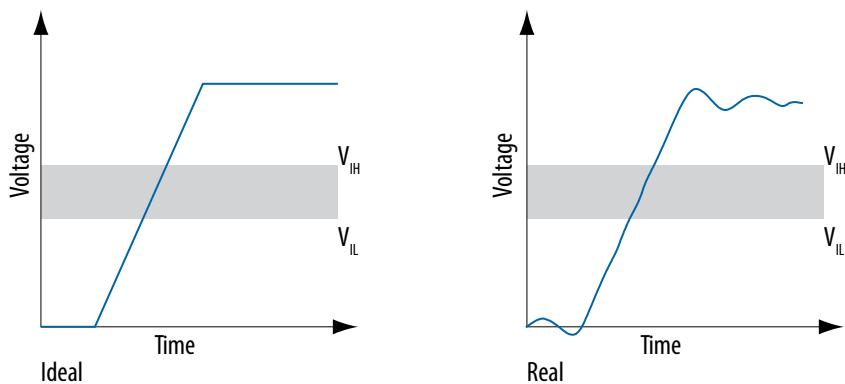
It is important to understand the trade-offs between different types of termination schemes, the effects of output drive strengths, and different loading types, so that you can swiftly navigate through the multiple combinations and choose the best possible settings for your designs.

The following key factors affect signal quality at the receiver:

- Leveling and dynamic ODT
- Proper use of termination
- Layout guidelines

As memory interface performance increases, board designers must pay closer attention to the quality of the signal seen at the receiver because poorly transmitted signals can dramatically reduce the overall data-valid margin at the receiver. The following figure shows the differences between an ideal and real signal seen by the receiver.

**Figure 2-1: Ideal and Real Signal at the Receiver**



© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

## Leveling and Dynamic Termination

DDR3 and DDR4 SDRAM DIMMs, as specified by JEDEC, always use a fly-by topology for the address, command, and clock signals.

Altera recommends that for full DDR3 or DDR4 SDRAM compatibility when using discrete DDR3 or DDR4 SDRAM components, you should mimic the JEDEC DDR3 or DDR4 fly-by topology on your custom printed circuit boards (PCB).

**Note:** Arria® II, Arria V GX, Arria V GT, Arria V SoC, Cyclone® V, and Cyclone V SoC devices do not support DDR3 SDRAM with read or write leveling, so these devices do not support standard DDR3 SDRAM DIMMs or DDR3 SDRAM components using the standard DDR3 SDRAM fly-by address, command, and clock layout topology.

**Table 2-1: Device Family Topology Support**

Device	I/O Support
Arria II	Non-leveling
Arria V GX, Arria V GT, Arria V SoC	Non-leveling
Arria V GZ	Leveling
Cyclone V GX, Cyclone V GT, Cyclone V SoC	Non-leveling
Stratix III	Leveling
Stratix IV	Leveling
Stratix V	Leveling
Arria 10	Leveling

### Related Information

[www.JEDEC.org](http://www.JEDEC.org)

## Read and Write Leveling

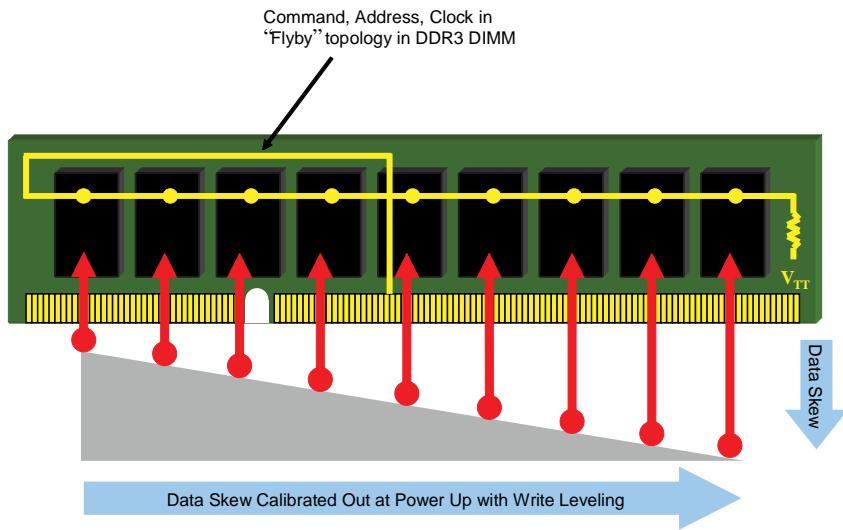
A major difference between DDR2 and DDR3/DDR4 SDRAM is the use of leveling. To improve signal integrity and support higher frequency operations, the JEDEC committee defined a fly-by termination scheme used with clocks, and command and address bus signals.

**Note:** This section describes read and write leveling in terms of a comparison between DDR3 and DDR2. Leveling in DDR4 is fundamentally similar to DDR3. Refer to the DDR4 JEDEC specifications for more information.

The following section describes leveling in DDR3, and is equally applicable to DDR4.

Fly-by topology reduces simultaneous switching noise (SSN) by deliberately causing flight-time skew between the data and strobes at every DRAM as the clock, address, and command signals traverse the DIMM, as shown in the following figure.

Figure 2-2: DDR3 DIMM Fly-By Topology Requiring Write Leveling



The flight-time skew caused by the fly-by topology led the JEDEC committee to introduce the write leveling feature on the DDR3 SDRAMs. Controllers must compensate for this skew by adjusting the timing per byte lane.

During a write, DQS groups launch at separate times to coincide with a clock arriving at components on the DIMM, and must meet the timing parameter between the memory clock and DQS defined as tDQSS of  $\pm 0.25$  tCK.

During the read operation, the memory controller must compensate for the delays introduced by the fly-by topology. The Stratix® III, Stratix IV, and Stratix V FPGAs have alignment and synchronization registers built in the I/O element to properly capture the data.

In DDR2 SDRAM, there are only two drive strength settings, full or reduced, which correspond to the output impedance of 18-ohm and 40-ohm, respectively. These output drive strength settings are static settings and are not calibrated; consequently, the output impedance varies as the voltage and temperature drifts.

The DDR3 SDRAM uses a programmable impedance output buffer. There are two drive strength settings, 34-ohm and 40-ohm. The 40-ohm drive strength setting is currently a reserved specification defined by JEDEC, but available on the DDR3 SDRAM, as offered by some memory vendors. Refer to the data sheet of the respective memory vendors for more information about the output impedance setting. You select the drive strength settings by programming the memory mode register defined by mode register 1 (MR1). To calibrate output driver impedance, an external precision resistor, RZQ, connects the ZQ pin and VSSQ. The value of this resistor must be 240-ohm  $\pm 1\%$ .

If you are using a DDR3 SDRAM DIMM, RZQ is soldered on the DIMM so you do not need to layout your board to account for it. Output impedance is set during initialization. To calibrate output driver impedance after power-up, the DDR3 SDRAM needs a calibration command that is part of the initialization and reset procedure and is updated periodically when the controller issues a calibration command.

In addition to calibrated output impedance, the DDR3 SDRAM also supports calibrated parallel ODT through the same external precision resistor, RZQ, which is possible by using a merged output driver

structure in the DDR3 SDRAM, which also helps to improve pin capacitance in the DQ and DQS pins. The ODT values supported in DDR3 SDRAM are 20-ohm, 30-ohm, 40-ohm, 60-ohm, and 120-ohm, assuming that RZQ is 240-ohm.

#### Related Information

[www.JEDEC.org](http://www.JEDEC.org)

## Dynamic ODT

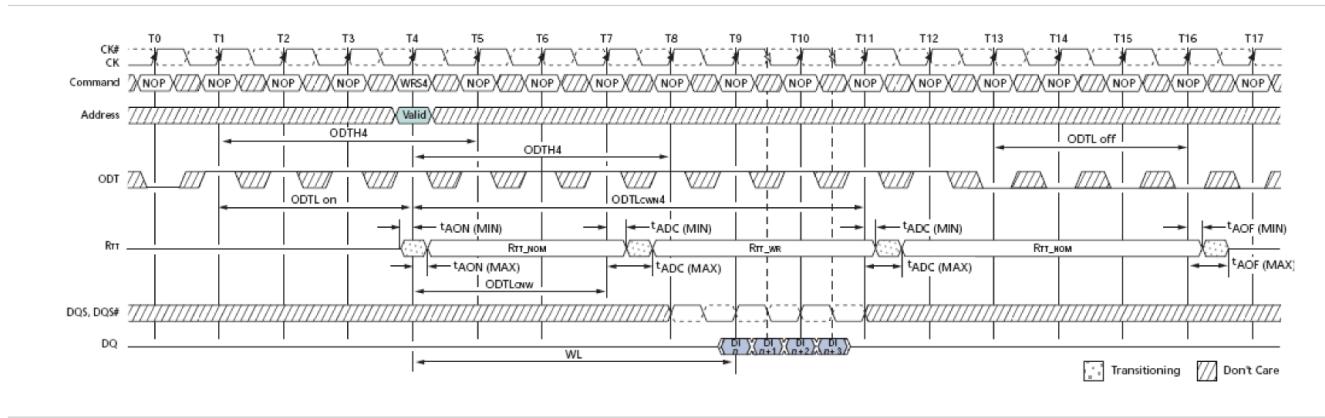
Dynamic ODT is a feature in DDR3 and DDR4 SDRAM that is not available in DDR2 SDRAM. Dynamic ODT can change the ODT setting without issuing a mode register set (MRS) command.

**Note:** This topic highlights the dynamic ODT feature in DDR3. To learn about dynamic ODT in DDR4, refer to the JEDEC DDR4 specifications.

When you enable dynamic ODT, and there is no write operation, the DDR3 SDRAM terminates to a termination setting of RTT\_NOM; when there is a write operation, the DDR3 SDRAM terminates to a setting of RTT\_WR. You can preset the values of RTT\_NOM and RTT\_WR by programming the mode registers, MR1 and MR2.

The following figure shows the behavior of ODT when you enable dynamic ODT.

**Figure 2-3: Dynamic ODT: Behavior with ODT Asserted Before and After the Write**



In the multi-load DDR3 SDRAM configuration, dynamic ODT helps reduce the jitter at the module being accessed, and minimizes reflections from any secondary modules.

For more information about using the dynamic ODT on DDR3 SDRAM, refer to the application note by Micron, *TN-41-04 DDR3 Dynamic On-Die Termination*.

#### Related Information

[www.JEDEC.org](http://www.JEDEC.org)

## Dynamic On-Chip Termination

Dynamic OCT is available in Arria V, Arria 10, Cyclone V, Stratix III, Stratix IV and Stratix V.

The dynamic OCT scheme enables series termination (RS) and parallel termination (RT) to be dynamically turned on and off during the data transfer. The series and parallel terminations are turned on or off depending on the read and write cycle of the interface. During the write cycle, the RS is turned on and the RT is turned off to match the line impedance. During the read cycle, the RS is turned off and the RT is turned on as the FPGA implements the far-end termination of the bus.

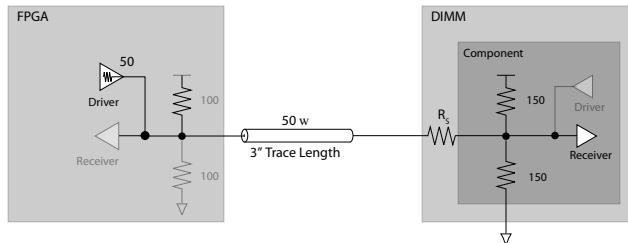
For more information about dynamic OCT, refer to the I/O features chapters in the devices handbook for your Altera device.

## FPGA Writing to Memory

The benefit of using dynamic series OCT is that when driver is driving the transmission line, it “sees” a matched transmission line with no external resistor termination.

The following figure shows dynamic series OCT scheme when the FPGA is writing to the memory.

**Figure 2-4: Dynamic Series OCT Scheme with ODT on the Memory**



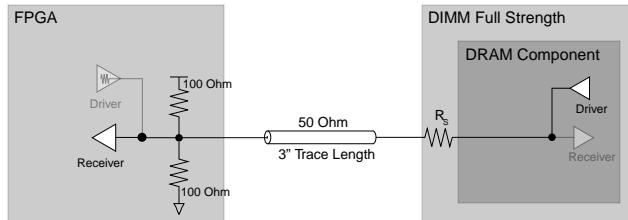
Refer to the memory vendors when determining the over- and undershoot. They typically specify a maximum limit on the input voltage to prevent reliability issues.

## FPGA Reading from Memory

The following figure shows the dynamic parallel termination scheme when the FPGA is reading from memory.

When the SDRAM DIMM is driving the transmission line, the ringing and reflection is minimal because the FPGA-side termination 50-ohm pull-up resistor is matched with the transmission line.

**Figure 2-5: Dynamic Parallel OCT Scheme with Memory-Side Series Resistor**



## Dynamic On-Chip Termination in Stratix III and Stratix IV Devices

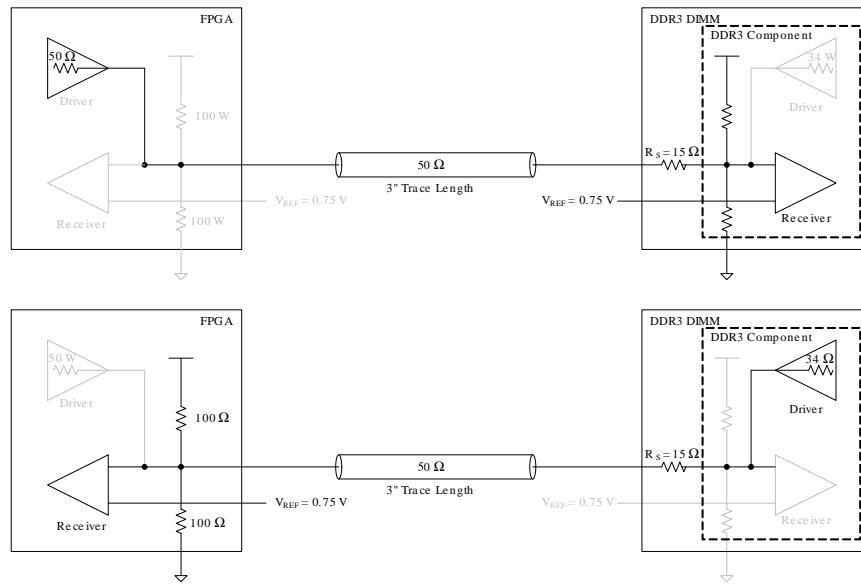
Stratix III and Stratix IV devices support on-off dynamic series and parallel termination for a bidirectional I/O in all I/O banks. Dynamic OCT is a new feature in Stratix III and Stratix IV FPGA devices.

You enable dynamic parallel termination only when the bidirectional I/O acts as a receiver and disable it when the bidirectional I/O acts as a driver. Similarly, you enable dynamic series termination only when the bidirectional I/O acts as a driver and is disable it when the bidirectional I/O acts as a receiver. The default setting for dynamic OCT is series termination, to save power when the interface is idle—no active reads or writes.

**Note:** The dynamic control operation of the OCT is separate to the output enable signal for the buffer.

UniPHY IP can enable parallel OCT only during read cycles, saving power when the interface is idle.

Figure 2-6: Dynamic OCT Between Stratix III and Stratix IV FPGA Devices



Dynamic OCT is useful for terminating any high-performance bidirectional path because signal integrity is optimized depending on the direction of the data. In addition, dynamic OCT also eliminates the need for external termination resistors when used with memory devices that support ODT (such as DDR3 SDRAM), thus reducing cost and easing board layout.

However, dynamic OCT in Stratix III and Stratix IV FPGA devices is different from dynamic ODT in DDR3 SDRAM mentioned in previous sections and these features should not be assumed to be identical.

For detailed information about the dynamic OCT feature in the Stratix III FPGA, refer to the *Stratix III Device I/O Features* chapter in volume 1 of the *Stratix III Device Handbook*.

For detailed information about the dynamic OCT feature in the Stratix IV FPGA, refer to the *I/O Features in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

#### Related Information

- [Stratix III Device I/O Features](#)
- [I/O Features in Stratix IV Devices](#)

## Dynamic OCT in Stratix V Devices

Stratix V devices also support the dynamic OCT feature and provide more flexibility. Stratix V OCT calibration uses one RZQ pin that exists in every OCT block.

You can use any one of the following as a reference resistor on the RZQ pin to implement different OCT values:

- 240-ohm reference resistor—to implement RS OCT of 34-ohm, 40-ohm, 48-ohm, 60-ohm, and 80-ohm; and RT OCT resistance of 20-ohm, 30-ohm, 40-ohm, and 120-ohm
- 100-ohm reference resistor—to implement RS OCT of 25-ohm and 50-ohm; and RT OCT resistance of 50-ohm

For detailed information about the dynamic OCT feature in the Stratix V FPGA, refer to the *I/O Features in Stratix V Devices* chapter in volume 1 of the *Stratix V Device Handbook*.

**Related Information****I/O Features in Stratix V Devices**

## Dynamic On-Chip Termination (OCT) in Arria 10 Devices

Depending upon the Rs (series) and Rt (parallel) OCT values that you want, you should choose appropriate values for the RZQ resistor and connect this resistor to the RZQ pin of the Arria 10 device.

- Select a 240-ohm reference resistor to ground to implement Rs OCT values of 34-ohm, 40-ohm, 48-ohm, 60-ohm, and 80-ohm, and Rt OCT resistance values of 20-ohm, 30-ohm, 34-ohm, 40-ohm, 60-ohm, 80-ohm, 120-ohm and 240 ohm.
- Select a 100-ohm reference resistor to ground to implement Rs OCT values of 25-ohm and 50-ohm, and an RT OCT resistance of 50-ohm.

The following table shows I/O standards and OCT values for DDR3 1.5V.

Signal Type	I/O Standard <sup>(1)</sup>	Termination Values (ohms) <sup>(1)</sup>
address/command	SSTL-15	Rs (Output Mode) - 34, 40
	SSTL-15 Class - I	Rs (Output Mode) - 50, No termination
	SSTL-15 Class - II	Rs (Output Mode) - 25, No termination
memory clock	SSTL-15	Rs (Output Mode) - 34, 40
	SSTL-15 Class - I	Rs (Output Mode) - 50, No termination
	SSTL-15 Class - II	Rs (Output Mode) - 25, No termination
data bus (DQ, DQS, DM)	SSTL-15	Rs (output Mode) - 34, 40
		Rt (Input Mode) - 20, 30, 40, 60, 120
	SSTL-15 Class - I	Rs (Output Mode) - 50, No termination
		Rt (Input Mode) - 50, No termination
	SSTL-15 Class - II	Rs (Output Mode) - 25, No termination
		Rt (Input Mode) - 50, No termination

Note to Table:

1. Shown I/O standards and termination values may not include all supported modes. For detailed information about the dynamic OCT feature in the Arria 10 FPGA, refer to the I/O Features chapter of the Arria 10 Devices Handbook.

The following table shows I/O standards and OCT values for DDR3L 1.35V.

Signal Type	I/O Standard <sup>(1)</sup>	Termination Values (ohms) <sup>(1)</sup>
address/command	SSTL-135	Rs (Output Mode) - 34, 40
memory clock	SSTL-135	Rs (Output Mode) - 34, 40

Signal Type	I/O Standard <sup>(1)</sup>	Termination Values (ohms) <sup>(1)</sup>
data bus (DQ, DQS, DM)	SSTL-135	Rs (Output Mode) - 34, 40
		Rt (Input Mode) - 20, 30, 40, 60, 120

Note to Table:

1. Shown I/O standards and termination values may not include all supported modes. For detailed information about the dynamic OCT feature in the Arria 10 FPGA, refer to the I/O Features chapter of the Arria 10 Devices Handbook.

The following table shows I/O standards and OCT values for DDR4 1.2V.

Signal Type	I/O Standard	Termination Values (ohms)
address/command	SSTL-12	Rs (Output Mode) - 40, 60
memory clock	SSTL-12	Rs (Output Mode) - 40, 60
data bus (DQ, DQS, DM, DBI) DBI	1.2-V POD	Rs (Output Mode) - 34, 40, 48, 60
		Rt (Input Mode) - 34, 40, 48, 60, 80, 120, 240

In cases where both Rs and Rt values are selected for the Data Bus, the OCT value will dynamically switch between Rs and Rt depending on the type of operation. Rs is applied during write (output) operations and Rt is applied during read (input) operations.

## Termination for DDR2 SDRAM

DDR2 adheres to the JEDEC standard of governing Stub-Series Terminated Logic (SSTL), JESD8-15a, which includes four different termination schemes.

Two commonly used termination schemes of SSTL are:

- Single parallel terminated output load with or without series resistors (Class I, as stated in JESD8-15a)
- Double parallel terminated output load with or without series resistors (Class II, as stated in JESD8-15a)

Depending on the type of signals you choose, you can use either termination scheme. Also, depending on your design's FPGA and SDRAM memory devices, you may choose external or internal termination schemes.

To reduce system cost and simplify printed circuit board layout, you may choose not to have any parallel termination on the transmission line, and use point-to-point connections between the memory interface and the memory. In this case, you may take advantage of internal termination schemes such as on-chip termination (OCT) on the FPGA side and on-die termination (ODT) on the SDRAM side when it is offered on your chosen device.

### Related Information

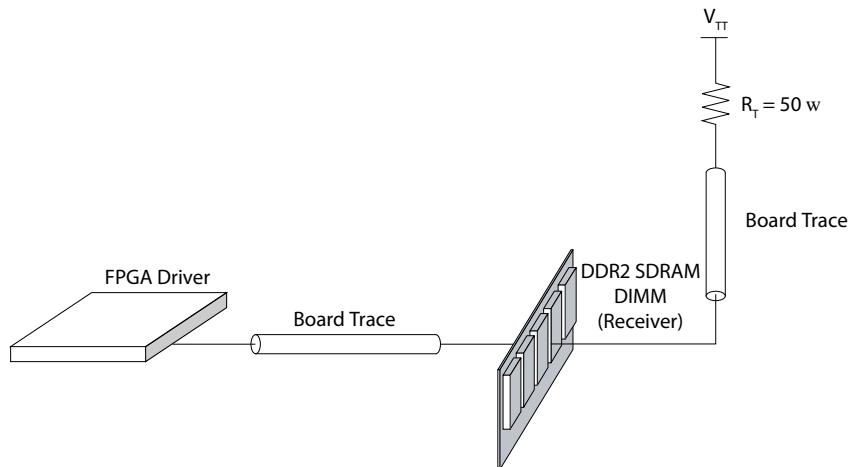
[Termination for DDR3 SDRAM](#) on page 2-15

## External Parallel Termination

If you use external termination, you must study the locations of the termination resistors to determine which topology works best for your design.

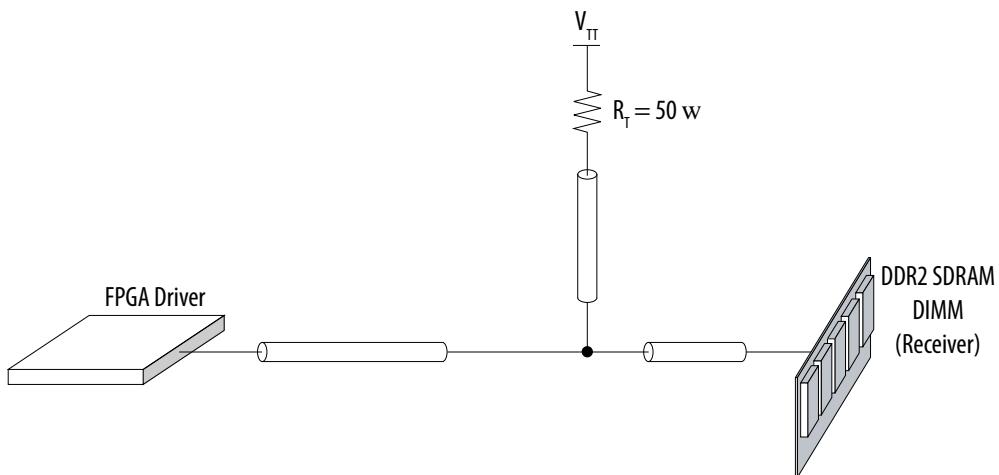
The following two figures illustrate the most common termination topologies: fly-by topology and non-fly-by topology, respectively.

**Figure 2-7: Fly-By Placement of a Parallel Resistor**



With fly-by topology, you place the parallel termination resistor after the receiver. This termination placement resolves the undesirable unterminated stub found in the non-fly-by topology. However, using this topology can be costly and complicate routing.

**Figure 2-8: Non-Fly-By Placement of a Parallel Resistor**



With non-fly-by topology, the parallel termination resistor is placed between the driver and receiver (closest to the receiver). This termination placement is easier for board layout, but results in a short stub, which causes an unterminated transmission line between the terminating resistor and the receiver. The unterminated transmission line results in ringing and reflection at the receiver.

If you do not use external termination, DDR2 offers ODT and Altera FPGAs have varying levels of OCT support. You should explore using ODT and OCT to decrease the board power consumption and reduce the required board space.

## On-Chip Termination

OCT technology is offered on Arria II GX, Arria II GZ, Arria V, Arria 10, Cyclone V, MAX 10, Stratix III, Stratix IV, and Stratix V devices.

The following table summarizes the extent of OCT support for devices earlier than Arria 10. This table provides information about SSTL-18 standards because SSTL-18 is the supported standard for DDR2 memory interface by Altera FPGAs.

For Arria II, Stratix III and Stratix IV devices, on-chip series (RS) termination is supported only on output and bidirectional buffers. The value of RS with calibration is calibrated against a 25-ohm resistor for class II and 50-ohm resistor for class I connected to RUP and RDN pins and adjusted to  $\pm 1\%$  of 25-ohm or 50-ohm . On-chip parallel (RT) termination is supported only on inputs and bidirectional buffers. The value of RT is calibrated against 100-ohm connected to the RUP and RDN pins. Calibration occurs at the end of device configuration. Dynamic OCT is supported only on bidirectional I/O buffers.

For Arria V, Cyclone V, and Stratix V devices, RS and RT values are calibrated against the on-board resistor RZQ. If you want 25 or 50 ohm values for your RS and RT, you must connect a 100 ohm resistor with a tolerance of  $+/-1\%$  to the RZQ pin .

For more information about on-chip termination, refer to the device handbook for the device that you are using.

**Table 2-2: On-Chip Termination Schemes**

Termination Scheme	SSTL-18	FPGA Device						
		Arria II GX	Arria II GZ	Arria V	Cyclone V	MAX 10	Stratix III and Stratix IV	Stratix V (1)
		Column and Row I/O	Column I/O					
On-Chip Series Termination without Calibration	Class I	50	50	50	50	50	50	50
	Class II	25	25	25	25	25	25	25
On-Chip Series Termination with Calibration	Class I	50	50	50	50	50	50	50
	Class II	25	25	25	25	25	25	25
On-Chip Parallel Termination with Calibration	Class I and Class II	—	50	50	50	—	50	50

Termination Scheme	SSTL-18	FPGA Device						
		Arria II GX	Arria II GZ	Arria V	Cyclone V	MAX 10	Stratix III and Stratix IV	Stratix V (1)
		Column and Row I/O	Column I/O					

Note to Table:

- Row I/O is not available for external memory interfaces in Stratix V devices.

## Recommended Termination Schemes

The following table provides the recommended termination schemes for major DDR2 memory interface signals.

Signals include data (DQ), data strobe (DQS/DQSn), data mask (DM), clocks (`mem_clk`/`mem_clk_n`), and address and command signals.

When interfacing with multiple DDR2 SDRAM components where the address, command, and memory clock pins are connected to more than one load, follow these steps:

- Simulate the system to get the new slew-rate for these signals.
- Use the derated tIS and tIH specifications from the DDR2 SDRAM data sheet based on the simulation results.
- If timing deration causes your interface to fail timing requirements, consider signal duplication of these signals to lower their loading, and hence improve timing.

**Note:** Altera uses Class I and Class II termination in this table to refer to drive strength, and not physical termination.

**Note:** You must simulate your design for your system to ensure correct operation.

**Table 2-3: Termination Recommendations (1)**

Device Family	Signal Type	SSTL 18 IO Standard (2) (3) (4) (5) (6)	FPGA-End Discrete Termination	Memory-End Termination 1 (Rank/DIMM)	Memory I/O Standard
---------------	-------------	--	-------------------------------	--------------------------------------	---------------------

### Arria II GX

DDR2 component	DQ	Class I R50 CAL	50-ohm Parallel to VTT discrete	ODT75 (7)	HALF (8)
	DQS DIFF (13)	DIFF Class R50 CAL	50-ohm Parallel to VTT discrete	ODT75 (7)	HALF (8)

Device Family	Signal Type	SSTL 18 IO Standard <sup>(2)(3)</sup> <sup>(4)(5)(6)</sup>	FPGA-End Discrete Termination	Memory-End Termination 1 (Rank/DIMM)	Memory I/O Standard
DDR2 DIMM	DQS SE <sup>(12)</sup>	Class I R50 CAL	50-ohm Parallel to VTT discrete	ODT75 <sup>(7)</sup>	HALF <sup>(8)</sup>
	DM	Class I R50 CAL	N/A	ODT75 <sup>(7)</sup>	N/A
	Address and command	Class I MAX	N/A	56-ohm parallel to VTT discrete	N/A
	Clock	DIFF Class I R50 CAL	N/A	$\times 1 = 100\text{-ohm differential}$ $\times 2 = 200\text{-ohm differential}$	N/A
	DQ	Class I R50 CAL	50-ohm Parallel to VTT discrete	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
	DQS DIFF <sup>(13)</sup>	DIFF Class I R50 CAL	50-ohm Parallel to VTT discrete	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
Arria V and Cyclone V	DQS SE <sup>(12)</sup>	Class I R50 CAL	50-ohm Parallel to VTT discrete	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
	DM	Class I R50 CAL	N/A	ODT75 <sup>(7)</sup>	N/A
	Address and command	Class I MAX	N/A	56-ohm parallel to VTT discrete	N/A
	Clock	DIFF Class I R50 CAL	N/A	N/A = on DIMM	N/A
	DQ	Class I R50/ P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	HALF <sup>(8)</sup>

Device Family	Signal Type	SSTL 18 IO Standard <sup>(2)(3)</sup> <sup>(4)(5)(6)</sup>	FPGA-End Discrete Termination	Memory-End Termination 1 (Rank/DIMM)	Memory I/O Standard
DDR2 DIMM	DQS DIFF <sup>(13)</sup>	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	HALF <sup>(8)</sup>
	DQS SE <sup>(12)</sup>	Class I R50/ P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	HALF <sup>(8)</sup>
	DM	Class I R50 CAL	N/A	ODT75 <sup>(7)</sup>	N/A
	Address and command	Class I MAX	N/A	56-ohm parallel to VTT discrete	N/A
	Clock	DIFF Class I R50 NO CAL	N/A	$\times 1 = 100\text{-ohm differential}$ <sup>(10)</sup> $\times 2 = 200\text{-ohm differential}$ <sup>(11)</sup>	N/A
	DQ	Class I R50/ P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
Arria II GZ, Stratix III, Stratix IV, and Stratix V	DQS DIFF <sup>(13)</sup>	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
	DQS SE <sup>(12)</sup>	Class I R50/ P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
	DM	Class I R50 CAL	N/A	ODT75 <sup>(7)</sup>	N/A
	Address and command	Class I MAX	N/A	56-ohm parallel to VTT discrete	N/A
	Clock	DIFF Class I R50 NO CAL	N/A	N/A = on DIMM	N/A

**Arria II GZ, Stratix III, Stratix IV, and Stratix V**

Device Family	Signal Type	SSTL 18 IO Standard <sup>(2)(3)</sup> <sup>(4)(5)(6)</sup>	FPGA-End Discrete Termination	Memory-End Termination 1 (Rank/DIMM)	Memory I/O Standard
DDR2 component	DQ	Class I R50/ P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	HALF <sup>(8)</sup>
	DQS DIFF <sup>(13)</sup>	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	HALF <sup>(8)</sup>
	DQS SE <sup>(12)</sup>	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	HALF <sup>(8)</sup>
	DM	Class I R50 CAL	N/A	ODT75 <sup>(7)</sup>	N/A
	Address and command	Class I MAX	N/A	56-ohm Parallel to VTT discrete	N/A
	Clock	DIFF Class I R50 NO CAL	N/A	x1 = 100-ohm differential <sup>(10)</sup>  x2 = 200-ohm differential <sup>(11)</sup>	N/A
DDR2 DIMM	DQ	Class I R50/ P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
	DQS DIFF <sup>(13)</sup>	DIFF Class I R50/P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
	DQS SE <sup>(12)</sup>	Class I R50/ P50 DYN CAL	N/A	ODT75 <sup>(7)</sup>	FULL <sup>(9)</sup>
	DM	Class I R50 CAL	N/A	ODT75 <sup>(7)</sup>	N/A
	Address and command	Class I MAX	N/A	56-ohm Parallel to VTT discrete	N/A

Device Family	Signal Type	SSTL 18 IO Standard <sup>(2)(3)</sup> <sup>(4)(5)(6)</sup>	FPGA-End Discrete Termination	Memory-End Termination 1 (Rank/DIMM)	Memory I/O Standard
	Clock	DIFF Class I R50 NO CAL	N/A	N/A = on DIMM	N/A
<b>MAX 10</b>					
DDR2 component	DQ/DQS	Class I 12 mA	50-ohm Parallel to VTT discrete	ODT75 <sup>(7)</sup>	HALF <sup>(8)</sup>
	DM	Class I 12 mA	N/A	80-ohm Parallel to VTT discrete	N/A
	Address and command	Class I MAX	N/A		N/A
	Clock	Class I 12 mA	N/A	x1 = 100-ohm differential <sup>(10)</sup> x2 = 200-ohm differential <sup>(11)</sup>	N/A

Notes to Table:

1. N/A is not available.
2. R is series resistor.
3. P is parallel resistor.
4. DYN is dynamic OCT.
5. NO CAL is OCT without calibration.
6. CAL is OCT with calibration.
7. ODT75 vs. ODT50 on the memory has the effect of opening the eye more, with a limited increase in overshoot/undershoot.
8. HALF is reduced drive strength.
9. FULL is full drive strength.
- 10.x1 is a single-device load.
- 11.x2 is two-device load. For example, you can feed two out of nine devices on a single rank DIMM with a single clock pair—except for MAX 10, which doesn't support DIMMs.
- 12.DQS SE is single-ended DQS.
- 13.DQS DIFF is differential DQS

## Termination for DDR3 SDRAM

DDR3 DIMMs have terminations on all unidirectional signals, such as memory clocks, and addresses and commands; thus eliminating the need for them on the FPGA PCB. In addition, using the ODT feature on the DDR3 SDRAM and the dynamic OCT feature of Stratix III, Stratix IV, Stratix V, and Arria 10 FPGA

devices completely eliminates any external termination resistors; thus simplifying the layout for the DDR3 SDRAM interface when compared to that of the DDR2 SDRAM interface.

The following topics describe the correct way to terminate a DDR3 SDRAM interface together with Stratix III, Stratix IV, and Stratix V FPGA devices.

**Note:** If you are using a DDR3 SDRAM without leveling interface, refer to “Board Termination for DDR2 SDRAM”. Note also that Arria V and Cyclone V devices do not support DDR3 with leveling.

#### Related Information

[Termination for DDR2 SDRAM](#) on page 2-8

## Terminations for Single-Rank DDR3 SDRAM Unbuffered DIMM

The most common implementation of the DDR3 SDRAM interface is the unbuffered DIMM (UDIMM). You can find DDR3 SDRAM UDIMMs in many applications, especially in PC applications.

The following table lists the recommended termination and drive strength setting for UDIMM and Stratix III, Stratix IV, and Stratix V FPGA devices.

**Note:** These settings are just recommendations for you to get started. Simulate with real board and try different settings to get the best SI.

**Table 2-4: Drive Strength and ODT Setting Recommendations for Single-Rank UDIMM**

Signal Type	SSTL 15 I/O Standard <sup>(1)</sup>	FPGA End On-Board Termination <sup>(2)</sup>	Memory End Termination for Write	Memory Driver Strength for Read
DQ	Class I R50C/ G50C <sup>(3)</sup>	—	60-ohm ODT <sup>(4)</sup>	40-ohm <sup>(4)</sup>
DQS	Differential Class I R50C/G50C <sup>(3)</sup>	—	60-ohm ODT <sup>(4)</sup>	40-ohm <sup>(4)</sup>
DM	Class I R50C <sup>(3)</sup>	—	60-ohm ODT <sup>(4)</sup>	40-ohm <sup>(4)</sup>
Address and Command	Class I with maximum drive strength	—	39-ohm on-board termination to V <sub>DD</sub> <sup>(5)</sup>	
CK/CK#	Differential Class I R50C	—	On-board <sup>(5)</sup> 2.2 pf compensation cap before the first component; 36-ohm termination to V <sub>DD</sub> for each arm (72-ohm differential); add 0.1 uF just before V <sub>DD</sub> .	

Signal Type	SSTL 15 I/O Standard <sup>(1)</sup>	FPGA End On-Board Termination <sup>(2)</sup>	Memory End Termination for Write	Memory Driver Strength for Read
Notes to Table:				
1.	UniPHY IP automatically implements these settings.			
2.	Altera recommends that you use dynamic on-chip termination (OCT) for Stratix III and Stratix IV device families.			
3.	R50C is series with calibration for write, G50C is parallel 50 with calibration for read.			
4.	You can specify these settings in the parameter editor.			
5.	For DIMM, these settings are already implemented on the DIMM card; for component topology, Altera recommends that you mimic termination scheme on the DIMM card on your board.			

You can implement a DDR3 SDRAM UDIMM interface in several permutations, such as single DIMM or multiple DIMMs, using either single-ranked or dual-ranked UDIMMs. In addition to the UDIMM's form factor, these termination recommendations are also valid for small-outline (SO) DIMMs and MicroDIMMs.

## Terminations for Multi-Rank DDR3 SDRAM Unbuffered DIMM

You can implement a DDR3 SDRAM UDIMM interface in several permutations, such as single DIMM or multiple DIMMs, using either single-ranked or dual-ranked UDIMMs. In addition to the UDIMM's form factor, these termination recommendations are also valid for small-outline (SO) DIMMs and MicroDIMMs.

The following table lists the different permutations of a two-slot DDR3 SDRAM interface and the recommended ODT settings on both the memory and controller when writing to memory.

**Table 2-5: DDR3 SDRAM ODT Matrix for Writes <sup>(1)(2)</sup>**

Slot 1	Slot 2	Write To	Controller OCT <sup>(3)</sup>	Slot 1		Slot 2	
				Rank 1	Rank 2	Rank 1	Rank 2
DR	DR	Slot 1	Series 50-ohm	120-ohm <sup>(4)</sup>	ODT off	ODT off	40-ohm <sup>(4)</sup>
		Slot 2	Series 50-ohm	ODT off	40-ohm <sup>(4)</sup>	120-ohm <sup>(4)</sup>	ODT off
SR	SR	Slot 1	Series 50-ohm	120-ohm <sup>(4)</sup>	Unpopulated	40-ohm <sup>(4)</sup>	Unpopulated
		Slot 2	Series 50-ohm	40-ohm <sup>(4)</sup>	Unpopulated	120-ohm <sup>(4)</sup>	Unpopulated
DR	Empty	Slot 1	Series 50-ohm	120-ohm <sup>(4)</sup>	ODT off	Unpopulated	Unpopulated

Slot 1	Slot 2	Write To	Controller OCT <sup>(3)</sup>	Slot 1		Slot 2	
				Rank 1	Rank 2	Rank 1	Rank 2
Empty	DR	Slot 2	Series 50-ohm	Unpopulated	Unpopulated	120-ohm <sup>(4)</sup>	ODT off
SR	Empty	Slot 1	Series 50-ohm	120-ohm <sup>(4)</sup>	Unpopulated	Unpopulated	Unpopulated
Empty	SR	Slot 2	Series 50-ohm	Unpopulated	Unpopulated	120-ohm <sup>(4)</sup>	Unpopulated

Notes to Table:

1. SR: single-ranked DIMM; DR: dual-ranked DIMM.
2. These recommendations are taken from the DDR3 ODT and Dynamic ODT session of the JEDEC DDR3 2007 Conference, Oct 3-4, San Jose, CA.
3. The controller in this case is the FPGA.
4. Dynamic ODT is required. For example, the ODT of Slot 2 is set to the lower ODT value of 40-ohms when the memory controller is writing to Slot 1, resulting in termination and thus minimizing any reflection from Slot 2. Without dynamic ODT, Slot 2 will not be terminated.

The following table lists the different permutations of a two-slot DDR3 SDRAM interface and the recommended ODT settings on both the memory and controller when reading from memory.

Table 2-6: DDR3 SDRAM ODT Matrix for Reads <sup>(1)(2)</sup>

Slot 1	Slot 2	Read From	Controller OCT <sup>(3)</sup>	Slot 1		Slot 2	
				Rank 1	Rank 2	Rank 1	Rank 2
DR	DR	Slot 1	Parallel 50-ohm	ODT off	ODT off	ODT off	40-ohm <sup>(4)</sup>
		Slot 2	Parallel 50-ohm	ODT off	40-ohm <sup>(4)</sup>	ODT off	ODT off
SR	SR	Slot 1	Parallel 50-ohm	ODT off	Unpopulated	40-ohm <sup>(4)</sup>	Unpopulated
		Slot 2	Parallel 50-ohm	40-ohm <sup>(4)</sup>	Unpopulated	ODT off	Unpopulated
DR	Empty	Slot 1	Parallel 50-ohm	ODT off	ODT off	Unpopulated	Unpopulated
Empty	DR	Slot 2	Parallel 50-ohm	Unpopulated	Unpopulated	ODT off	ODT off

Slot 1	Slot 2	Read From	Controller OCT <sup>(3)</sup>	Slot 1		Slot 2	
				Rank 1	Rank 2	Rank 1	Rank 2
SR	Empty	Slot 1	Parallel 50-ohm	ODT off	Unpopulated	Unpopulated	Unpopulated
Empty	SR	Slot 2	Parallel 50-ohm	Unpopulated	Unpopulated	ODT off	Unpopulated

Notes to Table:

1. SR: single-ranked DIMM; DR: dual-ranked DIMM.
2. These recommendations are taken from the DDR3 ODT and Dynamic ODT session of the JEDEC DDR3 2007 Conference, Oct 3-4, San Jose, CA.
3. The controller in this case is the FPGA. JEDEC typically recommends 60-ohms, but this value assumes that the typical motherboard trace impedance is 60-ohms and that the controller supports this termination. Altera recommends using a 50-ohm parallel OCT when reading from the memory.

## Terminations for DDR3 SDRAM Registered DIMM

The difference between a registered DIMM (RDIMM) and a UDIMM is that the clock, address, and command pins of the RDIMM are registered or buffered on the DIMM before they are distributed to the memory devices. For a controller, each clock, address, or command signal has only one load, which is the register or buffer. In a UDIMM, each controller pin must drive a fly-by wire with multiple loads.

You do not need to terminate the clock, address, and command signals on your board because these signals are terminated at the register. However, because of the register, these signals become point-to-point signals and have improved signal integrity making the drive strength requirements of the FPGA driver pins more relaxed. Similar to the signals in a UDIMM, the DQS, DQ, and DM signals on a RDIMM are not registered. To terminate these signals, refer to “DQS, DQ, and DM for DDR3 SDRAM UDIMM”.

### Related Information

[DQS, DQ, and DM for DDR3 SDRAM UDIMM](#)

## Terminations for DDR3 SDRAM Load-Reduced DIMM

RDIMM and LRDIMM differ in that DQ, DQS, and DM signals are registered or buffered in the LRDIMM. The LRDIMM buffer IC is a superset of the RDIMM buffer IC. The buffer IC isolates the memory interface signals from loading effects of the memory chip. Reduced electrical loading allows a system to operate at higher frequency and higher density.

**Note:** If you want to use your DIMM socket for UDIMM and RDIMM/LRDIMM, you must create the necessary redundant connections on the board from the FPGA to the DIMM socket. For example, the number of chip select signals required for a single-rank UDIMM is one, but for single-rank RDIMM the number of chip selects required is two. RDIMM and LRDIMM have parity signals associated with the address and command bus which UDIMM does not have. Consult the DIMM manufacturer’s data sheet for detailed information about the necessary pin connections for various DIMM topologies.

## Terminations for DDR3 SDRAM Components With Leveling

The following topics discusses terminations used to achieve optimum performance for designing the DDR3 SDRAM interface using discrete DDR3 SDRAM components.

In addition to using DDR3 SDRAM DIMM to implement your DDR3 SDRAM interface, you can also use DDR3 SDRAM components. However, for applications that have limited board real estate, using DDR3 SDRAM components reduces the need for a DIMM connector and places components closer, resulting in denser layouts.

### DDR3 SDRAM Components With or Without Leveling

The DDR3 SDRAM UDIMM is laid out to the JEDEC specification. The JEDEC specification is available from either the JEDEC Organization website ([www.JEDEC.org](http://www.JEDEC.org)) or from the memory vendors. However, when you are designing the DDR3 SDRAM interface using discrete SDRAM components, you may desire a layout scheme that is different than the DIMM specification.

You have the following options:

- Mimic the standard DDR3 SDRAM DIMM, using a fly-by topology for the memory clocks, address, and command signals. This option needs read and write leveling, so you must use the UniPHY IP with leveling.
- Mimic a standard DDR2 SDRAM DIMM, using a balanced (symmetrical) tree-type topology for the memory clocks, address, and command signals. Using this topology results in unwanted stubs on the command, address, and clock, which degrades signal integrity and limits the performance of the DDR3 SDRAM interface.

#### Related Information

- [Layout Guidelines for DDR3 and DDR4 SDRAM Interfaces](#) on page 2-30
- [www.JEDEC.org](http://www.JEDEC.org)

## DDR3 and DDR4 on Arria 10 Devices

The following topics describe considerations specific to DDR3 and DDR4 external memory interface protocols on Arria 10 devices.

#### Related Information

[www.JEDEC.org](http://www.JEDEC.org)

## Dynamic On-Chip Termination (OCT) in Arria 10 Devices

Depending upon the Rs (series) and Rt (parallel) OCT values that you want, you should choose appropriate values for the RZQ resistor and connect this resistor to the RZQ pin of the Arria 10 device.

- Select a 240-ohm reference resistor to ground to implement Rs OCT values of 34-ohm, 40-ohm, 48-ohm, 60-ohm, and 80-ohm, and Rt OCT resistance values of 20-ohm, 30-ohm, 34-ohm, 40-ohm, 60-ohm, 80-ohm, 120-ohm and 240 ohm.
- Select a 100-ohm reference resistor to ground to implement Rs OCT values of 25-ohm and 50-ohm, and an RT OCT resistance of 50-ohm.

The following table shows I/O standards and OCT values for DDR3 1.5V.

Signal Type	I/O Standard <sup>(1)</sup>	Termination Values (ohms) <sup>(1)</sup>
address/command	SSTL-15	Rs (Output Mode) - 34, 40
	SSTL-15 Class - I	Rs (Output Mode) - 50, No termination
	SSTL-15 Class - II	Rs (Output Mode) - 25, No termination
memory clock	SSTL-15	Rs (Output Mode) - 34, 40
	SSTL-15 Class - I	Rs (Output Mode) - 50, No termination
	SSTL-15 Class - II	Rs (Output Mode) - 25, No termination
data bus (DQ, DQS, DM)	SSTL-15	Rs (output Mode) - 34, 40
		Rt (Input Mode) - 20, 30, 40, 60, 120
	SSTL-15 Class - I	Rs (Output Mode) - 50, No termination
		Rt (Input Mode) - 50, No termination
	SSTL-15 Class - II	Rs (Output Mode) - 25, No termination
		Rt (Input Mode) - 50, No termination

Note to Table:

1. Shown I/O standards and termination values may not include all supported modes. For detailed information about the dynamic OCT feature in the Arria 10 FPGA, refer to the I/O Features chapter of the Arria 10 Devices Handbook.

The following table shows I/O standards and OCT values for DDR3L 1.35V.

Signal Type	I/O Standard <sup>(1)</sup>	Termination Values (ohms) <sup>(1)</sup>
address/command	SSTL-135	Rs (Output Mode) - 34, 40
memory clock	SSTL-135	Rs (Output Mode) - 34, 40
data bus (DQ, DQS, DM)	SSTL-135	Rs (Output Mode) - 34, 40
		Rt (Input Mode) - 20, 30, 40, 60, 120

Note to Table:

1. Shown I/O standards and termination values may not include all supported modes. For detailed information about the dynamic OCT feature in the Arria 10 FPGA, refer to the I/O Features chapter of the Arria 10 Devices Handbook.

The following table shows I/O standards and OCT values for DDR4 1.2V.

Signal Type	I/O Standard	Termination Values (ohms)
address/command	SSTL-12	Rs (Output Mode) - 40, 60

Signal Type	I/O Standard	Termination Values (ohms)
memory clock	SSTL-12	Rs (Output Mode) - 40, 60
data bus (DQ, DQS, DM, DBI) DBI	1.2-V POD	Rs (Output Mode) - 34, 40, 48, 60
		Rt (Input Mode) - 34, 40, 48, 60, 80, 120, 240

In cases where both Rs and Rt values are selected for the Data Bus, the OCT value will dynamically switch between Rs and Rt depending on the type of operation. Rs is applied during write (output) operations and Rt is applied during read (input) operations.

## Dynamic On-Die Termination (ODT) in DDR4

In DDR4, in addition to the Rtt\_nom and Rtt\_wr values, which are applied during read and write respectively, a third option called Rtt\_park is available. When Rtt\_park is enabled, a selected termination value is set in the DRAM when ODT is driven low.

Rtt\_nom and Rtt\_wr work the same as in DDR3, which is described in *Dynamic ODT for DDR3*.

Refer to the DDR4 JEDEC specification or your memory vendor data sheet for details about available termination values and functional description for dynamic ODT in DDR4 devices.

For DDR4 LRDIMM, if SPD byte 152 calls for different values of Rtt\_Park to be used for package ranks 0 and 1 versus package ranks 2 and 3, set the value to the larger of the two impedance settings.

## Choosing Terminations on Arria 10 Devices

To determine optimal on-chip termination (OCT) and on-die termination (ODT) values for best signal integrity, you should simulate your memory interface in HyperLynx or a similar tool.

If the optimal OCT and ODT termination values as determined by simulation are not available in the list of available values in the parameter editor, select the closest available termination values for OCT and ODT.

Refer to *Dynamic On-Chip Termination (OCT) in Arria 10 Devices* for examples of various OCT modes. Refer to the *Arria 10 Device Handbook* for more information about OCT. For information on available ODT choices, refer to your memory vendor data sheet.

### Related Information

[Dynamic On-Chip Termination \(OCT\) in Arria 10 Devices](#) on page 2-7

## On-Chip Termination Recommendations for DDR3 and DDR4 on Arria 10 Devices

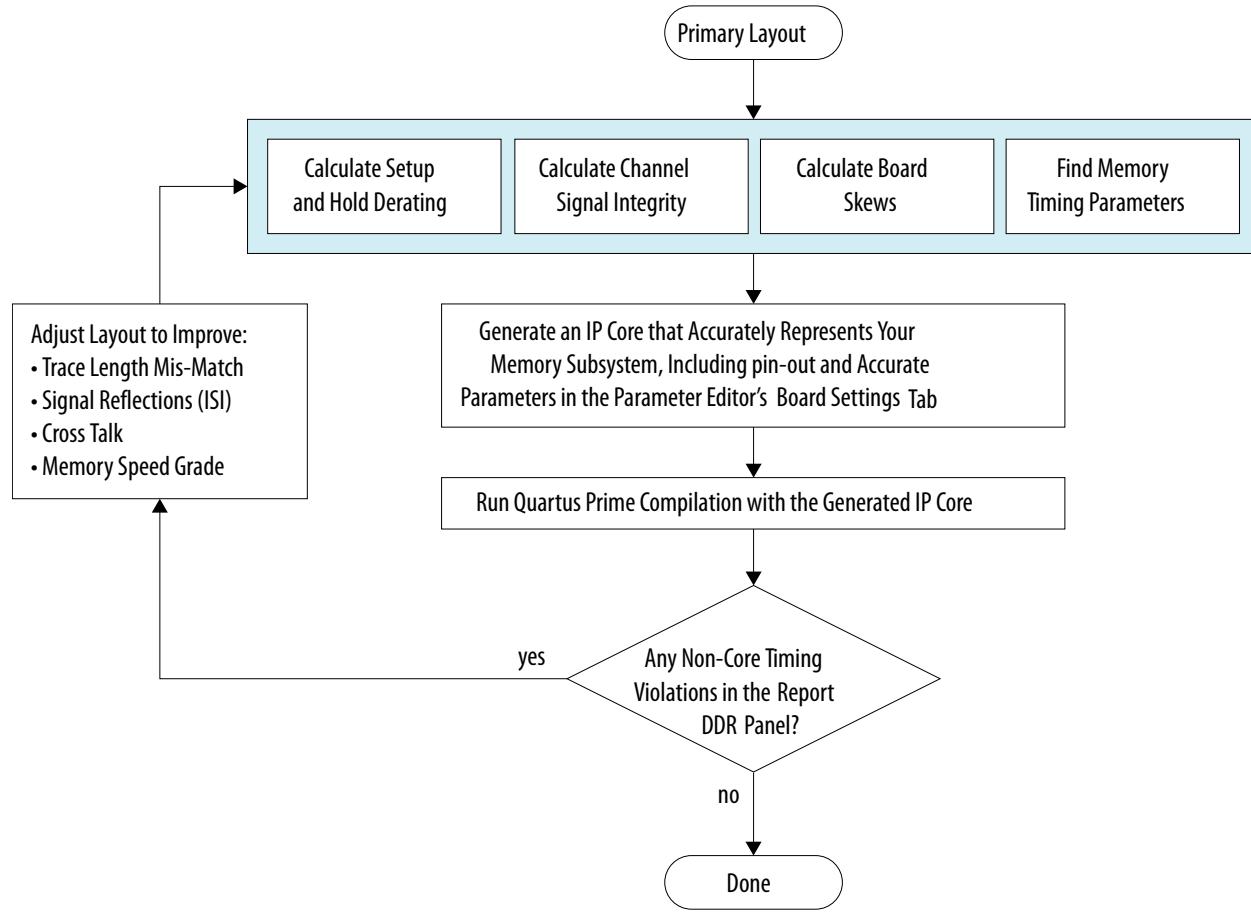
- Output mode (drive strength) for Address/Command/Clock and Data Signals: Depending upon the I/O standard that you have selected, you would have a range of selections expressed in terms of ohms or milamps. A value of 34 to 40 ohms or 12 mA is a good starting point for output mode drive strength.
- Input mode (parallel termination) for Data and Data Strobe signals: A value of 40 or 60 ohms is a good starting point for FPGA side input termination.

## Layout Approach

For all practical purposes, you can regard the TimeQuest timing analyzer's report on your memory interface as definitive for a given set of memory and board timing parameters.

You will find timing under **Report DDR** in TimeQuest and on the **Timing Analysis** tab in the parameter editor.

The following flowchart illustrates the recommended process to follow during the board design phase, to determine timing margin and make iterative improvements to your design.



### Setup and Hold Derating

For information on calculating derating parameters, refer to *Implementing and Parameterizing Memory IP*, in the *External Memory Interface Handbook*.

### Channel Signal Integrity

For information on determining channel signal integrity for Stratix V and earlier products, refer to the wiki page: [http://www.alterawiki.com/wiki/Measuring\\_Channel\\_Signal\\_Integrity](http://www.alterawiki.com/wiki/Measuring_Channel_Signal_Integrity).

For information on Arria 10 board timing and layout approach, refer to: [http://www.alterawiki.com/wiki/Arria\\_10\\_EMIF\\_Simulation\\_Guidance](http://www.alterawiki.com/wiki/Arria_10_EMIF_Simulation_Guidance).

## Board Skew

For information on calculating board skew parameters, refer to *Implementing and Parameterizing Memory IP*, in the *External Memory Interface Handbook*.

The Board Skew Parameter Tool is an interactive tool that can help you calculate board skew parameters if you know the absolute delay values for all the memory related traces.

## Memory Timing Parameters

For information on the memory timing parameters to be entered into the parameter editor, refer to the datasheet for your external memory device.

### Related Information

#### [Board Skew Parameter Tool](#)

## Design Layout Guidelines

The general layout guidelines in the following topic apply to DDR2, DDR3, and DDR4 SDRAM interfaces.

These guidelines will help you plan your board layout, but are not meant as strict rules that must be adhered to. Altera recommends that you perform your own board-level simulations to ensure that the layout you choose for your board allows you to achieve your desired performance.

For more information about how the memory manufacturers route these address and control signals on their DIMMs, refer to the Cadence PCB browser from the Cadence website, at [www.cadence.com](http://www.cadence.com). The various JEDEC example DIMM layouts are available from the JEDEC website, at [www.jedec.org](http://www.jedec.org).

For more information about board skew parameters, refer to Board Skews in the Implementing and Parameterizing Memory IP chapter. For assistance in calculating board skew parameters, refer to the board skew calculator tool, which is available at the Altera website.

- Note:**
1. The following layout guidelines include several +/- length based rules. These length based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristic of the interface. They do not include any margin for crosstalk.
  2. To ensure reliable timing closure to and from the periphery of the device, signals to and from the periphery should be registered before any further logic is connected.

Altera recommends that you get accurate time base skew numbers for your design when you simulate the specific implementation.

### Related Information

- [www.JEDEC.org](http://www.JEDEC.org)
- [www.cadence.com](http://www.cadence.com)
- [www.mentor.com](http://www.mentor.com)
- [Board Skew Parameters Tool](#)
- <http://www.jedec.org/download/DesignFiles/DDR2/default1.cfm>

## General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the

slew rate and propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

**Table 2-7: General Layout Guidelines**

Parameter	Guidelines
Impedance	<ul style="list-style-type: none"><li>• All unused via pads must be removed, because they cause unwanted capacitance.</li><li>• Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.</li></ul>
Decoupling Parameter	<ul style="list-style-type: none"><li>• Use 0.1 uF in 0402 size to minimize inductance</li><li>• Make VTT voltage decoupling close to pull-up resistors</li><li>• Connect decoupling caps between VTT and ground</li><li>• Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin</li><li>• Verify the capacitive decoupling using the Altera Power Distribution Network Design Tool</li></ul>
Power	<ul style="list-style-type: none"><li>• Route GND and V<sub>CC</sub> as planes</li><li>• Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation</li><li>• Route VTT as islands or 250-mil (6.35-mm) power traces</li><li>• Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces</li></ul>
General Routing	<p>All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Altera recommend that signals from the same net group always be routed on the same layer.</p> <ul style="list-style-type: none"><li>• Use 45° angles (<i>not</i> 90° corners)</li><li>• Avoid T-Junctions for critical nets or clocks</li><li>• Avoid T-junctions greater than 250 mils (6.35 mm)</li><li>• Disallow signals across split planes</li><li>• Restrict routing other signals close to system reset signals</li><li>• Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks</li></ul>

#### Related Information

[Power Distribution Network Design Tool](#)

## Layout Guidelines for DDR2 SDRAM Interface

Unless otherwise specified, the following guidelines apply to the following topologies:

- DIMM—UDIMM topology
- DIMM—RDIMM topology
- Discrete components laid out in UDIMM topology
- Discrete components laid out in RDIMM topology

Trace lengths for CLK and DQS should tightly match for each memory component. To match the trace lengths on the board, a balanced tree topology is recommended for clock and address and command signal routing. In addition to matching the trace lengths, you should ensure that DDR timing is passing in the Report DDR Timing report. For Stratix devices, this timing is shown as Write Leveling  $t_{DQSS}$  timing. For Arria and Cyclone devices, this timing is shown as CK vs DQS timing.

For a table of device family topology support, refer to *Leveling and Dynamic ODT*.

The following table lists DDR2 SDRAM layout guidelines. These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the slew rate and propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

**Note:** The following layout guidelines also apply to DDR3 SDRAM without leveling interfaces.

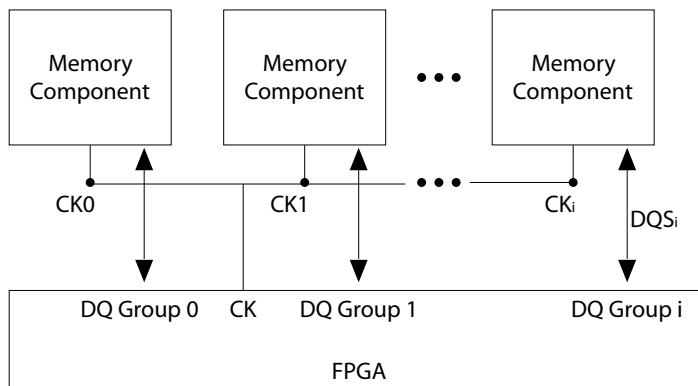
**Table 2-8: DDR2 SDRAM Layout Guidelines <sup>(1)</sup>**

Parameter	Guidelines
DIMMs	If you consider a normal DDR2 unbuffered, unregistered DIMM, essentially you are planning to perform the DIMM routing directly on your PCB. Therefore, each address and control pin routes from the FPGA (single pin) to all memory devices must be on the same side of the FPGA.
General Routing	<ul style="list-style-type: none"> <li>• All data, address, and command signals must have matched length traces <math>\pm 50</math> ps.</li> <li>• All signals within a given <b>Byte Lane Group</b> should be matched length with maximum deviation of <math>\pm 10</math> ps and routed in the same layer.</li> </ul>

Parameter	Guidelines
Clock Routing	<ul style="list-style-type: none"><li>• A 4.7 K-ohm resistor to ground is recommended for each Clock Enable signal. You can place the resistor at either the memory end or the FPGA end of the trace.</li><li>• Route clocks on inner layers with outer-layer run lengths held to under 500 mils (12.7 mm)</li><li>• These signals should maintain a 10-mil (0.254 mm) spacing from other nets</li><li>• Clocks should maintain a length-matching between clock pairs of <math>\pm 5</math> ps.</li><li>• Differential clocks should maintain a length-matching between P and N signals of <math>\pm 2</math> ps, routed in parallel.</li><li>• Space between different pairs should be at least three times the space between the differential pairs and must be routed differentially (5-mil trace, 10-15 mil space on centers), and equal to the signals in the Address/Command Group or up to 100 mils (2.54 mm) longer than the signals in the Address/Command Group.</li><li>• Trace lengths for CLK and DQS should closely match for each memory component. To match trace lengths on the board, a balanced tree topology is recommended for clock and address and command signal routing. For Stratix device families, ensure that Write Leveling tDQSS is passing in the DDR timing report; for Arria and Cyclone device families, verify that CK vs DQS timing is passing in the DDR timing report.</li></ul>
Address and Command Routing	<ul style="list-style-type: none"><li>• Unbuffered address and command lines are more susceptible to cross-talk and are generally noisier than buffered address or command lines. Therefore, un-buffered address and command signals should be routed on a different layer than data signals (DQ) and data mask signals (DM) and with greater spacing.</li><li>• Do not route differential clock (CK) and clock enable (CKE) signals close to address signals.</li></ul>

Parameter	Guidelines
DQ, DM, and DQS Routing Rules	<ul style="list-style-type: none"> <li>• Keep the distance from the pin on the DDR2 DIMM or component to the termination resistor pack (VTT) to less than 500 mils for <code>DQS[x]</code> Data Groups.</li> <li>• Keep the distance from the pin on the DDR2 DIMM or component to the termination resistor pack (VTT) to less than 1000 mils for the <code>ADR_CMD_CTL</code> Address Group.</li> <li>• Parallelism rules for the <code>DQS[x]</code> Data Groups are as follows: <ul style="list-style-type: none"> <li>• 4 mils for parallel runs &lt; 0.1 inch (approximately 1× spacing relative to plane distance)</li> <li>• 5 mils for parallel runs &lt; 0.5 inch (approximately 1× spacing relative to plane distance)</li> <li>• 10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance)</li> <li>• 15 mils for parallel runs between 1.0 and 6.0 inch (approximately 3× spacing relative to plane distance)</li> </ul> </li> <li>• Parallelism rules for the <code>ADR_CMD_CTL</code> group and <code>CLOCKS</code> group are as follows: <ul style="list-style-type: none"> <li>• 4 mils for parallel runs &lt; 0.1 inch (approximately 1× spacing relative to plane distance)</li> <li>• 10 mils for parallel runs &lt; 0.5 inch (approximately 2× spacing relative to plane distance)</li> <li>• 15 mils for parallel runs between 0.5 and 1.0 inches (approximately 3× spacing relative to plane distance)</li> <li>• 20 mils for parallel runs between 1.0 and 6.0 inches (approximately 4× spacing relative to plane distance)</li> </ul> </li> <li>• All signals are to maintain a 20-mil separation from other, non-related nets.</li> <li>• All signals must have a total length of &lt; 6 inches.</li> <li>• Trace lengths for CLK and DQS should closely match for each memory component. To match trace lengths on the board, a balanced tree topology is recommended for clock and address and command signal routing. For Stratix device families, ensure that Write Leveling tDQSS is passing in the DDR timing report; for Arria and Cyclone device families, verify that CK vs DQS timing is passing in the DDR timing report.</li> </ul>

Parameter	Guidelines
Termination Rules	<ul style="list-style-type: none"><li>When pull-up resistors are used, fly-by termination configuration is recommended. Fly-by helps reduce stub reflection issues.</li><li>Pull-ups should be within 0.5 to no more than 1 inch.</li><li>Pull up is typically 56-ohms.</li><li>If using resistor networks:<ul style="list-style-type: none"><li>Do not share R-pack series resistors between address/command and data lines (<math>D_Q</math>, <math>D_{QS}</math>, and <math>D_M</math>) to eliminate crosstalk within pack.</li><li>Series and pull up tolerances are 1–2%.</li><li>Series resistors are typically 10 to 20-ohm.</li></ul></li><li>Address and control series resistor typically at the FPGA end of the link.</li><li><math>D_M</math>, <math>D_{QS}</math>, <math>D_Q</math> series resistor typically at the memory end of the link (or just before the first DIMM).</li><li>If termination resistor packs are used:<ul style="list-style-type: none"><li>The distance to your memory device should be less than 750 mils.</li><li>The distance from your Altera's FPGA device should be less than 1250 mils.</li></ul></li></ul>
Quartus Prime Software Settings for Board Layout	<ul style="list-style-type: none"><li>To perform timing analyses on board and I/O buffers, use third party simulation tool to simulate all timing information such as skew, ISI, crosstalk, and type the simulation result into the UniPHY board setting panel.</li><li>Do not use advanced I/O timing model (AIOT) or board trace model unless you do not have access to any third party tool. AIOT provides reasonable accuracy but tools like HyperLynx provides better result. In operations with higher frequency, it is crucial to properly simulate all signal integrity related uncertainties.</li><li>The Quartus Prime software does timing check to find how fast the controller issues a write command after a read command, which limits the maximum length of the <math>D_Q/D_{QS}</math> trace. Check the turnaround timing in the Report DDR timing report and ensure the margin is positive before board fabrication. Functional failure happens if the margin is less than 0.</li></ul>
Note to Table:	
1. For point-to-point and DIMM interface designs, refer to the Micron website, <a href="http://www.micron.com">www.micron.com</a> .	

**Figure 2-9: Balanced Tree Topology**

$CK_i$  = Clock signal propagation delay to device  $i$

$DQS_i$  = DQ/DQS signals propagation delay to group  $i$

#### Related Information

- [External Memory Interface Spec Estimator](#)
- [www.micron.com](http://www.micron.com)
- [Leveling and Dynamic Termination](#) on page 2-2

## Layout Guidelines for DDR3 and DDR4 SDRAM Interfaces

The following table lists DDR3 and DDR4 SDRAM layout guidelines.

Unless otherwise specified, the guidelines in the following table apply to the following topologies:

- DIMM—UDIMM topology
- DIMM—RDIMM topology
- DIMM—LRDIMM topology
- Not all versions of the Quartus Prime software support LRDIMM.
- Discrete components laid out in UDIMM topology
- Discrete components laid out in RDIMM topology

These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface.

For information on the simulation flow for 28nm products, refer to [http://www.alterawiki.com/wiki/Measuring\\_Channel\\_Signal\\_Integrity](http://www.alterawiki.com/wiki/Measuring_Channel_Signal_Integrity).

For information on the simulation flow for Arria 10 products, refer to [http://www.alterawiki.com/wiki/Arria\\_10\\_EMIF\\_Simulation\\_Guidance](http://www.alterawiki.com/wiki/Arria_10_EMIF_Simulation_Guidance).

<http://www.altera.com/technology/memory/estimator/mem-emif-index.html>

For supported frequencies and topologies, refer to the *External Memory Interface Spec Estimator* <http://www.altera.com/technology/memory/estimator/mem-emif-index.html>.

For frequencies greater than 800 MHz, when you are calculating the delay associated with a trace, you must take the FPGA package delays into consideration. For more information, refer to *Package Deskew*.

For device families that do not support write leveling, refer to *Layout Guidelines for DDR2 SDRAM Interfaces*.

**Table 2-9: DDR3 and DDR4 SDRAM Layout Guidelines <sup>(1)</sup>**

Parameter	Guidelines
Decoupling Parameter	<ul style="list-style-type: none"><li>• Make VTT voltage decoupling close to the components and pull-up resistors.</li><li>• Connect decoupling caps between VTT and VDD using a 0.1F cap for every other VTT pin.</li><li>• Use a 0.1 uF cap and 0.01 uF cap for every VDDQ pin.</li></ul>
Maximum Trace Length <sup>(2)</sup>	<ul style="list-style-type: none"><li>• Even though there are no hard requirements for minimum trace length, you need to simulate the trace to ensure the signal integrity. Shorter routes result in better timing.</li><li>• For DIMM topology only:<ul style="list-style-type: none"><li>• Maximum trace length for all signals from FPGA to the first DIMM slot is 4.5 inches.</li><li>• Maximum trace length for all signals from DIMM slot to DIMM slot is 0.425 inches.</li></ul></li><li>• For discrete components only:<ul style="list-style-type: none"><li>• Maximum trace length for address, command, control, and clock from FPGA to the first component must not be more than 7 inches.</li><li>• Maximum trace length for DQ, DQS, DQS#, and DM from FPGA to the first component is 5 inches.</li></ul></li></ul>
General Routing	<ul style="list-style-type: none"><li>• Route over appropriate VCC and GND planes.</li><li>• Keep signal routing layers close to GND and power planes.</li></ul>
Spacing Guidelines	<ul style="list-style-type: none"><li>• Avoid routing two signal layers next to each other. Always make sure that the signals related to memory interface are routed between appropriate GND or power layers.</li><li>• For DQ/DQS/DM traces: Maintain at least 3H spacing between the edges (air-gap) for these traces. (Where H is the vertical distance to the closest return path for that particular trace.)</li><li>• For Address/Command/Control traces: Maintain at least 3H spacing between the edges (air-gap) these traces. (Where H is the vertical distance to the closest return path for that particular trace.)</li><li>• For Clock traces: Maintain at least 5H spacing between two clock pair or a clock pair and any other memory interface trace. (Where H is the vertical distance to the closest return path for that particular trace.)</li></ul>

Parameter	Guidelines
Clock Routing	<ul style="list-style-type: none"> <li>Route clocks on inner layers with outer-layer run lengths held to under 500 mils (12.7 mm).</li> <li>Route clock signals in a daisy chain topology from the first SDRAM to the last SDRAM. The maximum length of the first SDRAM to the last SDRAM must not exceed 0.69 tCK for DDR3 and 1.5 tCK for DDR4. For different DIMM configurations, check the appropriate JEDEC specification.</li> <li>These signals should maintain the following spacings:</li> <li>Clocks should maintain a length-matching between clock pairs of <math>\pm 5</math> ps.</li> <li>Clocks should maintain a length-matching between positive (<math>p</math>) and negative (<math>n</math>) signals of <math>\pm 2</math> ps, routed in parallel.</li> <li>Space between different pairs should be at least two times the trace width of the differential pair to minimize loss and maximize interconnect density.</li> <li>To avoid mismatched transmission line to via, Altera recommends that you use Ground Signal Signal Ground (GSSG) topology for your clock pattern—GND CLKP CKLN GND.</li> <li>Route all addresses and commands to match the clock signals to within <math>\pm 20</math> ps to each discrete memory component. Refer to the following figure.</li> </ul>
Address and Command Routing	<ul style="list-style-type: none"> <li>Route address and command signals in a daisy chain topology from the first SDRAM to the last SDRAM. The maximum length of the first SDRAM to the last SDRAM must not be more than 0.69 tCK for DDR3 and 1.5 tCK for DDR4. For different DIMM configurations, check the appropriate JEDEC specifications.</li> <li>UDIMMs are more susceptible to cross-talk and are generally noisier than buffered DIMMs. Therefore, route address and command signals of UDIMMs on a different layer than data signals (<math>DQ</math>) and data mask signals (<math>DM</math>) and with greater spacing.</li> <li>Do not route differential clock (<math>CK</math>) and clock enable (<math>CKE</math>) signals close to address signals.</li> <li>Route all addresses and commands to match the clock signals to within <math>\pm 20</math> ps to each discrete memory component. Refer to the following figure.</li> </ul>

Parameter	Guidelines
DQ, DM, and DQS Routing Rules	<ul style="list-style-type: none"><li>All the trace length matching requirements are from the FPGA package ball to the SDRAM package ball, which means you must consider trace mismatching on different DIMM raw cards.</li><li>Match in length all DQ, DQS, and DM signals within a given byte-lane group with a maximum deviation of <math>\pm 10</math> ps.</li><li>Ensure to route all DQ, DQS, and DM signals within a given byte-lane group on the same layer to avoid layer to layer transmission velocity differences, which otherwise increase the skew within the group.</li><li>Do not count on FPGAs to deskew for more than 20 ps of DQ group skew. The skew algorithm only removes the following possible uncertainties:<ul style="list-style-type: none"><li>Minimum and maximum die IOE skew or delay mismatch</li><li>Minimum and maximum device package skew or mismatch</li><li>Board delay mismatch of 20 ps</li><li>Memory component DQ skew mismatch</li><li>Increasing any of these four parameters runs the risk of the deskew algorithm limiting, failing to correct for the total observed system skew. If the algorithm cannot compensate without limiting the correction, timing analysis shows reduced margins.</li></ul></li><li>For memory interfaces with leveling, the timing between the DQS and clock signals on each device calibrates dynamically to meet tDQSS. To make sure the skew is not too large for the leveling circuit's capability, follow these rules:<ul style="list-style-type: none"><li>Propagation delay of clock signal must not be shorter than propagation delay of DQS signal at every device: <math>(CK_i) - DQS_i &gt; 0; 0 &lt; i &lt;</math> number of components – 1 . For DIMMs, ensure that the CK trace is longer than the longest DQS trace at the DIMM connector.</li><li>Total skew of CLK and DQS signal between groups is less than one clock cycle: <math>(CK_i + DQS_i)_{\text{max}} - (CK_i + DQS_i)_{\text{min}} &lt; 1 \times tCK</math>(If you are using a DIMM topology, your delay and skew must take into consideration values for the actual DIMM.)</li></ul></li></ul>
Spacing Guidelines	<ul style="list-style-type: none"><li>Avoid routing two signal layers next to each other. Always ensure that the signals related to the memory interface are routed between appropriate GND or power layers.</li><li>For DQ/DQS/DM traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace.</li><li>For Address/Command/Control traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace.</li><li>For Clock traces: Maintain at least 5H spacing between two clock pairs or a clock pair and any other memory interface trace, where H is the vertical distance to the closest return path for that particular trace.</li></ul>

Parameter	Guidelines
Quartus Prime Software Settings for Board Layout	<ul style="list-style-type: none"> <li>To perform timing analyses on board and I/O buffers, use third party simulation tool to simulate all timing information such as skew, ISI, crosstalk, and type the simulation result into the UniPHY board setting panel.</li> <li>Do not use advanced I/O timing model (AIOT) or board trace model unless you do not have access to any third party tool. AIOT provides reasonable accuracy but tools like HyperLynx provide better results.</li> </ul>

Notes to Table:

- For point-to-point and DIMM interface designs, refer to the Micron website, [www.micron.com](http://www.micron.com).
- For better efficiency, the UniPHY IP requires faster turnarounds from read commands to write.

#### Related Information

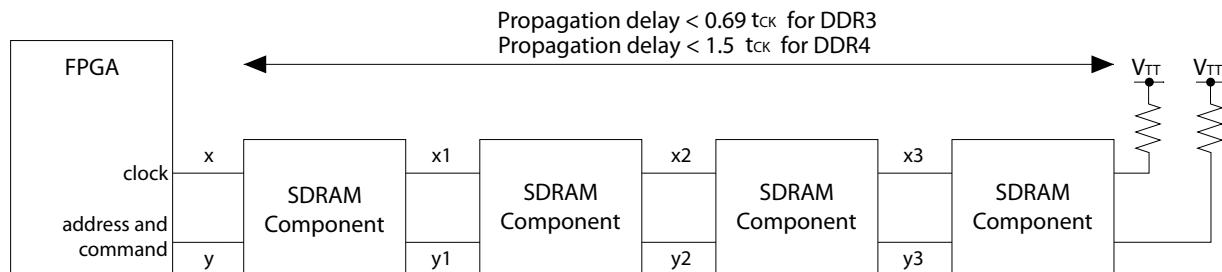
- [Layout Guidelines for DDR2 SDRAM Interface](#) on page 2-26
- [Package Deskeew](#) on page 2-39
- [External Memory Interface Spec Estimator](#)
- [www.micron.com](http://www.micron.com)

## Length Matching Rules

The following topics provide guidance on length matching for different types of DDR3 signals.

Route all addresses and commands to match the clock signals to within  $\pm 20$  ps to each discrete memory component. The following figure shows the DDR3 and DDR4 SDRAM component routing guidelines for address and command signals.

**Figure 2-10: DDR3 and DDR4 SDRAM Component Address and Command Routing Guidelines**



If using discrete components:

$$x = y \pm 20 \text{ ps}$$

$$x + x_1 = y + y_1 \pm 20 \text{ ps}$$

$$x + x_1 + x_2 = y + y_1 + y_2 \pm 20 \text{ ps}$$

$$x + x_1 + x_2 + x_3 = y + y_1 + y_2 + y_3 \pm 20 \text{ ps}$$

If using a DIMM topology:

$$x = y \pm 20 \text{ ps}$$

The timing between the DQS and clock signals on each device calibrates dynamically to meet tDQSS. The following figure shows the delay requirements to align DQS and clock signals. To ensure that the skew is not too large for the leveling circuit's capability, follow these rules:

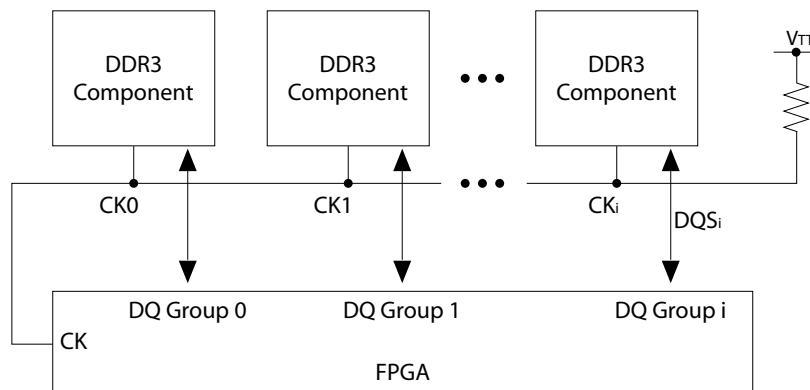
- Propagation delay of clock signal must not be shorter than propagation delay of DQS signal at every device:

$$CK_i - DQS_i > 0; 0 < i < \text{number of components} - 1$$

- Total skew of CLK and DQS signal between groups is less than one clock cycle:

$$(CK_i + DQS_i)_{\max} - (CK_i + DQS_i)_{\min} < 1 \times t_{CK}$$

**Figure 2-11: Delaying DQS Signal to Align DQS and Clock**



$CK_i$  = Clock signal propagation delay to device  $i$

$DQS_i$  = DQ/DQS signals propagation delay to group  $i$

**Clk pair matching**—If you are using a DIMM (UDIMM, RDIMM, or LRDIMM) topology, match the trace lengths up to the DIMM connector. If you are using discrete components, match the lengths for all the memory components connected in the fly-by chain.

**DQ group length matching**—If you are using a DIMM (UDIMM, RDIMM, or LRDIMM) topology, match the trace lengths up to the DIMM connector. If you are using discrete components, match the lengths up to the respective memory components.

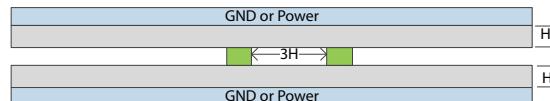
When you are using DIMMs, it is assumed that lengths are tightly matched within the DIMM itself. You should check that appropriate traces are length-matched within the DIMM.

## Spacing Guidelines

This topic provides recommendations for minimum spacing between board traces for various signal traces.

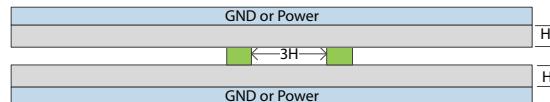
### Spacing Guidelines for DQ, DQS, and DM Traces

Maintain a minimum of  $3H$  spacing between the edges (air-gap) of these traces. (Where  $H$  is the vertical distance to the closest return path for that particular trace.)



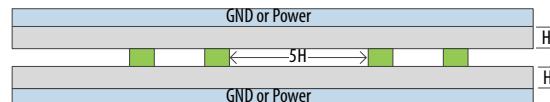
### Spacing Guidelines for Address and Command and Control Traces

Maintain at least  $3H$  spacing between the edges (air-gap) of these traces. (Where  $H$  is the vertical distance to the closest return path for that particular trace.)



### Spacing Guidelines for Clock Traces

Maintain at least  $5H$  spacing between two clock pair or a clock pair and any other memory interface trace. (Where  $H$  is the vertical distance to the closest return path for that particular trace.)



## Layout Guidelines for DDR3 SDRAM Wide Interface (>72 bits)

The following topics discuss different ways to lay out a wider DDR3 SDRAM interface to the FPGA. Choose the topology based on board trace simulation and the timing budget of your system.

The UniPHY IP supports up to a 144-bit wide DDR3 interface. You can either use discrete components or DIMMs to implement a wide interface (any interface wider than 72 bits). Altera recommends using leveling when you implement a wide interface with DDR3 components.

When you lay out for a wider interface, all rules and constraints discussed in the previous sections still apply. The DQS, DQ, and DM signals are point-to-point, and all the same rules discussed in “Design Layout Guidelines” apply.

The main challenge for the design of the fly-by network topology for the clock, command, and address signals is to avoid signal integrity issues, and to make sure you route the DQS, DQ, and DM signals with the chosen topology.

#### Related Information

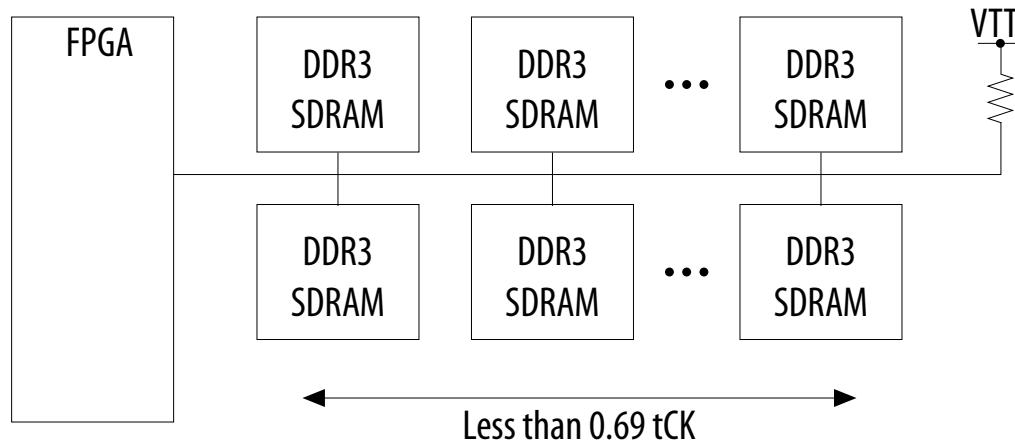
[Design Layout Guidelines](#) on page 2-24

### Fly-By Network Design for Clock, Command, and Address Signals

The UniPHY IP requires the flight-time skew between the first DDR3 SDRAM component and the last DDR3 SDRAM component to be less than 0.69 tCK for memory clocks. This constraint limits the number of components you can have for each fly-by network.

If you design with discrete components, you can choose to use one or more fly-by networks for the clock, command, and address signals.

The following figure shows an example of a single fly-by network topology.

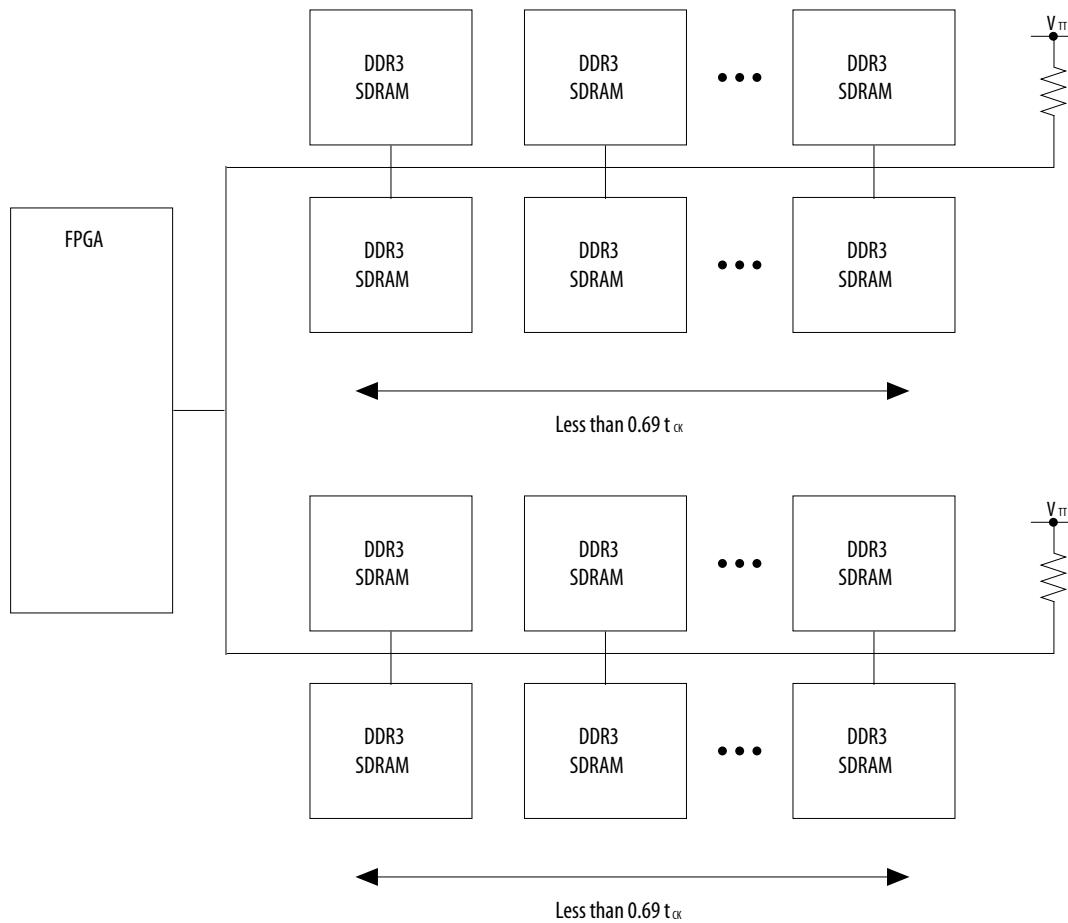
**Figure 2-12: Single Fly-By Network Topology**

Every DDR3 SDRAM component connected to the signal is a small load that causes discontinuity and degrades the signal. When using a single fly-by network topology, to minimize signal distortion, follow these guidelines:

- Use  $\times 16$  device instead  $\times 4$  or  $\times 8$  to minimize the number of devices connected to the trace.
- Keep the stubs as short as possible.
- Even with added loads from additional components, keep the total trace length short; keep the distance between the FPGA and the first DDR3 SDRAM component less than 5 inches.
- Simulate clock signals to ensure a decent waveform.

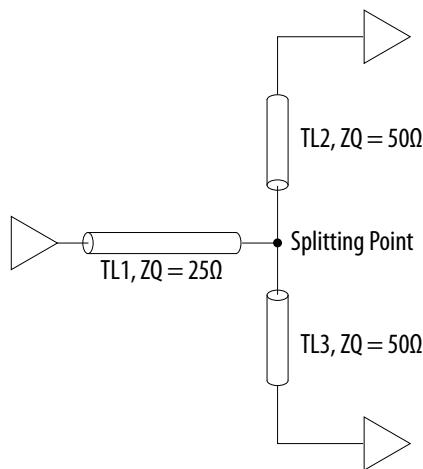
The following figure shows an example of a double fly-by network topology. This topology is not rigid but you can use it as an alternative option. The advantage of using this topology is that you can have more DDR3 SDRAM components in a system without violating the 0.69 tCK rule. However, as the signals branch out, the components still create discontinuity.

Figure 2-13: Double Fly-By Network Topology



You must perform simulations to find the location of the split, and the best impedance for the traces before and after the split.

The following figure shows a way to minimize the discontinuity effect. In this example, keep TL2 and TL3 matches in length. Keep TL1 longer than TL2 and TL3, so that it is easier to route all the signals during layout.

**Figure 2-14: Minimizing Discontinuity Effect**

You can also consider using a DIMM on each branch to replace the components. Because the trade impedance on the DIMM card is 40-ohm to 60-ohm, perform a board trace simulation to control the reflection to within the level your system can tolerate.

By using the new features of the DDR3 SDRAM controller with UniPHY and the Stratix III, Stratix IV, or Stratix V devices, you simplify your design process. Using the fly-by daisy chain topology increases the complexity of the datapath and controller design to achieve leveling, but also greatly improves performance and eases board layout for DDR3 SDRAM.

You can also use the DDR3 SDRAM components without leveling in a design if it may result in a more optimal solution, or use with devices that support the required electrical interface standard, but do not support the required read and write leveling functionality.

## Package Deskew

Trace lengths inside the device package are not uniform for all package pins. The nonuniformity of package traces can affect system timing for high frequencies. In the Quartus II software version 12.0 and later, and the Quartus Prime software, a package deskew option is available.

If you do not enable the package deskew option, the Quartus Prime software uses the package delay numbers to adjust skews on the appropriate signals; you do not need to adjust for package delays on the board traces. If you do enable the package deskew option, the Quartus Prime software does not use the package delay numbers for timing analysis, and you must deskew the package delays with the board traces for the appropriate signals for your design.

## Package Deskew Recommendation for Stratix V Devices

Package deskew is not required for any memory protocol operating at 800 MHz or below.

For DDR3 and RLDRAM3 designs operating above 800 MHz, you should run timing analysis with accurately entered board skew parameters in the parameter editor. If **Report DDR** reports non-core timing violations, you should then perform the steps in the following topics, and modify your board layout. Package deskew is not required for any protocols other than DDR3 and RLDRAM 3.

## DQ/DQS/DM Deskew

To get the package delay information, follow these steps:

1. Select the **FPGA DQ/DQS Package Skews Deskewed on Board** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Run **Analysis and Synthesis** in the Quartus Prime software. (Skip this step if you are using an Arria 10 device.)
5. Run the `<core_name>.p0_pin_assignment.tcl` script. (Skip this step if you are using an Arria 10 device.)
6. Compile your design.
7. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

## Address and Command Deskew

Deskew address and command delays as follows:

1. Select the **FPGA Address/Command Package Skews Deskewed on Board** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Run **Analysis and Synthesis** in the Quartus Prime software. (Skip this step if you are using an Arria 10 device.)
5. Run the `<core_name>.p0_pin_assignment.tcl` script. (Skip this step if you are using an Arria 10 device.)
6. Compile your design.
7. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

## Package Deskew Recommendations for Arria 10 Devices

The following table shows package deskew recommendations for all protocols supported on Arria 10 devices.

As operating frequencies increase, it becomes increasingly critical to perform package deskew. The frequencies listed in the table are the *minimum* frequencies for which you must perform package deskew.

If you plan to use a listed protocol at the specified frequency or higher, you must perform package deskew. For example, you must perform package deskew if you plan to use dual-rank DDR4 at 800 MHz or above.

Protocol	Minimum Frequency (MHz) for Which to Perform Package Deskew		
	Single Rank	Dual Rank	Quad Rank
DDR4	933	800	667
DDR3	933	800	667
LPDDR3	667	533	Not required
QDR IV	933	Not applicable	Not applicable
RLLDRAM 3	933	667	Not applicable

Protocol	Minimum Frequency (MHz) for Which to Perform Package Deskew		
	Single Rank	Dual Rank	Quad Rank
RLDRAM II	Not required	Not applicable	Not applicable
QDR II, II+, II+ Xtreme	Not required	Not applicable	Not applicable

**Note:** QDR IV SRAM for Arria 10 EMIF IP will be supported in a future release of the Quartus Prime software.

The recommendations in the above table are based on preliminary timing models, and may be updated in the future. If you are designing a board with Arria 10 devices and require exact package trace delays, contact Altera Support.

## Deskew Example

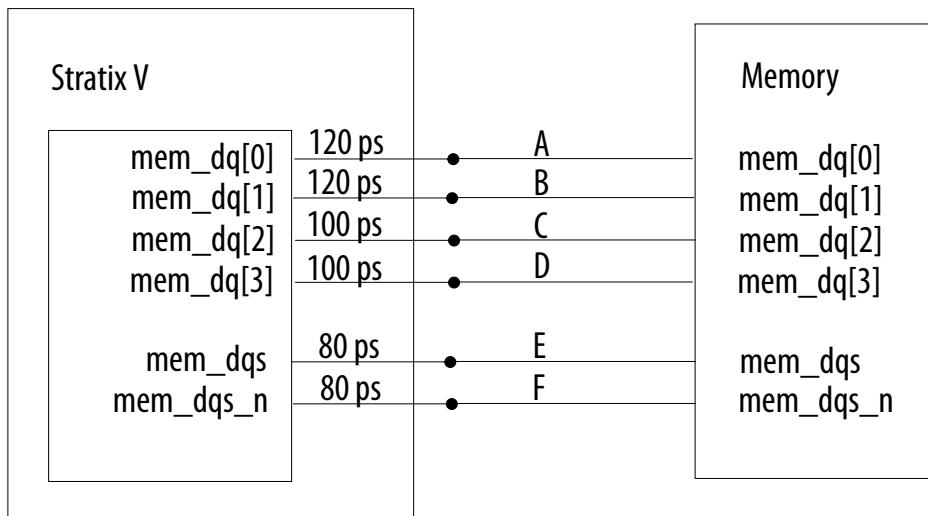
Consider an example where you want to deskew an interface with 4 DQ pins, 1 DQS pin, and 1 DQSn pin.

Let's assume an operating frequency of 667 MHz, and the package lengths for the pins reported in the .pin file as follows:

```
dq[ 0 ] = 120 ps
dq[ 1 ] = 120 ps
dq[ 2 ] = 100 ps
dq[ 3 ] = 100 ps
dqs     = 80 ps
dqs_n   = 80 ps
```

The following figure illustrates this example.

Figure 2-15: Deskew Example

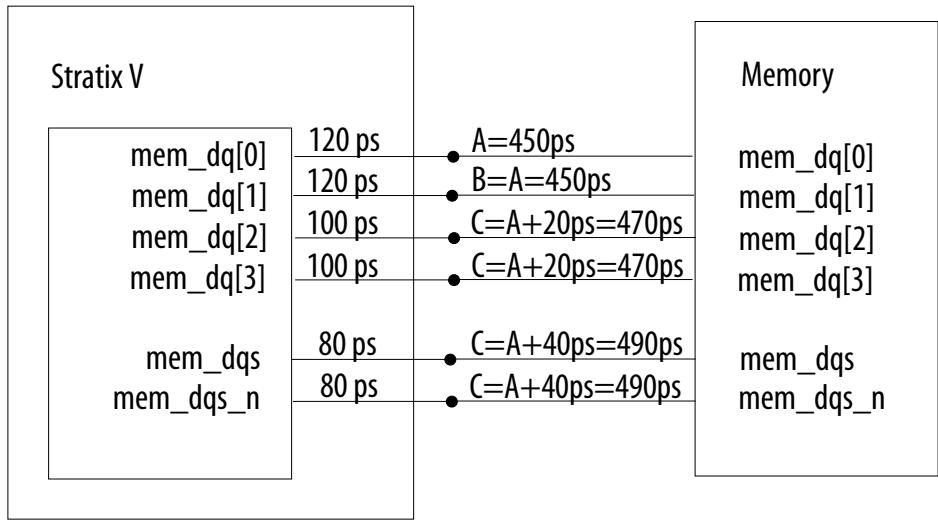


When you perform length matching for all the traces in the DQS group, you must take package delays into consideration. Because the package delays of traces A and B are 40 ps longer than the package delays of traces E and F, you would need to make the board traces for E and F 40 ps longer than the board traces for A and B.

A similar methodology would apply to traces C and D, which should be 20 ps longer than the lengths of traces A and B.

The following figure shows this scenario with the length of trace A at 450 ps.

**Figure 2-16: Deskew Example with Trace Delay Calculations**



When you enter the board skews into the Board Settings tab of the DDR3 parameter editor, you should calculate the board skew parameters as the sums of board delay and corresponding package delay. If a pin does not have a package delay (such as address and command pins), you should use the board delay only.

The example of the preceding figure shows an ideal case where board skews are perfectly matched. In reality, you should allow plus or minus 10 ps of skew mismatch within a DQS group (DQ/DQS/DM).

## Package Migration

Package delays can be different for the same pin in different packages. If you want to use multiple migratable packages in your system, you should compensate for package skew as described in this topic.

Assume two migratable packages, device A and device B, and that you want to compensate for the board trace lengths for device A. Follow these steps:

1. Compile your design for device A, with the Package Skew option enabled.
2. Note the skews in the `<core_name>.pin` file for device A. Deskew these package skews with board trace lengths as described in the preceding examples.
3. Recompile your design for device A.
4. For Device B open the parameter editor and deselect Package Deskew option.
5. Calculate board skew parameters only taking into account the board traces for Device B and enter that value into the parameter editor for Device B.
6. Regenerate the IP and recompile the design for Device B.
7. Verify that timing requirements are met for both device A and device B.

## Package Deskew for RLDRAM II and RLDRAM 3

You should follow Altera's package deskew guidance if you are using Stratix V or Arria 10 devices.

For more information on package deskew, refer to *Package Deskew*.

**Related Information**  
**Package Deskew**

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"><li>Minor change to <i>Clock Routing</i> description in the <i>DDR2 SDRAM Layout Guidelines</i> table in <i>Layout Guidelines for DDR2 SDRAM Interface</i>.</li><li>Added maximum length of the first SDRAM to the last SDRAM for clock routing and address and command routing for DDR4, in <i>Layout Guidelines for DDR3 and DDR4 SDRAM Interfaces</i>.</li><li>Removed <i>DRAM Termination Guidance</i> from <i>Layout Guidelines for DDR3 and DDR4 SDRAM Interfaces</i>.</li><li>Added DDR4 support to <i>Length Matching Rules</i>.</li></ul>
November 2015	2015.11.02	<ul style="list-style-type: none"><li>Minor additions to procedure steps in <i>DQ/DQS/DM Deskew</i> and <i>Address and Command Deskew</i>.</li><li>Added reference to Micron Technical Note in <i>Layout Guidelines for DDR3 and DDR4 SDRAM Interfaces</i>.</li><li>Changed title of <i>Board Termination for DDR2 SDRAM</i> to <i>Termination for DDR2 SDRAM</i> and <i>Board Termination for DDR3 SDRAM</i> to <i>Termination for DDR3 SDRAM</i>.</li><li>Changed title of <i>Leveling and Dynamic ODT</i> to <i>Leveling and Dynamic Termination</i>.</li><li>Added DDR4 support in <i>Dynamic ODT</i>.</li><li>Removed topics pertaining to older device families.</li><li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li></ul>
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	<ul style="list-style-type: none"><li>Added MAX 10 to <i>On-Chip Termination</i> topic.</li><li>Added MAX 10 to <i>Termination Recommendations</i> table in <i>Recommended Termination Schemes</i> topic.</li></ul>

Date	Version	Changes
August 2014	2014.08.15	<ul style="list-style-type: none"> <li>Added Arria V Soc and Cyclone V SoC devices to note in <i>Leveling and Dynamic ODT</i> section.</li> <li>Added DDR4 to <i>Read and Write Leveling</i> section.</li> <li>Revised text in <i>On-Chip Termination</i> section.</li> <li>Added text to note in <i>Board Termination for DDR3 SDRAM</i> section.</li> <li>Added <i>Layout Approach</i> information in the <i>DDR3 and DDR4 on Arria 10 Devices</i> section.</li> <li>Recast expressions of length-matching measurements throughout <i>DDR2 SDRAM Layout Guidelines</i> table.</li> <li>Made several changes to <i>DDR3 and DDR4 SDRAM Layout Guidelines</i> table:           <ul style="list-style-type: none"> <li>Added <i>Spacing Guidelines</i> section.</li> <li>Removed millimeter approximations from lengths expressed in picoseconds.</li> <li>Revised Guidelines for <i>Clock Routing, Address and Command Routing</i>, and <i>DQ, DM, and DQS Routing Rules</i> sections.</li> </ul> </li> <li>Added <i>Spacing Guidelines</i> information to <i>Design Layout Guidelines</i> section.</li> </ul>
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>Review and minor updates of content.</li> <li>Consolidated General Layout Guidelines.</li> <li>Added DDR3 and DDR4 information for Arria 10 devices.</li> <li>Updated chapter title to include DDR4 support.</li> <li>Removed references to ALTMEMPHY.</li> <li>Removed references to Cyclone III and Cyclone IV devices.</li> <li>Removed references to Stratix II devices.</li> <li>Corrected Vtt to Vdd in <i>Memory Clocks for DDR3 SDRAM UDIMM</i> section.</li> </ul>
November 2012	5.0	<ul style="list-style-type: none"> <li>Updated <i>Layout Guidelines for DDR2 SDRAM Interface</i> and <i>Layout Guidelines for DDR3 SDRAM Interface</i>.</li> <li>Added LRDIMM support.</li> <li>Added Package Deskew section.</li> </ul>
June 2012	4.1	Added Feedback icon.
November 2011	4.0	Added Arria V and Cyclone V information.
June 2011	3.0	<ul style="list-style-type: none"> <li>Merged DDR2 and DDR3 chapters to <i>DDR2 and DDR3 SDRAM Interface Termination and Layout Guidelines</i> and updated with leveling information.</li> <li>Added Stratix V information.</li> </ul>

Date	Version	Changes
December 2010	2.1	Added <i>DDR3 SDRAM Interface Termination, Drive Strength, Loading, and Board Layout Guidelines</i> chapter with Stratix V information.
July 2010	2.0	Updated Arria II GX information.
April 2010	1.0	Initial release.

# Dual-DIMM DDR2 and DDR3 SDRAM Board Design Guidelines

3

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The following topics describe guidelines for implementing dual unbuffered DIMM (UDIMM) DDR2 and DDR3 SDRAM interfaces.

The following topics discuss the impact on signal integrity of the data signal with the following conditions in a dual-DIMM configuration:

- Populating just one slot versus populating both slots
- Populating slot 1 versus slot 2 when only one DIMM is used
- On-die termination (ODT) setting of 75-ohm versus an ODT setting of 150-ohm

For detailed information about a single-DIMM DDR2 SDRAM interface, refer to the *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

## Related Information

[DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines](#) on page 2-1

## General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the slew rate and propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

**Table 3-1: General Layout Guidelines**

Parameter	Guidelines
Impedance	<ul style="list-style-type: none"><li>• All unused via pads must be removed, because they cause unwanted capacitance.</li><li>• Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.</li></ul>

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

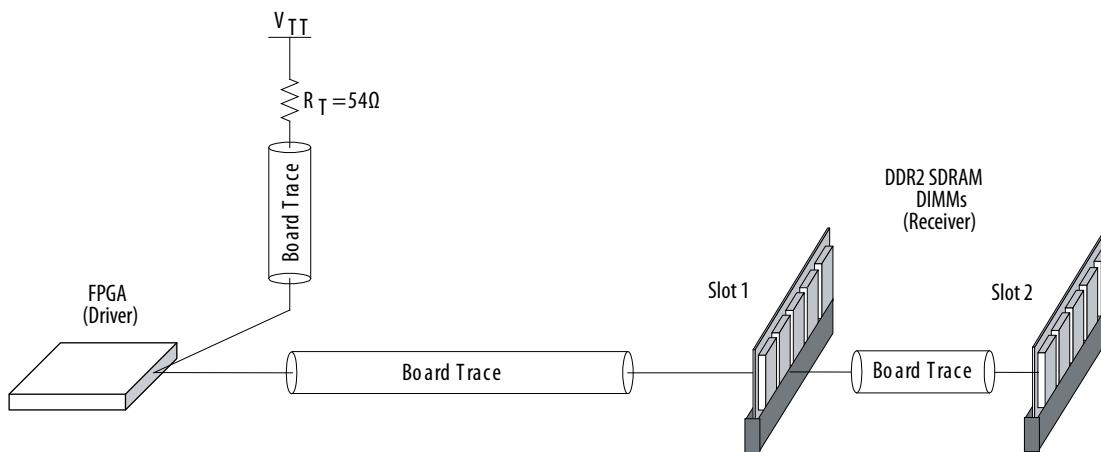
Parameter	Guidelines
Decoupling Parameter	<ul style="list-style-type: none"> <li>• Use 0.1 uF in 0402 size to minimize inductance</li> <li>• Make VTT voltage decoupling close to pull-up resistors</li> <li>• Connect decoupling caps between VTT and ground</li> <li>• Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin</li> <li>• Verify the capacitive decoupling using the Altera Power Distribution Network Design Tool</li> </ul>
Power	<ul style="list-style-type: none"> <li>• Route GND and V<sub>CC</sub> as planes</li> <li>• Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation</li> <li>• Route VTT as islands or 250-mil (6.35-mm) power traces</li> <li>• Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces</li> </ul>
General Routing	<p>All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Altera recommend that signals from the same net group always be routed on the same layer.</p> <ul style="list-style-type: none"> <li>• Use 45° angles (<i>not</i> 90° corners)</li> <li>• Avoid T-Junctions for critical nets or clocks</li> <li>• Avoid T-junctions greater than 250 mils (6.35 mm)</li> <li>• Disallow signals across split planes</li> <li>• Restrict routing other signals close to system reset signals</li> <li>• Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks</li> </ul>

**Related Information**[Power Distribution Network Design Tool](#)

## Dual-Slot Unbuffered DDR2 SDRAM

This topic describes guidelines for implementing a dual slot unbuffered DDR2 SDRAM interface, operating at up to 400-MHz and 800-Mbps data rates.

The following figure shows a typical DQS, DQ, and DM signal topology for a dual-DIMM interface configuration using the ODT feature of the DDR2 SDRAM components.

**Figure 3-1: Dual-DIMM DDR2 SDRAM Interface Configuration**

The simulations in this section use a Stratix® II device-based board. Because of limitations of this FPGA device family, simulations are limited to 266 MHz and 533 Mbps so that comparison to actual hardware results can be directly made.

## Overview of ODT Control

When there is only a single-DIMM on the board, the ODT control is relatively straightforward. During write to the memory, the ODT feature of the memory is turned on; during read from the memory, the ODT feature of the memory is turned off. However, when there are multiple DIMMs on the board, the ODT control becomes more complicated.

With a dual-DIMM interface on the system, the controller has different options for turning the memory ODT on or off during read or write. The following table lists the DDR2 SDRAM ODT control during write to the memory. These DDR2 SDRAM ODT controls are recommended by Samsung Electronics. The JEDEC DDR2 specification was updated to include optional support for  $R_{TT}(\text{nominal}) = 50\text{-ohm}$ .

For more information about the DDR2 SDRAM ODT controls recommended by Samsung, refer to the *Samsung DDR2 Application Note: ODT (On Die Termination) Control*.

**Table 3-2: DDR2 SDRAM ODT Control—Writes<sup>(1)</sup>**

Slot 1 <sup>(2)</sup>	Slot 2 <sup>(2)</sup>	Write To	FPGA	Module in Slot 1		Module in Slot 2	
				Rank 1	Rank 2	Rank 3	Rank 4
DR	DR	Slot 1	Series 50-ohms	Infinite	Infinite	75 or 50-ohm	Infinite
		Slot 2	Series 50-ohms	75 or 50-ohm	Infinite	Infinite	Infinite

Slot 1 <sup>(2)</sup>	Slot 2 <sup>(2)</sup>	Write To	FPGA	Module in Slot 1		Module in Slot 2	
				Rank 1	Rank 2	Rank 3	Rank 4
SR	SR	Slot 1	Series 50-ohms	Infinite	Unpopulated	75 or 50-ohm	Unpopulated
		Slot 2	Series 50-ohms	75 or 50-ohm	Unpopulated	Infinite	Unpopulated
DR	Empty	Slot 1	Series 50-ohms	150-ohm	Infinite	Unpopulated	Unpopulated
Empty	DR	Slot 2	Series 50-ohms	Unpopulated	Unpopulated	150-ohm	Infinite
SR	Empty	Slot 1	Series 50-ohms	150-ohm	Unpopulated	Unpopulated	Unpopulated
Empty	SR	Slot 2	Series 50-ohms	Unpopulated	Unpopulated	150-ohm	Unpopulated

Notes to Table:

1. For DDR2 at 400 MHz and 533 Mbps = 75-ohm; for DDR2 at 667 MHz and 800 Mbps = 50-ohm.
2. SR = single ranked; DR = dual ranked.

**Table 3-3: DDR2 SDRAM ODT Control—Reads <sup>(1)</sup>**

Slot 1 <sup>(2)</sup>	Slot 2 <sup>(2)</sup>	Read From	FPGA	Module in Slot 1		Module in Slot 2	
				Rank 1	Rank 2	Rank 3	Rank 4
DR	DR	Slot 1	Parallel 50-ohms	Infinite	Infinite	75 or 50-ohm	Infinite
		Slot 2	Parallel 50-ohms	75 or 50-ohm	Infinite	Infinite	Infinite
SR	SR	Slot 1	Parallel 50-ohms	Infinite	Unpopulated	75 or 50-ohm	Unpopulated
		Slot 2	Parallel 50-ohms	75 or 50-ohm	Unpopulated	Infinite	Unpopulated
DR	Empty	Slot 1	Parallel 50-ohms	Infinite	Infinite	Unpopulated	Unpopulated
Empty	DR	Slot 2	Parallel 50-ohms	Unpopulated	Unpopulated	Infinite	Infinite

Slot 1 <sup>(2)</sup>	Slot 2 <sup>(2)</sup>	Read From	FPGA	Module in Slot 1		Module in Slot 2	
				Rank 1	Rank 2	Rank 3	Rank 4
SR	Empty	Slot 1	Parallel 50-ohms	Infinite	Unpopulated	Unpopulated	Unpopulated
Empty	SR	Slot 2	Parallel 50-ohms	Unpopulated	Unpopulated	Infinite	Unpopulated

Notes to Table:

1. For DDR2 at 400 MHz and 533 Mbps = 75-ohm; for DDR2 at 667 MHz and 800 Mbps = 50-ohm.
2. SR = single ranked; DR = dual ranked.

## DIMM Configuration

Although populating both memory slots is common in a dual-DIMM memory system, there are some instances when only one slot is populated.

For example, some systems are designed to have a certain amount of memory initially and as applications get more complex, the system can be easily upgraded to accommodate more memory by populating the second memory slot without re-designing the system. The following topics discuss a dual-DIMM system where the dual-DIMM system only has one slot populated at one time and a dual-DIMM system where both slots are populated. ODT controls recommended by memory vendors, as well as other possible ODT settings are evaluated for usefulness in an FPGA system.

### Dual-DIMM Memory Interface with Slot 1 Populated

The following topics focus on a dual-DIMM memory interface where slot 1 is populated and slot 2 is unpopulated.

These topics examine the impact on the signal quality due to an unpopulated DIMM slot and compares it to a single-DIMM memory interface.

#### Related Information

#### [ODT Generation Logic](#)

### FPGA Writing to Memory

In the DDR2 SDRAM, the ODT feature has two settings: 150-ohms and 75-ohms.

The recommended ODT setting for a dual DIMM configuration with one slot occupied is 150-ohm.

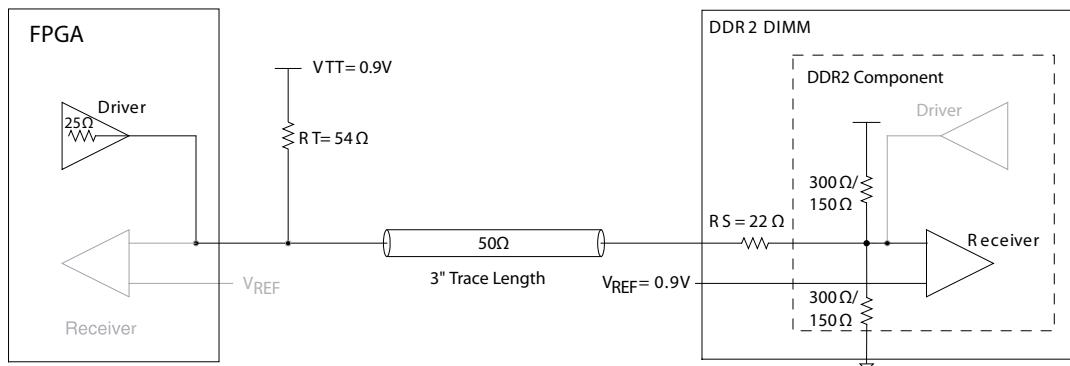
**Note:** On DDR2 SDRAM devices running at 333 MHz/667 Mbps and above, the ODT feature supports an additional setting of 50-ohm.

Refer to the respective memory decathlete for additional information about the ODT settings in DDR2 SDRAM devices.

### Write to Memory Using an ODT Setting of 150-ohm

The following figure shows a double parallel termination scheme (Class II) using ODT on the memory with a memory-side series resistor when the FPGA is writing to the memory using a 25-ohm OCT drive strength setting on the FPGA.

**Figure 3-2: Double Parallel Termination Scheme (Class II) Using ODT on DDR2 SDRAM DIMM with Memory-Side Series Resistor**



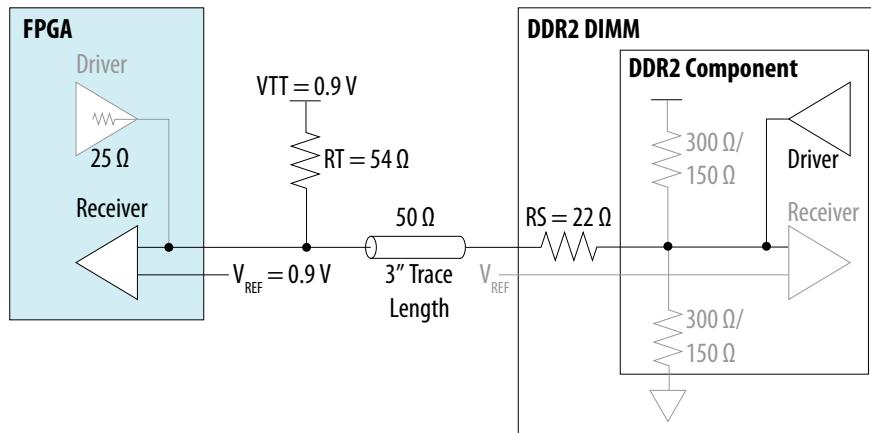
#### Related Information

[DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines](#) on page 2-1

### Reading from Memory

During read from the memory, the ODT feature is turned off. Thus, there is no difference between using an ODT setting of 150-ohm and 75-ohm. As such, the termination scheme becomes a single parallel termination scheme (Class I) where there is an external resistor on the FPGA side and a series resistor on the memory side as shown in the following figure.

**Figure 3-3: Single Parallel Termination Scheme (Class I) Using External Resistor and Memory-Side Series Resistor**



#### Related Information

[DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines](#) on page 2-1

### Dual-DIMM with Slot 2 Populated

The following topics focus on a dual-DIMM memory interface where slot 2 is populated and slot 1 is unpopulated. Specifically, these topics discuss the impact of location of the DIMM on the signal quality.

## Related Information

### ODT Generation Logic

## FPGA Writing to Memory

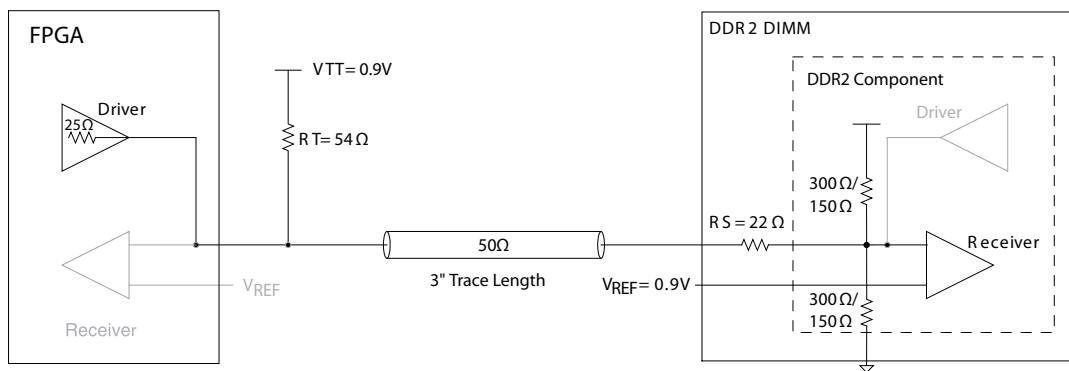
The following topics explore the differences between populating slot 1 and slot 2 of the dual-DIMM memory interface.

Previous topics focused on the dual-DIMM memory interface where slot 1 is populated resulting in the memory being located closer to the FPGA. When slot 2 is populated, the memory is located further away from the FPGA, resulting in additional trace length that potentially affects the signal quality seen by the memory. The following topics explore the differences between populating slot 1 and slot 2 of the dual-DIMM memory interface.

### Write to Memory Using an ODT Setting of 150-ohm

The following figure shows the double parallel termination scheme (Class II) using ODT on the memory with the memory-side series resistor when the FPGA is writing to the memory using a 25-ohm OCT drive strength setting on the FPGA.

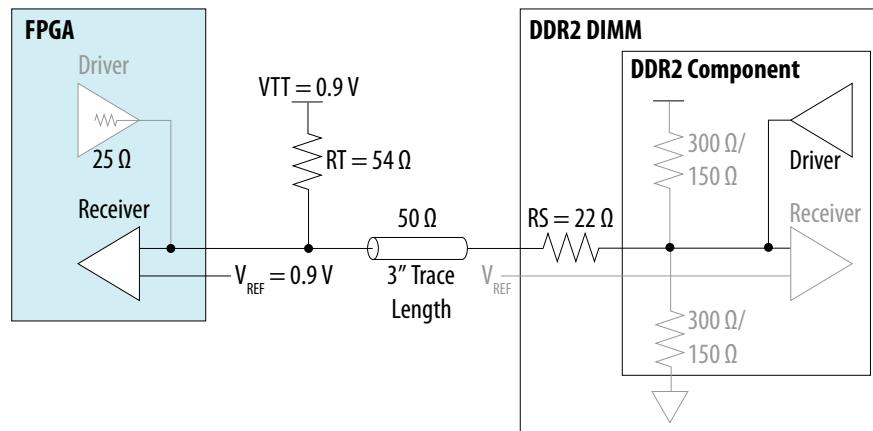
**Figure 3-4: Double Parallel Termination Scheme (Class II) Using ODT on DDR2 SDRAM DIMM with Memory-side Series Resistor**



### Reading from Memory

During read from memory, the ODT feature is turned off, thus there is no difference between using an ODT setting of 150-ohm and 75-ohm. As such, the termination scheme becomes a single parallel termination scheme (Class I) where there is an external resistor on the FPGA side and a series resistor on the memory side, as shown in the following figure.

**Figure 3-5: Single Parallel Termination Scheme (Class I) Using External Resistor and Memory-Side Series Resistor**



## Dual-DIMM Memory Interface with Both Slot 1 and Slot 2 Populated

The following topics focus on a dual-DIMM memory interface where both slot 1 and slot 2 are populated. As such, you can write to either the memory in slot 1 or the memory in slot 2.

### Related Information

#### [ODT Generation Logic](#)

### FPGA Writing to Memory

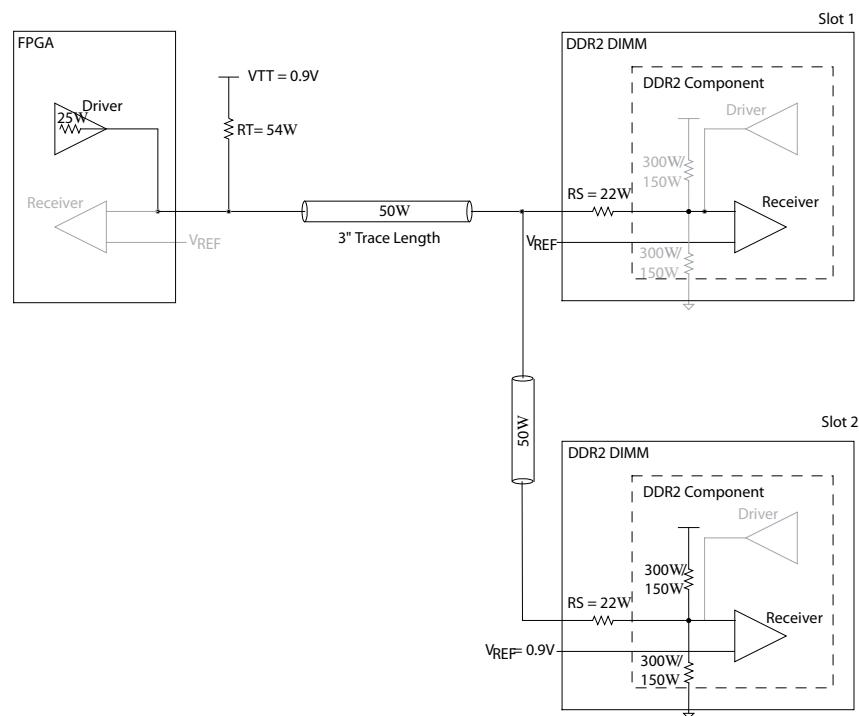
The following topics explore the use of the 150-ohm setting and compares the results to that of the recommended 75-ohm.

In Table 5–1, the recommended ODT setting for a dual DIMM configuration with both slots occupied is 75-ohm. Because there is an option for an ODT setting of 150-ohm, this section explores the usage of the 150-ohm setting and compares the results to that of the recommended 75-ohm.

### Write to Memory in Slot 1 Using an ODT Setting of 75-ohm

The following figure shows the double parallel termination scheme (Class II) using ODT on the memory with the memory-side series resistor when the FPGA is writing to the memory using a 25-ohm OCT drive strength setting on the FPGA. In this scenario, the FPGA is writing to the memory in slot 1 and the ODT feature of the memory at slot 2 is turned on.

**Figure 3-6: Double Parallel Termination Scheme (Class II) Using ODT on DDR2 SDRAM DIMM with a Memory-Side Series Resistor**



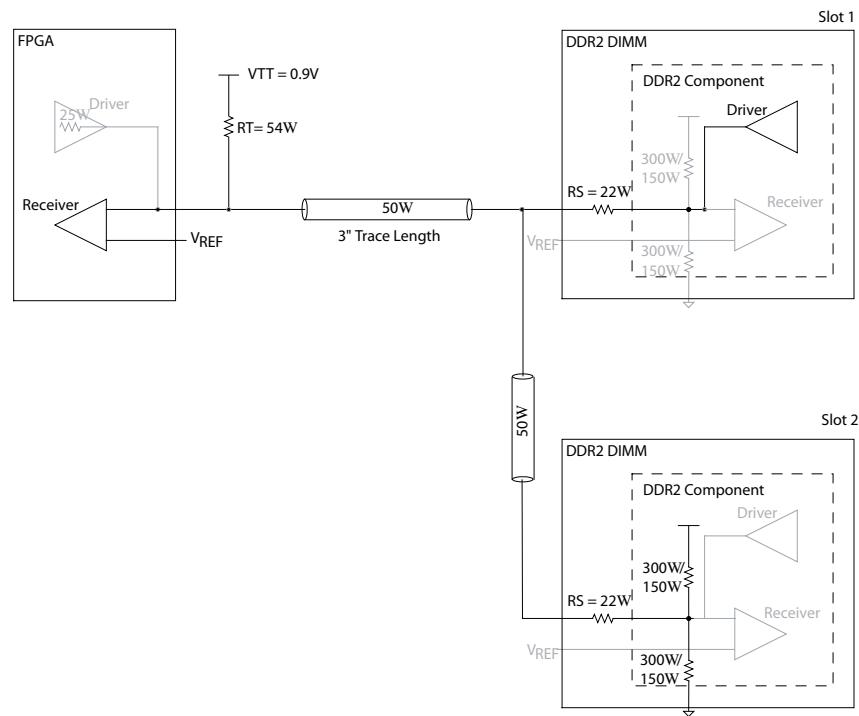
## Reading From Memory

In Table 5–2, the recommended ODT setting for a dual-DIMM configuration with both slots occupied is to turn on the ODT feature using a setting of 75-ohm on the slot that is not read from. As there is an option for an ODT setting of 150-ohm, this section explores the usage of the 150-ohm setting and compares the results to that of the recommended 75-ohm.

### Read From Memory in Slot 1 Using an ODT Setting of 75-ohms on Slot 2

The following figure shows the double parallel termination scheme (Class II) using ODT on the memory with the memory-side series resistor when the FPGA is reading from the memory using a full drive strength setting on the memory. In this scenario, the FPGA is reading from the memory in slot 1 and the ODT feature of the memory at slot 2 is turned on.

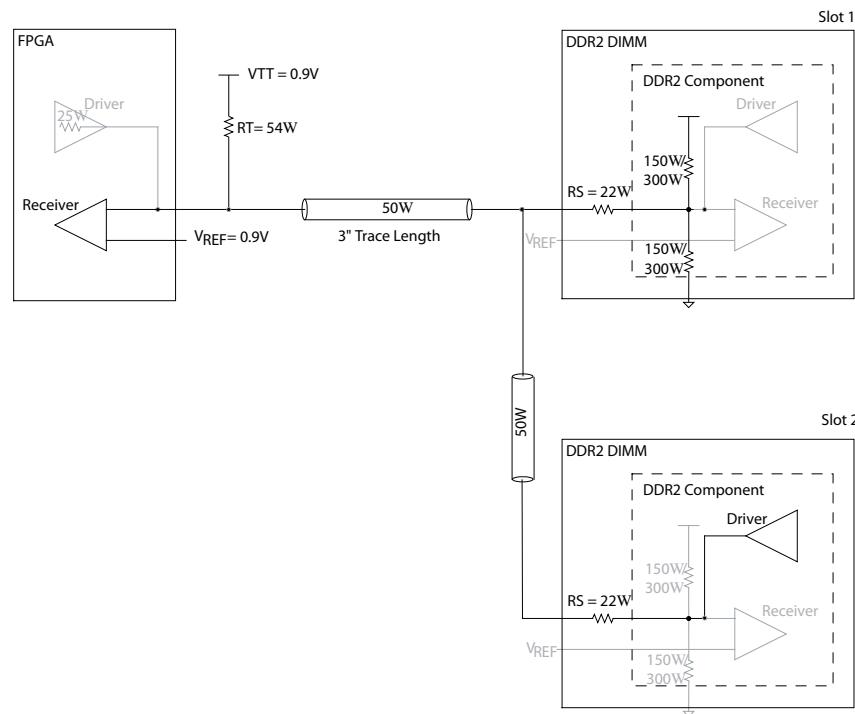
**Figure 3-7: Double Parallel Termination Scheme (Class II) Using External Resistor and Memory-Side Series Resistor and ODT Feature Turned On**



### Read From Memory in Slot 2 Using an ODT Setting of 75-ohms on Slot 1

In this scenario, the FPGA is reading from the memory in slot 2 and the ODT feature of the memory at slot 1 is turned on.

**Figure 3-8: Double Parallel Termination Scheme (Class II) Using External Resistor and a Memory-Side Series Resistor and ODT Feature Turned On**

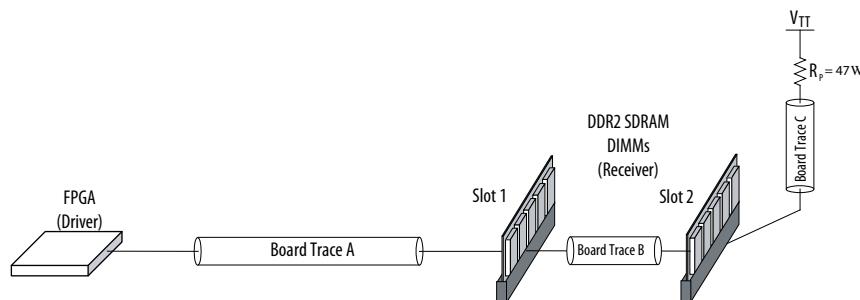


## Dual-DIMM DDR2 Clock, Address, and Command Termination and Topology

The address and command signals on a DDR2 SDRAM interface are unidirectional signals that the FPGA memory controller drives to the DIMM slots. These signals are always Class-I terminated at the memory end of the line, as shown in the following figure.

Always place DDR2 SDRAM address and command Class-I termination after the last DIMM. The interface can have one or two DIMMs, but never more than two DIMMs total.

**Figure 3-9: Multi DIMM DDR2 Address and Command Termination Topology**



In the above figure, observe the following points:

- Board trace A = 1.9 to 4.5 inches (48 to 115 mm)
- Board trace B = 0.425 inches (10.795 mm)
- Board trace C = 0.2 to 0.55 inches (5 to 13 mm)
- Total of board trace A + B + C = 2.5 to 5 inches (63 to 127 mm)
- $R_p = 36$  to 56-ohm
- Length match all address and command signals to +250 mils (+5 mm) or +/- 50 ps of memory clock length at the DIMM.

You may place a compensation capacitor directly before the first DIMM slot 1 to improve signal quality on the address and command signal group. If you fit a capacitor, Altera recommends a value of 24 pF.

For more information, refer to *Micron TN47-01*.

#### Related Information

##### [ODT Generation Logic](#)

## Control Group Signals

The control group of signals: chip select CS#, clock enable CKE, and ODT are always 1T regardless of whether you implement a full-rate or half-rate design.

As the signals are also SDR, the control group signals operate at a maximum frequency of  $0.5 \times$  the data rate. For example, in a 400-MHz design, the maximum control group frequency is 200 MHz.

## Clock Group Signals

Depending on the specific form factor, DDR2 SDRAM DIMMs have two or three differential clock pairs, to ensure that the loading on the clock signals is not excessive. The clock signals are always terminated on the DIMMs and hence no termination is required on your PCB.

Additionally, each DIMM slot is required to have its own dedicated set of clock signals. Hence clock signals are always point-to-point from the FPGA PHY to each individual DIMM slot. Individual memory clock signals should never be shared between two DIMM slots.

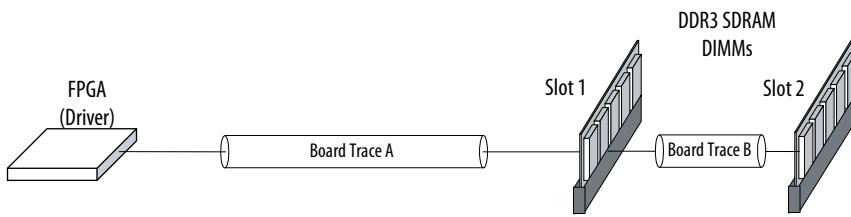
A typical two slot DDR2 DIMM design therefore has six differential memory clock pairs—three to the first DIMM and three to the second DIMM. All six memory clock pairs must be delay matched to each other to  $\pm 25$  mils ( $\pm 0.635$  mm) and  $\pm 10$  mils ( $\pm 0.254$  mm) for each CLK to CLK# signal.

You may place a compensation capacitor between each clock pair directly before the DIMM connector, to improve the clock slew rates. As FPGA devices have fully programmable drive strength and slew rate options, this capacitor is usually not required for FPGA design. However, Altera advise that you simulate your specific implementation to ascertain if this capacitor is required or not. If fitted the best value is typically 5 pF.

## Dual-Slot Unbuffered DDR3 SDRAM

The following topics detail the system implementation of a dual slot unbuffered DDR3 SDRAM interface, operating at up to 400 MHz and 800 Mbps data rates.

The following figure shows a typical DQS, DQ, and DM, and address and command signal topology for a dual-DIMM interface configuration, using the ODT feature of the DDR3 SDRAM components combined with the dynamic OCT features available in Stratix III and Stratix IV devices.

**Figure 3-10: Multi DIMM DDR3 DQS, DQ, and DM, and Address and Command Termination Topology**

In the above figure, observe the following points:

- Board trace A = 1.9 to 4.5 inches (48 to 115 mm)
- Board trace B = 0.425 inches (10.795 mm)
- This topology to both DIMMs is accurate for DQS, DQ, and DM, and address and command signals
- This topology is not correct for CLK and CLK# and control group signals (CS#, CKE, and ODT), which are always point-to-point single rank only.

## Comparison of DDR3 and DDR2 DQ and DQS ODT Features and Topology

DDR3 and DDR2 SDRAM systems are quite similar. The physical topology of the data group of signals may be considered nearly identical.

The FPGA end (driver) I/O standard changes from SSTL18 for DDR2 to SSTL15 for DDR3, but all other OCT settings are identical. DDR3 offers enhanced ODT options for termination and drive-strength settings at the memory end of the line.

For more information, refer to the DDR3 SDRAM ODT matrix for writes and the DDR3 SDRAM ODT matrix for reads tables in the *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

### Related Information

[DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines](#) on page 2-1

## Dual-DIMM DDR3 Clock, Address, and Command Termination and Topology

One significant difference between DDR3 and DDR2 DIMM based interfaces is the address, command and clock signals. DDR3 uses a daisy chained based architecture when using JEDEC standard modules.

The address, command, and clock signals are routed on each module in a daisy chain and feature a fly-by termination on the module. Impedance matching is required to make the dual-DIMM topology work effectively—40 to 50-ohm traces should be targeted on the main board.

### Related Information

[ODT Generation Logic](#)

## Address and Command Signals

Two UDIMMs result in twice the effective load on the address and command signals, which reduces the slew rate and makes it more difficult to meet setup and hold timing ( $t_{IS}$  and  $t_{IH}$ ). However, address and command signals operate at half the interface rate and are SDR. Hence a 400-Mbps data rate equates to an address and command fundamental frequency of 100 MHz.

## Control Group Signals

The control group signals (chip Select CS#, clock enable CKE, and ODT) are only ever single rank. A dual-rank capable DDR3 DIMM slot has two copies of each signal, and a dual-DIMM slot interface has four copies of each signal.

The signal quality of these signals is identical to a single rank case. The control group of signals, are always 1T regardless of whether you implement a full-rate or half-rate design. As the signals are also SDR, the control group signals operate at a maximum frequency of  $0.5 \times$  the data rate. For example, in a 400 MHz design, the maximum control group frequency is 200 MHz.

## Clock Group Signals

Like the control group signals, the clock signals in DDR3 SDRAM are only ever single rank loaded. A dual-rank capable DDR3 DIMM slot has two copies of the signal, and a dual-slot interface has four copies of the `mem_clk` and `mem_clk_n` signals.

For more information about a DDR3 two-DIMM system design, refer to Micron *TN-41-08: DDR3 Design Guide for Two-DIMM Systems*.

## FPGA OCT Features

Many FPGA devices offer OCT. Depending on the chosen device family, series (output), parallel (input) or dynamic (bidirectional) OCT may be supported.

For more information specific to your device family, refer to the respective I/O features chapter in the relevant device handbook.

Use series OCT in place of the near-end series terminator typically used in both Class I or Class II termination schemes that both DDR2 and DDR3 type interfaces use.

Use parallel OCT in place of the far-end parallel termination typically used in Class I termination schemes on unidirectional input only interfaces. For example, QDR-II type interfaces, when the FPGA is at the far end.

Use dynamic OCT in place of both the series and parallel termination at the FPGA end of the line. Typically use dynamic OCT for DQ and DQS signals in both DDR2 and DDR3 type interfaces. As the parallel termination is dynamically disabled during writes, the FPGA driver only ever drives into a Class I transmission line. When combined with dynamic ODT at the memory, a truly dynamic Class I termination scheme exists where both reads and writes are always fully Class I terminated in each direction. Hence, you can use a fully dynamic bidirectional Class I termination scheme instead of a static discretely terminated Class II topology, which saves power, printed circuit board (PCB) real estate, and component cost.

## Arria V, Cyclone V, Stratix III, Stratix IV, and Stratix V Devices

Arria® V, Cyclone® V, Stratix III, Stratix IV, and Stratix V devices feature full dynamic OCT termination capability, Altera advise that you use this feature combined with the SDRAM ODT to simplify PCB layout and save power.

## Arria II GX Devices

Arria II GX devices do not support dynamic OCT. Altera recommends that you use series OCT with SDRAM ODT. Use parallel discrete termination at the FPGA end of the line when necessary.

For more information, refer to the *DDR2 and DDR3 SDRAM Board Design Guidelines* chapter.

**Related Information****DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines** on page 2-1

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Removed <i>Address and Command Signals</i> section from <i>Dual-DIMM DDR2 Clock, Address, and Command Termination and Topology</i>
December 2013	2013.12.16	<ul style="list-style-type: none"><li>• Reorganized content.</li><li>• Consolidated General Layout Guidelines.</li><li>• Removed references to ALTMEMPHY.</li><li>• Removed references to Stratix II devices.</li></ul>
June 2012	4.1	Added Feedback icon.
November 2011	4.0	Added Arria V and Cyclone V information.
June 2011	3.0	Added Stratix V information.
December 2010	2.1	Maintenance update.
July 2010	2.0	Updated Arria II GX information.
April 2010	1.0	Initial release.

# LPDDR2 and LPDDR3 SDRAM Board Design Guidelines

4

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The following topics provide guidelines to improve your system's signal integrity and to successfully implement an LPDDR2 or LPDDR3 SDRAM interface in your system.

## LPDDR2 Guidance

The LPDDR2 SDRAM Controller with UniPHY intellectual property (IP) enables you to implement LPDDR2 SDRAM interfaces with Arria® V and Cyclone® V devices.

The following topics focus on key factors that affect signal integrity:

- I/O standards
- LPDDR2 configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

### I/O Standards

LPDDR2 SDRAM interface signals use HSUL-12 JEDEC I/O signaling standards, which provide low power and low emissions. The HSUL-12 JEDEC I/O standard is mainly for point-to-point unterminated bus topology. This standard eliminates the need for external series or parallel termination resistors in LPDDR2 SDRAM implementation. With this standard, termination power is greatly reduced and programmable drive strength is used to match the impedance.

To select the most appropriate standard for your interface, refer to the the *Device Datasheet for Arria V Devices* chapter in the *Arria V Device Handbook*, or the *Device Datasheet for Cyclone V Devices* chapter in the *Cyclone V Device Handbook*.

### Related Information

- [Arria V Device Datasheet](#)
- [Cyclone V Device Datasheet](#)

## LPDDR2 SDRAM Configurations

The LPDDR2 SDRAM Controller with UniPHY IP supports interfaces for LPDDR2 SDRAM with a single device, and multiple devices up to a maximum width of 32 bits.

When using multiple devices, a balanced-T topology is recommended for the signal connected from single point to multiple point, to maintain equal flight time.

---

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

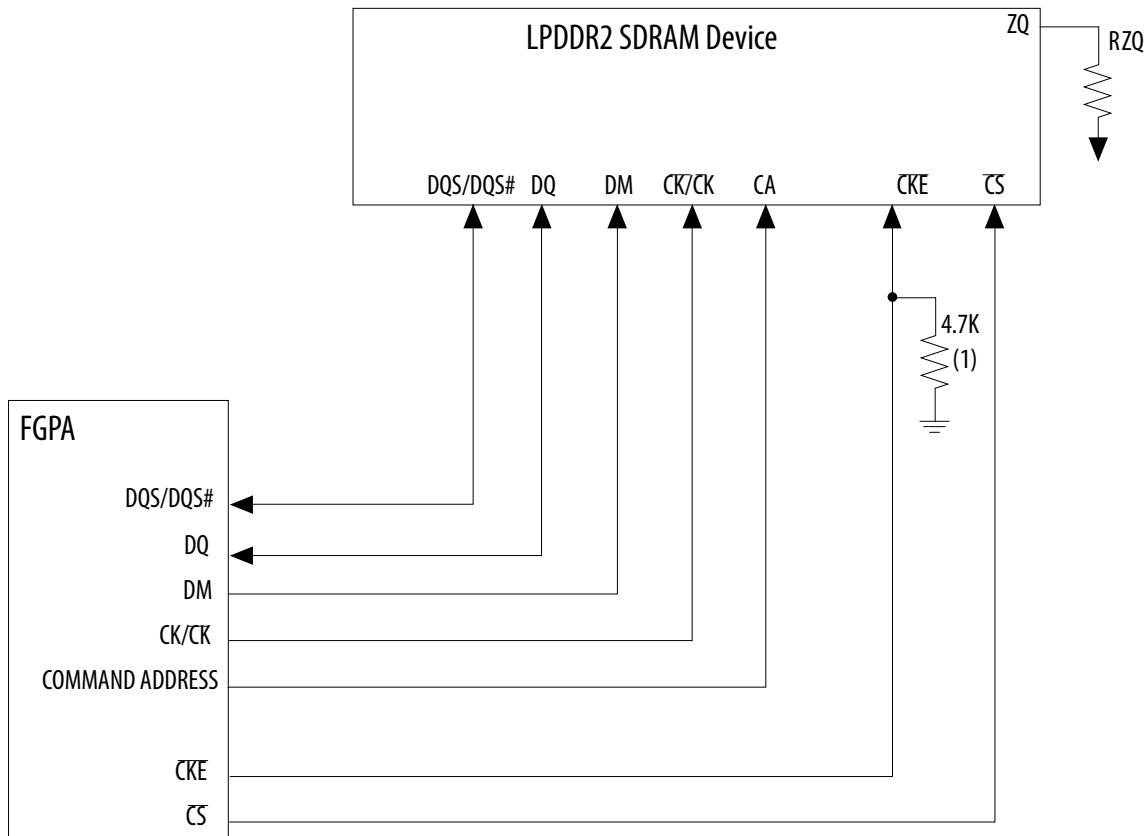
ISO  
9001:2008  
Registered

You should connect a 200 ohm differential termination resistor between CK/CK# in multiple device designs as shown in the second figure below, to maintain an effective resistance of 100 ohms.

You should also simulate your multiple device design to obtain the optimum drive strength settings and ensure correct operation.

The following figure shows the main signal connections between the FPGA and a single LPDDR2 SDRAM component.

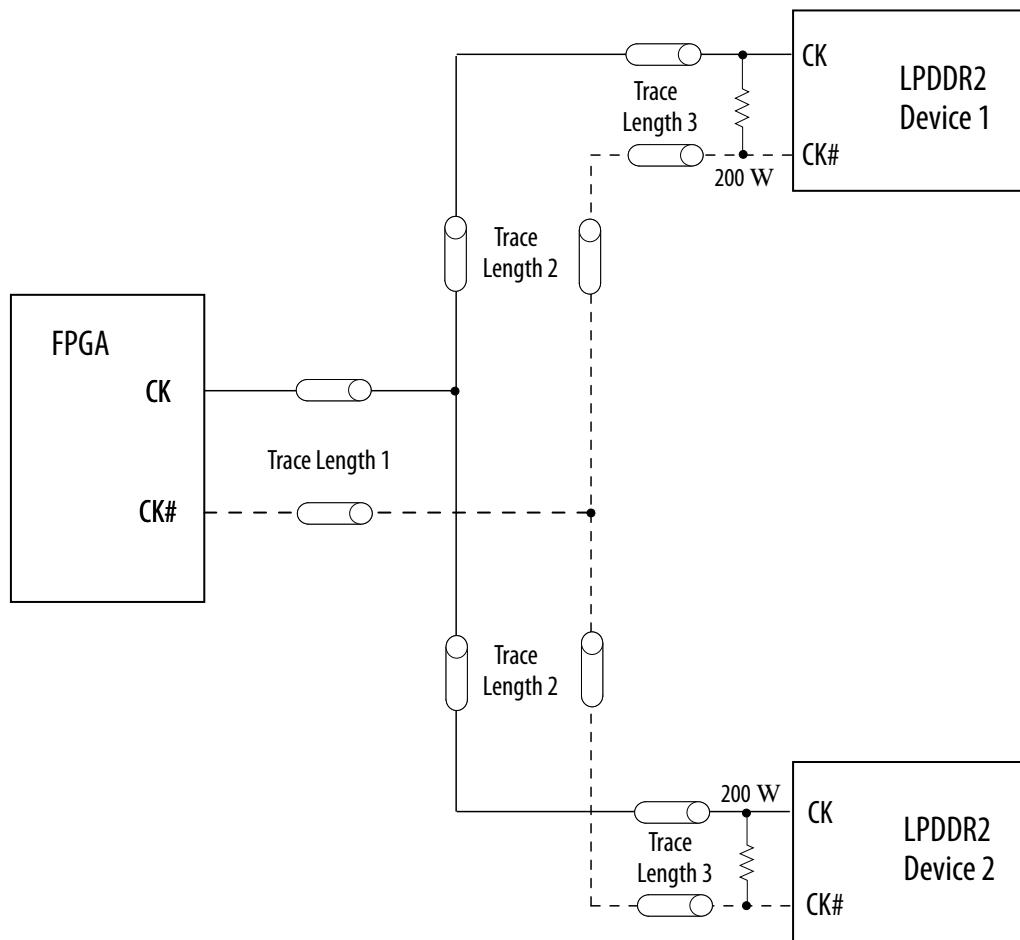
**Figure 4-1: Configuration with a Single LPDDR2 SDRAM Component**



Note to Figure:

1. Use external discrete termination, as shown for CKE, but you may require a pull-down resistor to GND. Refer to the LPDDR2 SDRAM device data sheet for more information about LPDDR2 SDRAM power-up sequencing.

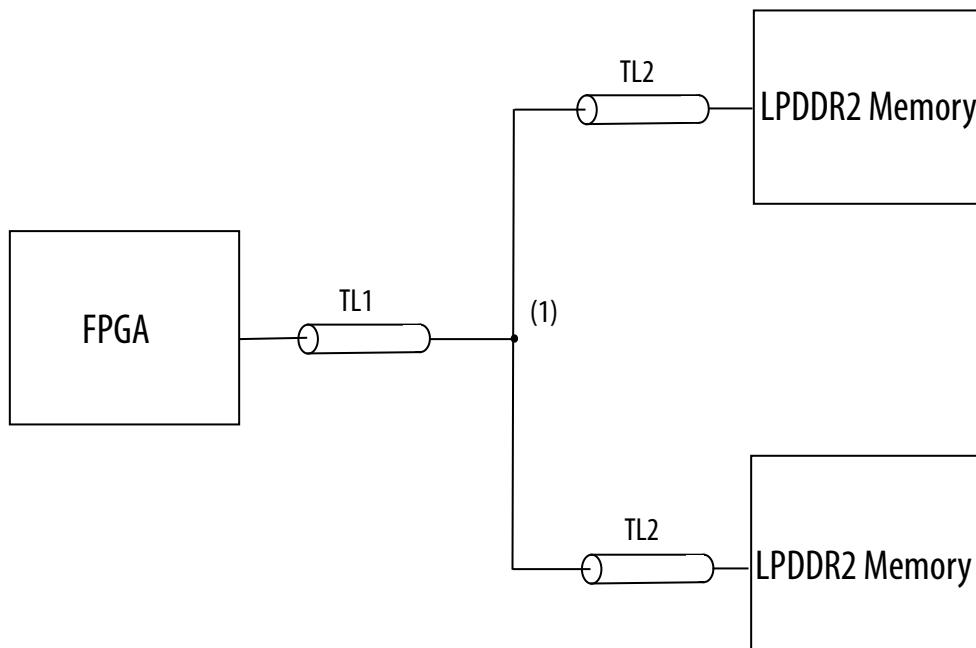
The following figure shows the differential resistor placement for CK/CK# for multi-point designs.

**Figure 4-2: CK Differential Resistor Placement for Multi Point Design**

Note to Figure:

1. Place 200-ohm differential resistors near the memory devices at the end of the last board trace segments.

The following figure shows the detailed balanced topology recommended for the address and command signals in the multi-point design.

**Figure 4-3: Address Command Balanced-T Topology**

Notes to Figure:

1. Split the trace close to the memory devices to minimize signal reflections and impedance nonuniformity.
2. Keep the TL2 traces as short as possible, so that the memory devices appear as a single load.

## OCT Signal Terminations for Arria V and Cyclone V Devices

Arria V and Cyclone V devices offer OCT technology. The following table lists the extent of OCT support for each device.

**Table 4-1: On-Chip Termination Schemes**

Termination Scheme	I/O Standard	Arria V and Cyclone V
On-Chip Series Termination without Calibration	HSUL-12	34/40/48/60/80
On-Chip Series Termination with Calibration	HSUL-12	34/40/48/60/80

On-chip series ( $R_s$ ) termination supports output buffers, and bidirectional buffers only when they are driving output signals. LPDDR2 SDRAM interfaces have bidirectional data paths. The UniPHY IP uses series OCT for memory writes but no parallel OCT for memory reads because Arria V and Cyclone V support only on-chip series termination in the HSUL-12 I/O standard.

For Arria V and Cyclone V devices, the HSUL-12 I/O calibrated terminations are calibrated against 240 ohm 1% resistors connected to the  $R_{ZQ}$  pins in an I/O bank with the same  $V_{CCIO}$  as the LPDDR2 interface.

Calibration occurs at the end of the device configuration.

LPDDR2 SDRAM memory components have a ZQ pin which connects through a resistor  $R_{ZQ}$  (240 ohm) to ground. The output signal impedances for LPDDR2 SDRAM are 34.3 ohm, 40 ohm, 48 ohm, 60 ohm, 80 ohm, and 120 ohm. The output signal impedance is set by mode register during initialization. Refer to the LPDDR2 SDRAM device data sheet for more information.

For information about OCT, refer to the *I/O Features in Arria V Devices* chapter in the *Arria V Device Handbook*, or the *I/O Features in Cyclone V Devices* chapter in the *Cyclone V Device Handbook*.

The following section shows HyperLynx simulation eye diagrams to demonstrate signal termination options. Altera strongly recommends signal terminations to optimize signal integrity and timing margins, and to minimize unwanted emissions, reflections, and crosstalk.

All of the eye diagrams shown in this section are for a 50 ohm trace with a propagation delay of 509 ps which is approximately a 2.8-inch trace on a standard FR4 PCB. The signal I/O standard is HSUL-12.

The eye diagrams in this section show the best case achievable and do not take into account PCB vias, crosstalk, and other degrading effects such as variations in the PCB structure due to manufacturing tolerances.

**Note:** Simulate your design to ensure correct operation.

#### Related Information

- [I/O Features in Arria V Devices](#)
- [I/O Features in Cyclone V Devices](#)

## Outputs from the FPGA to the LPDDR2 Component

The following output signals are from the FPGA to the LPDDR2 SDRAM component:

- write data (DQ)
- data mask (DM)
- data strobe (DQS/DQS#)
- command address
- command (CS, and CKE)
- clocks (CK/CK#)

No far-end memory termination is needed when driving output signals from FPGA to LPDDR2 SDRAM. Cyclone V and Arria V devices offer the OCT series termination for impedance matching.

## Input to the FPGA from the LPDDR2 SDRAM Component

The LPDDR2 SDRAM component drives the following input signals into the FPGA:

- read data
- DQS

LPDDR2 SDRAM provides the flexibility to adjust drive strength to match the impedance of the memory bus, eliminating the need for termination voltage (VTT) and series termination resistors.

The programmable drive strength options are 34.3 ohms, 40 ohms (default), 48 ohms, 60 ohms, 80 ohms, and 120 ohms. You must perform board simulation to determine the best option for your board layout.

**Note:** By default, Altera LPDDR2 SDRAM UniPHY IP uses 40 ohm drive strength.

## Termination Schemes

The following table lists the recommended termination schemes for major LPDDR2 SDRAM memory interface signals.

These signals include data ( $D_Q$ ), data strobe ( $DQS$ ), data mask ( $DM$ ), clocks ( $CK$ , and  $CK\#$ ), command address ( $CA$ ), and control ( $CS\#$ , and  $CKE$ ).

**Table 4-2: Termination Recommendations for Arria V and Cyclone V Devices**

Signal Type	HSUL-12 Standard <sup>(1) (2)</sup>	Memory End Termination
DQS/DQS#	R34 CAL	ZQ40
Data (Write)	R34 CAL	-
Data (Read)	-	ZQ40
Data Mask ( $DM$ )	R34 CAL	-
CK/CK# Clocks	R34 CAL	$\times 1 = -$ <sup>(4)</sup> $\times 2 = 200$ -ohm Differential <sup>(5)</sup>
Command Address ( $CA$ ),	R34 CAL	-
Chip Select ( $CS\#$ )	R34 CAL	-
Clock Enable ( $CKE$ ) <sup>(3)</sup>	R34 CAL	4.7 K-ohm parallel to GND

Notes to Table:

1. R is effective series output impedance.
2. CAL is OCT with calibration.
3. Altera recommends that you use a 4.7 K-ohm parallel to GND if your design meets the power sequencing requirements of the LPDDR2 SDRAM component. Refer to the LPDDR2 SDRAM data sheet for further information.
4.  $\times 1$  is a single-device load.
5.  $\times 2$  is a double-device load. An alternative option is to use a 100 -ohm differential termination at the trace split.

**Note:** The recommended termination schemes in the above table are based on 2.8 inch maximum trace length analysis. You may add the external termination resistor or adjust the drive strength to improve signal integrity for longer trace lengths. Recommendations for external termination are as follows:

- Class I termination (50 ohms parallel to VTT at the memory end) — Unidirectional signal (Command Address, control, and CK/CK# signals)
- Class II termination (50 ohms parallel to VTT at both ends) — Bidirectional signal (  $D_Q$  and  $DQS/DQS\#$  signal)

Altera recommends that you simulate your design to ensure good signal integrity.

## General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the slew rate and propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

**Table 4-3: General Layout Guidelines**

Parameter	Guidelines
Impedance	<ul style="list-style-type: none"><li>• All unused via pads must be removed, because they cause unwanted capacitance.</li><li>• Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.</li></ul>
Decoupling Parameter	<ul style="list-style-type: none"><li>• Use 0.1 uF in 0402 size to minimize inductance</li><li>• Make VTT voltage decoupling close to pull-up resistors</li><li>• Connect decoupling caps between VTT and ground</li><li>• Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin</li><li>• Verify the capacitive decoupling using the Altera Power Distribution Network Design Tool</li></ul>
Power	<ul style="list-style-type: none"><li>• Route GND and V<sub>CC</sub> as planes</li><li>• Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation</li><li>• Route VTT as islands or 250-mil (6.35-mm) power traces</li><li>• Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces</li></ul>

Parameter	Guidelines
General Routing	<p>All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Altera recommend that signals from the same net group always be routed on the same layer.</p> <ul style="list-style-type: none"> <li>• Use 45° angles (<i>not</i> 90° corners)</li> <li>• Avoid T-Junctions for critical nets or clocks</li> <li>• Avoid T-junctions greater than 250 mils (6.35 mm)</li> <li>• Disallow signals across split planes</li> <li>• Restrict routing other signals close to system reset signals</li> <li>• Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks</li> </ul>

**Related Information**[Power Distribution Network Design Tool](#)

## LPDDR2 Layout Guidelines

The following table lists the LPDDR2 SDRAM general routing layout guidelines.

**Note:** The following layout guidelines include several +/- length-based rules. These length-based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristics of your PCB implementation. They do not include any margin for crosstalk. Altera recommends that you get accurate time base skew numbers when you simulate your specific implementation.

**Table 4-4: LPDD2 Layout Guidelines**

Parameter	Guidelines
General Routing	<ul style="list-style-type: none"> <li>• If you must route signals of the same net group on different layers with the same impedance characteristic, simulate your worst case PCB trace tolerances to ascertain actual propagation delay differences. Typical layer to layer trace delay variations are of 15 ps/inch order.</li> <li>• Avoid T-junctions greater than 75 ps (approximately 25 mils, 6.35 mm).</li> <li>• Match all signals within a given DQ group with a maximum skew of <math>\pm 10</math> ps and route on the same layer.</li> </ul>
Clock Routing	<ul style="list-style-type: none"> <li>• Route clocks on inner layers with outer-layer run lengths held to under 150 ps.</li> <li>• These signals should maintain a 10-mil (0.254 mm) spacing from other nets.</li> <li>• Clocks should maintain a length-matching between clock pairs of <math>\pm 5</math> ps.</li> <li>• Differential clocks should maintain a length-matching between P and N signals of <math>\pm 2</math> ps.</li> <li>• Space between different clock pairs should be at least three times the space between the traces of a differential pair.</li> </ul>

Parameter	Guidelines
Address and Command Routing	<ul style="list-style-type: none"><li>To minimize crosstalk, route address, and command signals on a different layer than the data and data mask signals.</li><li>Do not route the differential clock (CK/CK#) and clock enable (CKE) signals close to the address signals.</li></ul>
External Memory Routing Rules	<ul style="list-style-type: none"><li>Apply the following parallelism rules for the LPDDR2 SDRAM data groups:<ul style="list-style-type: none"><li>4 mils for parallel runs &lt; 0.1 inch (approximately 1× spacing relative to plane distance).</li><li>5 mils for parallel runs &lt; 0.5 inch (approximately 1× spacing relative to plane distance).</li><li>10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance).</li><li>15 mils for parallel runs between 1.0 and 2.8 inch (approximately 3× spacing relative to plane distance).</li></ul></li><li>Apply the following parallelism rules for the address/command group and clocks group:<ul style="list-style-type: none"><li>4 mils for parallel runs &lt; 0.1 inch (approximately 1× spacing relative to plane distance)</li><li>10 mils for parallel runs &lt; 0.5 inch (approximately 2× spacing relative to plane distance)</li><li>15 mils for parallel runs between 0.5 and 1.0 inches (approximately 3× spacing relative to plane distance)</li><li>20 mils for parallel runs between 1.0 and 2.8 inches (approximately 4× spacing relative to plane distance)</li></ul></li></ul>
Maximum Trace Length	<ul style="list-style-type: none"><li>Keep traces as short as possible. The maximum trace length of all signals from the FPGA to the LPDDR2 SDRAM components should be less than 509 ps. Altera recommends that you simulate your design to ensure good signal integrity.</li></ul>

#### Related Information

[Altera Power Distribution Network \(PDN\) Design tool](#)

## LPDDR2 SDRAM Layout Approach

Altera recommends the following layout approach, based on the layout guidelines in the above table:

1. Route the differential clocks (`CK/CK#`) and data strobe (`DQS/DQS#`) with a length-matching between P and N signals of  $\pm 2$  ps.
2. Route the `DQS/DQS#` associated with a DQ group on the same PCB layer. Match these DQS pairs to within  $\pm 5$  ps.
3. Set the `DQS/DQS#` as the target trace propagation delay for the associated data and data mask signals.
4. Route the data and data mask signals for the DQ group ideally on the same layer as the associated `DQS/DQS#` to within  $\pm 10$  ps skew of the target `DQS/DQS#`.
5. Route the `CK/CK#` clocks and set as the target trace propagation delays for the DQ group. Match the `CK/CK#` clock to within  $\pm 50$  ps of all the `DQS/DQS#`.
6. Route the address/control signal group (address, cs, CKE) ideally on the same layer as the `CK/CK#` clocks, to within  $\pm 20$  ps skew of the `CK/CK#` traces.

This layout approach provides a good starting point for a design requirement of the highest clock frequency supported for the LPDDR2 SDRAM interface.

**Note:** Altera recommends that you create your project in the Quartus® Prime software with a fully implemented LPDDR2 SDRAM Controller with UniPHY interface, and observe the interface timing margins to determine the actual margins for your design.

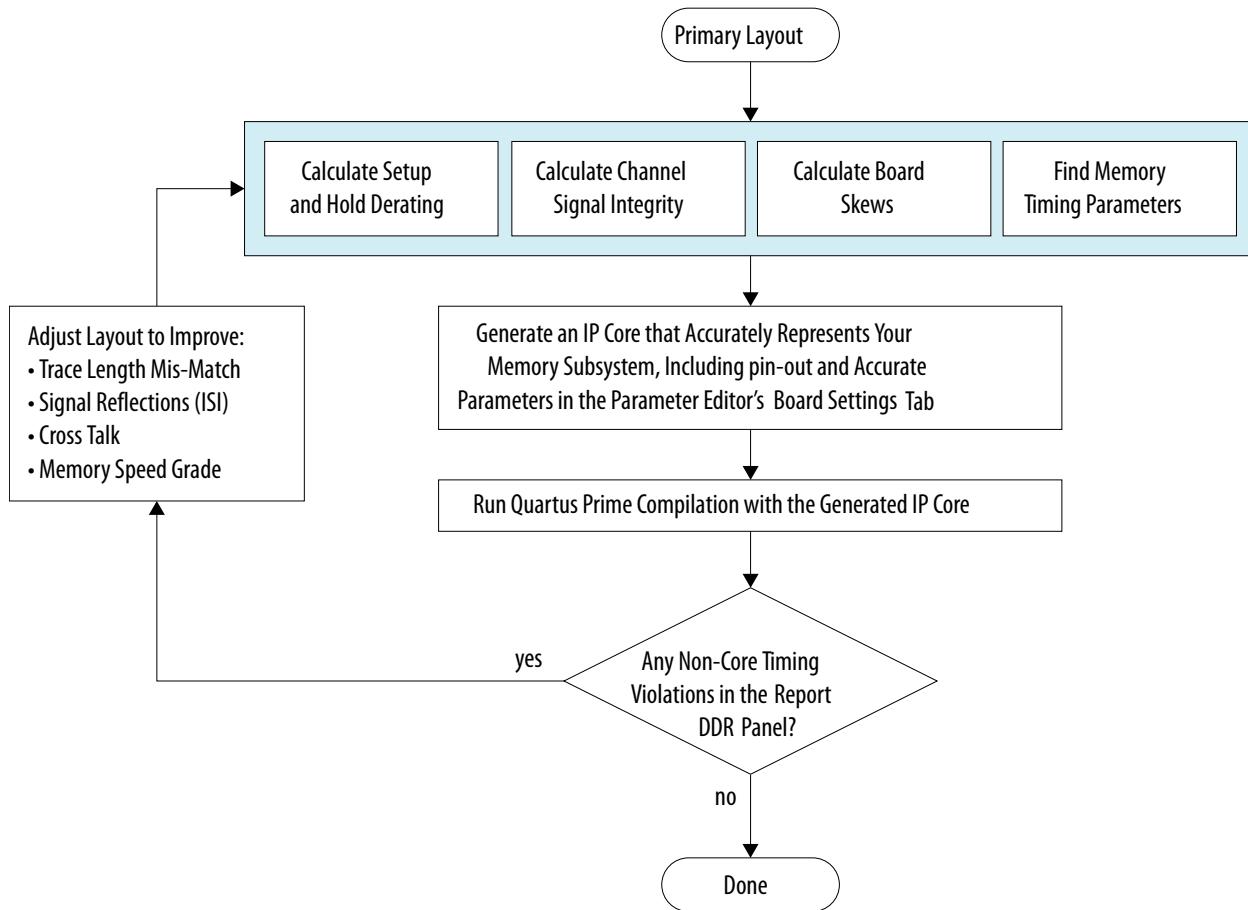
Although the recommendations in this chapter are based on simulations, you can apply the same general principles when determining the best termination scheme, drive strength setting, and loading style to any board designs. Even armed with this knowledge, it is still critical that you perform simulations, either using IBIS or HSPICE models, to determine the quality of signal integrity on your designs.

## LPDDR3 Guidance

The LPDDR3 SDRAM Controller intellectual property (IP) enables you to implement LPDDR3 SDRAM interfaces with Arria® 10 devices.

For all practical purposes, you can regard the TimeQuest timing analyzer's report on your memory interface as definitive for a given set of memory and board timing parameters. You can find timing information under **Report DDR** in TimeQuest and on the **Timing Analysis** tab in the parameter editor.

The following flowchart illustrates the recommended process to follow during the design phase, to determine timing margin and make iterative improvements to your design.



## Signal Integrity, Board Skew, and Board Setting Parameters

### Channel Signal Integrity

For information on determining channel signal integrity, refer to the Altera wiki page: [http://www.alterawiki.com/wiki/Arria\\_10\\_EMIF\\_Simulation\\_Guidance](http://www.alterawiki.com/wiki/Arria_10_EMIF_Simulation_Guidance).

### Board Skew

For information on calculating board skew parameters, refer to *Implementing and Parameterizing Memory IP*. The Board Skew Parameter Tool is an interactive tool that can help you calculate board skew parameters if you know the absolute delay values for all the memory related traces.

### Arria 10 Board Setting Parameters

For Board Setting and layout approach information for Arria 10 devices, refer to the Altera wiki page: [http://www.alterawiki.com/wiki/Arria\\_10\\_EMIF\\_Simulation\\_Guidance](http://www.alterawiki.com/wiki/Arria_10_EMIF_Simulation_Guidance).

### LPDDR3 Layout Guidelines

The following table lists the LPDDR3 SDRAM general routing layout guidelines.

**Table 4-5: LPDDR3 Layout Guidelines**

Parameter	Guidelines
Max Length Discrete	500 ps.
Data Group Skew	Match DM and DQ within 5 ps of DQS.
Address/Command vs Clock Skew	Match Address/Command signals within 10 ps of mem CK.
Package Skew Matching	Yes.
Clock matching	<ul style="list-style-type: none"> <li>• 2 ps within a clock pair</li> <li>• 5 ps between clock pairs</li> </ul>
Spacing Guideline Data/ Data Strobe/Address/ Command	3H spacing between any Data and Address/Command traces, where H is distance to the nearest return path.
Spacing Guideline Mem Clock	5H spacing between mem clock and any other signal, where H is distance to the nearest return path.

## Package Deskew

Trace lengths inside the device package are not uniform for all package pins. The nonuniformity of package traces can affect system timing for high frequencies. In the Quartus II software version 12.0 and later, and the Quartus Prime software, a package deskew option is available.

If you do not enable the package deskew option, the Quartus Prime software uses the package delay numbers to adjust skews on the appropriate signals; you do not need to adjust for package delays on the board traces. If you do enable the package deskew option, the Quartus Prime software does not use the package delay numbers for timing analysis, and you must deskew the package delays with the board traces for the appropriate signals for your design.

## DQ/DQS/DM Deskew

To get the package delay information, follow these steps:

1. Select the **FPGA DQ/DQS Package Skews Deskewed on Board** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Run **Analysis and Synthesis** in the Quartus Prime software. (Skip this step if you are using an Arria 10 device.)
5. Run the `<core_name>.p0_pin_assignment.tcl` script. (Skip this step if you are using an Arria 10 device.)
6. Compile your design.
7. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

## Address and Command Deskew

Deskew address and command delays as follows:

1. Select the **FPGA Address/Command Package Skews Deskewed on Board** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Run **Analysis and Synthesis** in the Quartus Prime software. (Skip this step if you are using an Arria 10 device.)
5. Run the `<core_name>.p0_pin_assignment.tcl` script. (Skip this step if you are using an Arria 10 device.)
6. Compile your design.
7. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

## Package Deskew Recommendations for Arria 10 Devices

The following table shows package deskew recommendations for all protocols supported on Arria 10 devices.

As operating frequencies increase, it becomes increasingly critical to perform package deskew. The frequencies listed in the table are the *minimum* frequencies for which you must perform package deskew.

If you plan to use a listed protocol at the specified frequency or higher, you must perform package deskew. For example, you must perform package deskew if you plan to use dual-rank DDR4 at 800 MHz or above.

Protocol	Minimum Frequency (MHz) for Which to Perform Package Deskew		
	Single Rank	Dual Rank	Quad Rank
DDR4	933	800	667
DDR3	933	800	667
LPDDR3	667	533	Not required
QDR IV	933	Not applicable	Not applicable
RLDRAM 3	933	667	Not applicable
RLDRAM II	Not required	Not applicable	Not applicable
QDR II, II+, II+ Xtreme	Not required	Not applicable	Not applicable

**Note:** QDR IV SRAM for Arria 10 EMIF IP will be supported in a future release of the Quartus Prime software.

The recommendations in the above table are based on preliminary timing models, and may be updated in the future. If you are designing a board with Arria 10 devices and require exact package trace delays, contact Altera Support.

## Deskew Example

Consider an example where you want to deskew an interface with 4 DQ pins, 1 DQS pin, and 1 DQSn pin.

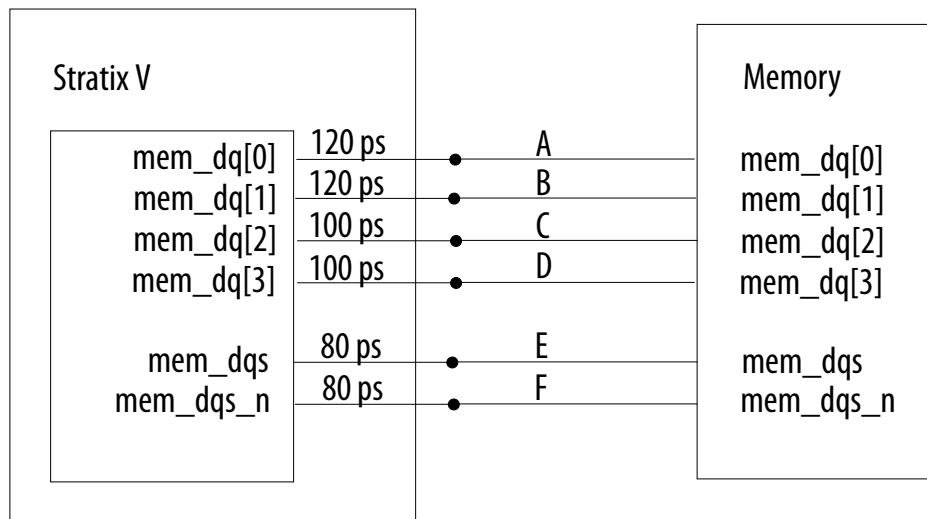
Let's assume an operating frequency of 667 MHz, and the package lengths for the pins reported in the `.pin` file as follows:

```
dq[0] = 120 ps
dq[1] = 120 ps
dq[2] = 100 ps
dq[3] = 100 ps
```

```
dqs      = 80 ps
dqs_n   = 80 ps
```

The following figure illustrates this example.

**Figure 4-4: Deskew Example**

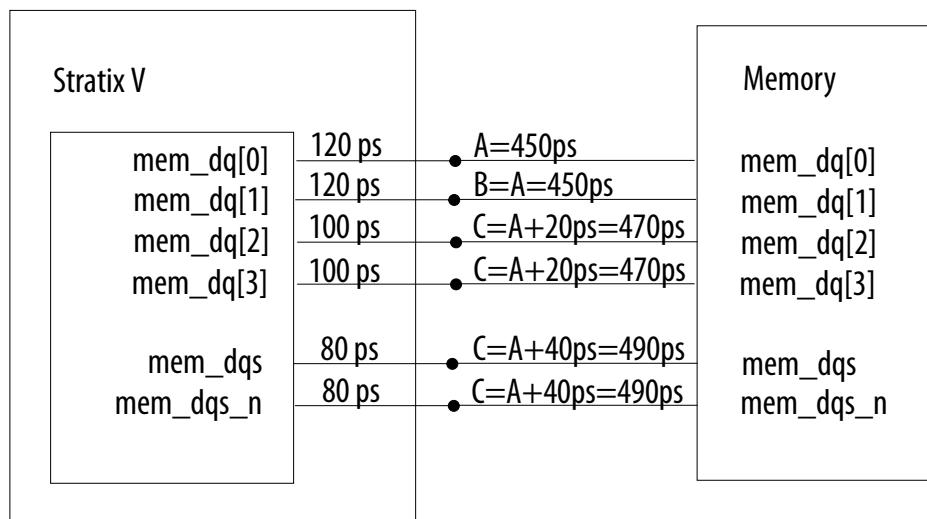


When you perform length matching for all the traces in the DQS group, you must take package delays into consideration. Because the package delays of traces A and B are 40 ps longer than the package delays of traces E and F, you would need to make the board traces for E and F 40 ps longer than the board traces for A and B.

A similar methodology would apply to traces C and D, which should be 20 ps longer than the lengths of traces A and B.

The following figure shows this scenario with the length of trace A at 450 ps.

**Figure 4-5: Deskew Example with Trace Delay Calculations**



When you enter the board skew into the Board Settings tab of the DDR3 parameter editor, you should calculate the board skew parameters as the sums of board delay and corresponding package delay. If a pin does not have a package delay (such as address and command pins), you should use the board delay only.

The example of the preceding figure shows an ideal case where board skews are perfectly matched. In reality, you should allow plus or minus 10 ps of skew mismatch within a DQS group (DQ/DQS/DM).

## Package Migration

Package delays can be different for the same pin in different packages. If you want to use multiple migratable packages in your system, you should compensate for package skew as described in this topic.

Assume two migratable packages, device A and device B, and that you want to compensate for the board trace lengths for device A. Follow these steps:

1. Compile your design for device A, with the Package Skew option enabled.
2. Note the skews in the `<core_name>.pin` file for device A. Deskew these package skews with board trace lengths as described in the preceding examples.
3. Recompile your design for device A.
4. For Device B open the parameter editor and deselect Package Deskew option.
5. Calculate board skew parameters only taking into account the board traces for Device B and enter that value into the parameter editor for Device B.
6. Regenerate the IP and recompile the design for Device B.
7. Verify that timing requirements are met for both device A and device B.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"><li>• Changed recommended value of skew mismatch in <i>Deskew Example</i> topic.</li></ul>
November 2015	2015.11.02	<ul style="list-style-type: none"><li>• Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li><li>• Added content for LPDDR3.</li></ul>
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	<ul style="list-style-type: none"><li>• Removed millimeter approximations from lengths expressed in picoseconds in <i>LPDDR2 Layout Guidelines</i> table.</li><li>• Minor formatting fixes in <i>LPDDR2 Layout Guidelines</i> table.</li></ul>
December 2013	2013.12.16	Consolidated General Layout Guidelines.
November 2012	1.0	Initial release.

# RLDRAM II and RLDRAM 3 Board Design Guidelines

5

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The following topics provide layout guidelines for you to improve your system's signal integrity and to successfully implement an RLDRAM II or RLDRAM 3 interface.

The RLDRAM II Controller with UniPHY intellectual property (IP) enables you to implement common I/O (CIO) RLDRAM II interfaces with Arria® V, Stratix® III, Stratix IV, and Stratix V devices. The RLDRAM 3 UniPHY IP enables you to implement CIO RLDRAM 3 interfaces with Stratix V and Arria V GZ devices. You can implement separate I/O (SIO) RLDRAM II or RLDRAM 3 interfaces with the ALTDQ\_DQS or ALTDQ\_DQS2 IP cores.

The following topics focus on the following key factors that affect signal integrity:

- I/O standards
- RLDRAM II and RLDRAM 3 configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

## I/O Standards

RLDRAM II interface signals use one of the following JEDEC I/O signalling standards:

- HSTL-15—provides the advantages of lower power and lower emissions.
- HSTL-18—provides increased noise immunity with slightly greater output voltage swings.

RLDRAM 3 interface signals use the following JEDEC I/O signalling standards: HSTL 1.2 V and SSTL-12.

To select the most appropriate standard for your interface, refer to the following:

- *Device Data Sheet for Arria II Devices* chapter in the *Arria II Device Handbook*
- *Device Data Sheet for Arria V Devices* chapter in the *Arria V Device Handbook*
- *Stratix III Device Data Sheet: DC and Switching Characteristics* chapter in the *Stratix III Device Handbook*
- *DC and Switching Characteristics for Stratix IV Devices* chapter in the *Stratix IV Device Handbook*
- *DC and Switching Characteristics for Stratix V Devices* chapter in the *Stratix V Device Handbook*

The RLDRAM II Controller with UniPHY IP defaults to HSTL 1.8 V Class I outputs and HSTL 1.8 V inputs. The RLDRAM 3 UniPHY IP defaults to HSTL 1.2 V Class I outputs and HSTL 1.2 V inputs.

**Note:** The default for RLDRAM 3 changes from Class I to Class II, supporting up to 933 MHz, with the release of the Quartus II software version 12.1 SP1.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

### Related Information

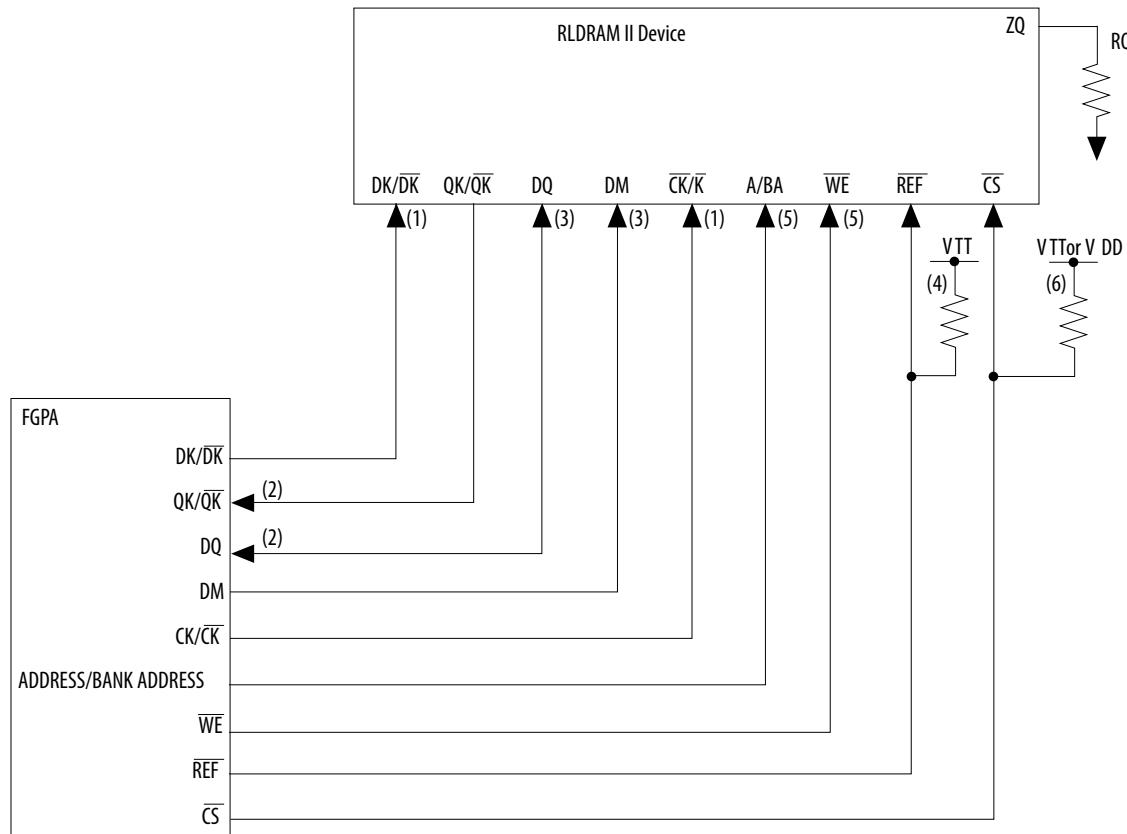
- [Device Data Sheet for Arria II Devices](#)
- [Device Data Sheet for Arria V Devices](#)
- [Stratix III Device Data Sheet: DC and Switching Characteristics](#)
- [DC and Switching Characteristics for Stratix IV Devices](#)
- [DC and Switching Characteristics for Stratix V Devices](#)

## RLDRAM II Configurations

The RLDRAM II Controller with UniPHY IP supports CIO RLDRAM II interfaces with one or two devices. With two devices, the interface supports a width expansion configuration up to 72-bits. The termination and layout principles for SIO RLDRAM II interfaces are similar to CIO RLDRAM II, except that SIO RLDRAM II interfaces have unidirectional data buses.

The following figure shows the main signal connections between the FPGA and a single CIO RLDRAM II component.

**Figure 5-1: Configuration with a Single CIO RLDRAM II Component**

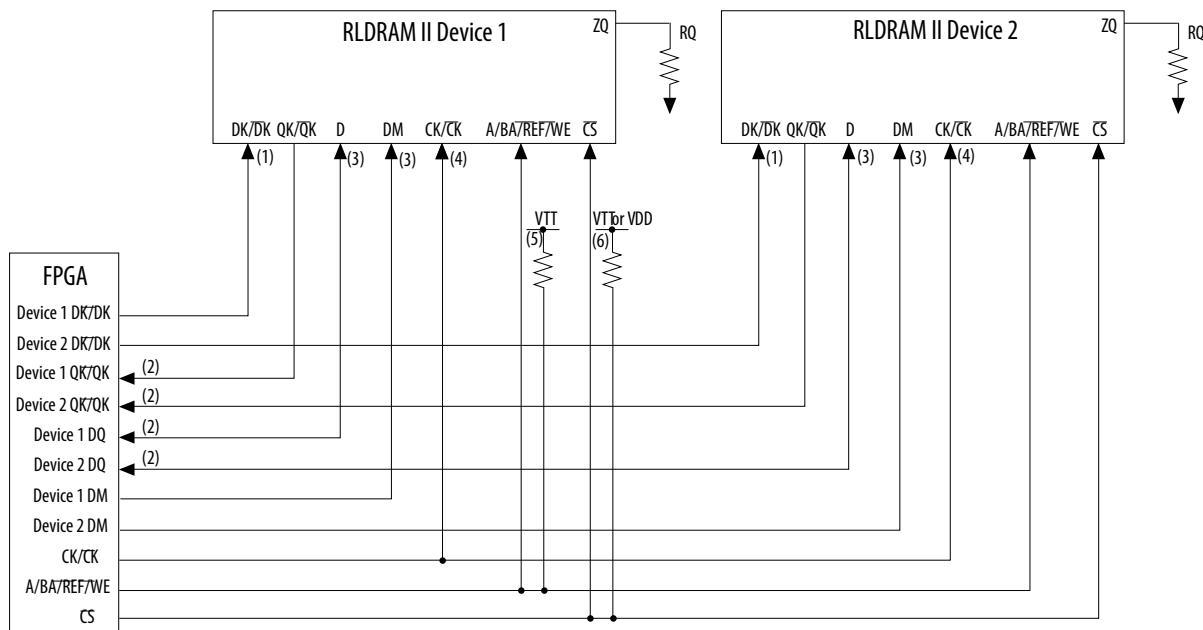


Notes to Figure:

1. Use external differential termination on DK/DK# and CK/CK#.
2. Use FPGA parallel on-chip termination (OCT) for terminating QK/QK# and DQ on reads.
3. Use RLDRAM II component on-die termination (ODT) for terminating DQ and DM on writes.
4. Use external discrete termination with fly-by placement to avoid stubs.
5. Use external discrete termination for this signal, as shown for REF.
6. Use external discrete termination, as shown for REF, but you may require a pull-up resistor to VDD as an alternative option. Refer to the RLDRAM II device data sheet for more information about RLDRAM II power-up sequencing.

The following figure shows the main signal connections between the FPGA and two CIO RLDRAM II components in a width expansion configuration.

**Figure 5-2: Configuration with Two CIO RLDRAM II Components in a Width Expansion Configuration**



Notes to Figure:

1. Use external differential termination on DK/DK#.
2. Use FPGA parallel on-chip termination (OCT) for terminating QK/QK# and DQ on reads.
3. Use RLDRAM II component on-die termination (ODT) for terminating DQ and DM on writes.
4. Use external dual  $200\ \Omega$  differential termination.
5. Use external discrete termination at the trace split of the balanced T or Y topology.
6. Use external discrete termination at the trace split of the balanced T or Y topology, but you may require a pull-up resistor to VDD as an alternative option. Refer to the RLDRAM II device data sheet for more information about RLDRAM II power-up sequencing.

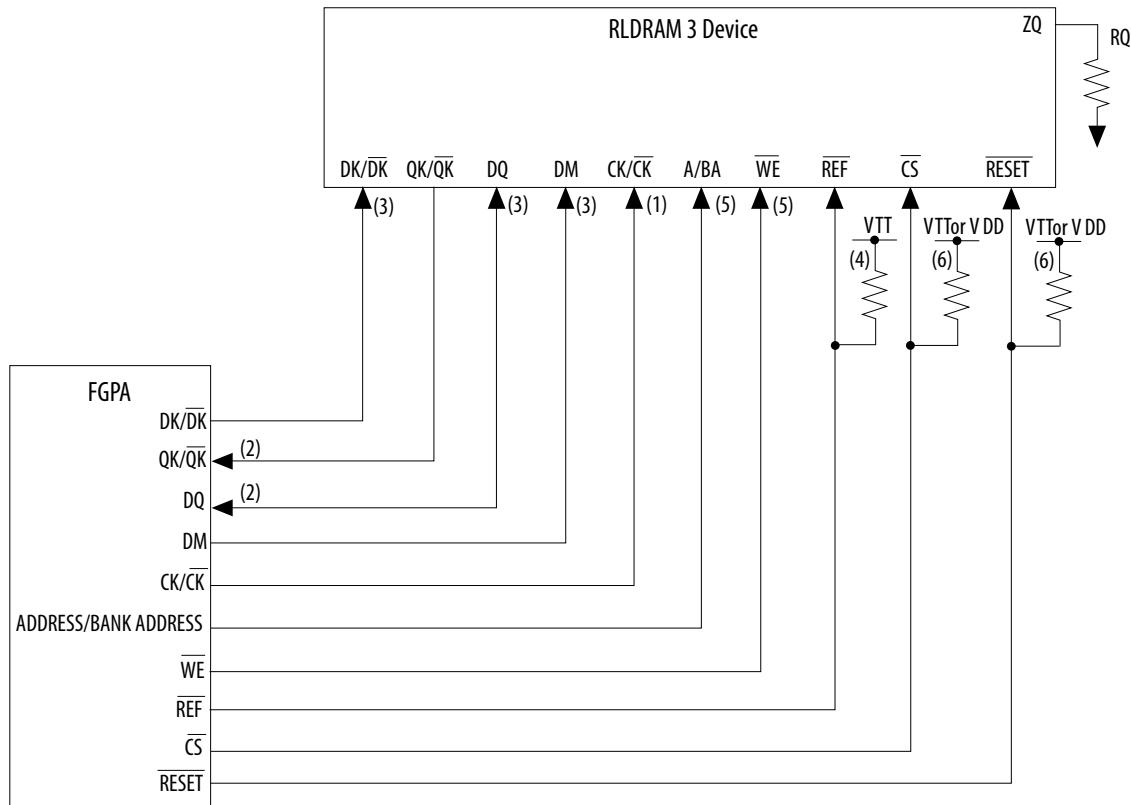
## RLDRAM 3 Configurations

The RLDRAM 3 UniPHY IP supports interfaces for CIO RLDRAM 3 with one or two devices. With two devices, the interface supports a width expansion configuration up to 72-bits. The termination and layout

principles for SIO RLDRAm 3 interfaces are similar to CIO RLDRAm 3, except that SIO RLDRAm 3 interfaces have unidirectional data buses.

The following figure shows the main signal connections between the FPGA and a single CIO RLDRAm 3 component.

**Figure 5-3: Configuration with a Single CIO RLDRAm 3 Component**

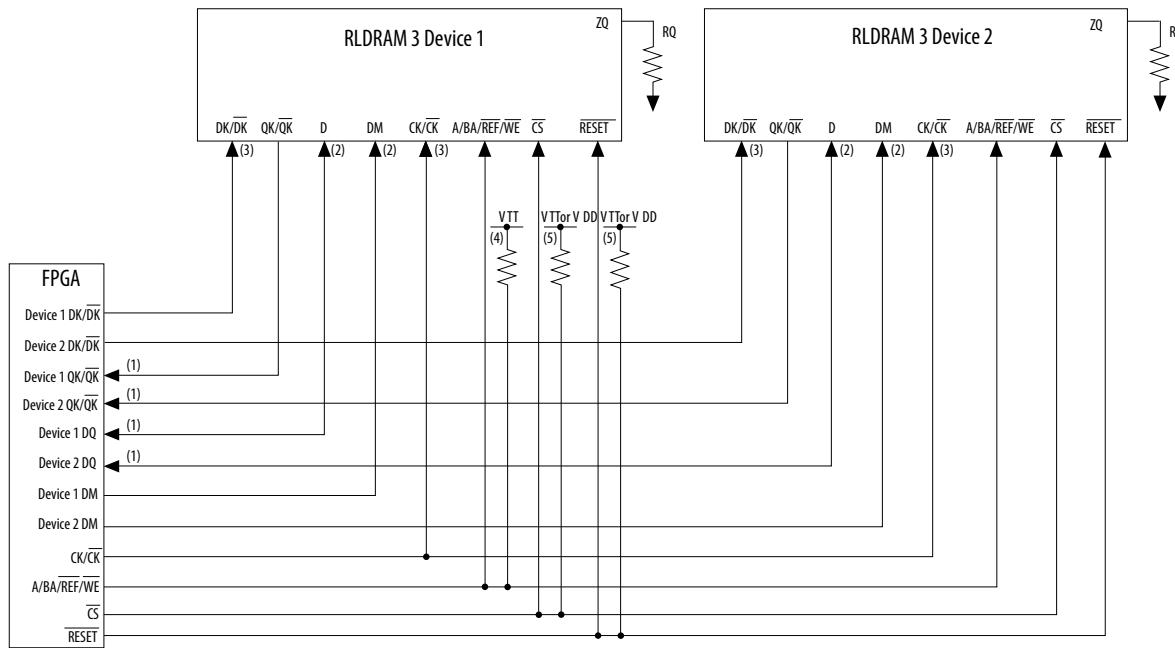


#### Notes to Figure:

1. Use external differential termination on CK/CK#.
2. Use FPGA parallel on-chip termination (OCT) for terminating QK/QK# and DQ on reads.
3. Use RLDRAm 3 component on-die termination (ODT) for terminating DQ, DM, and DK, DK# on writes.
4. Use external discrete termination with fly-by placement to avoid stubs.
5. Use external discrete termination for this signal, as shown for REF.
6. Use external discrete termination, as shown for REF, but you may require a pull-up resistor to VDD as an alternative option. Refer to the RLDRAm 3 device data sheet for more information about RLDRAm 3 power-up sequencing.

The following figure shows the main signal connections between the FPGA and two CIO RLDRAm 3 components in a width expansion configuration.

Figure 5-4: Configuration with Two CIO RLDRAM 3 Components in a Width Expansion Configuration



## Notes to Figure:

1. Use FPGA parallel OCT for terminating QK/QK# and DQ on reads.
2. Use RLDRAM 3 component ODT for terminating DQ, DM, and DK on writes.
3. Use external dual 200  $\Omega$  differential termination.
4. Use external discrete termination at the trace split of the balanced T or Y topology.
5. Use external discrete termination at the trace split of the balanced T or Y topology, but you may require a pull-up resistor to VDD as an alternative option. Refer to the RLDRAM 3 device data sheet for more information about RLDRAM 3 power-up sequencing.

## Signal Terminations

The following table lists the on-chip series termination ( $R_S$  OCT) and on-chip parallel termination ( $R_T$  OCT) schemes for supported devices.

**Note:** For RLDRAM 3, the default output termination resistance ( $R_S$ ) changes from 50 ohm to 25 ohm with the release of the Quartus II software version 12.1 SP1.

**Table 5-1: On-Chip Termination Schemes**

Termination Scheme	Class I Signal Standards	FPGA Device	
		Arria II GZ, Stratix III, and Stratix IV	Arria V and Stratix V
		Row/Column I/O	Row/Column I/O
$R_S$ OCT without Calibration	RLDRAM II - HSTL-15 and HSTL-18	50	50
	RLDRAM 3 - HSTL 1.2 V		
$R_S$ OCT with Calibration	RLDRAM II - HSTL-15 and HSTL-18	50	50 <sup>(1)</sup>
	RLDRAM 3 - HSTL 1.2 V		
$R_T$ OCT with Calibration	RLDRAM II - HSTL-15 and HSTL-18	50	50 <sup>(1)</sup>
	RLDRAM 3 - HSTL 1.2 V		

Note to Table:

1. Although 50-ohms is the recommended option, Stratix V devices offer a wider range of calibrated termination impedances.

RLDRAM II and RLDRAm 3 CIO interfaces have bidirectional data paths. The UniPHY IP uses dynamic OCT on the datapath, which switches between series OCT for memory writes and parallel OCT for memory reads. The termination schemes also follow these characteristics:

- Although 50 -ohm. is the recommended option, Stratix V devices offer a wider range of calibrated termination impedances.
- $R_S$  OCT supports output buffers.
- $R_T$  OCT supports input buffers.
- $R_S$  OCT supports bidirectional buffers only when they are driving output signals.
- $R_T$  OCT bidirectional buffers only when they are input signals.

For Arria II GZ, Stratix III, and Stratix IV devices, the HSTL Class I I/O calibrated terminations are calibrated against 50-ohm 1% resistors connected to the  $R_{UP}$  and  $R_{DN}$  pins in an I/O bank with the same  $V_{CCIO}$  as the RLDRAm II interface. For Arria V and Stratix V devices, the HSTL Class I I/O calibrated terminations are calibrated against 100-ohm 1% resistors connected to the  $R_{ZQ}$  pins in an I/O bank with the same  $V_{CCIO}$  as the RLDRAm II and RLDRAm 3 interfaces.

The calibration occurs at the end of the device configuration.

RLDRAM II and RLDRAm 3 memory components have a  $Z_Q$  pin that connects through a resistor  $R_Q$  to ground. Typically the RLDRAm II and RLDRAm 3 output signal impedance is a fraction of  $R_Q$ . Refer to the RLDRAm II and RLDRAm 3 device data sheets for more information.

For information about OCT, refer to the following:

- *I/O Features in Arria II Devices* chapter in the *Arria II Device Handbook*
- *I/O Features in Arria V Devices* chapter in the *Arria V Device Handbook*
- *Stratix III Device I/O Features* chapter in the *Stratix III Device Handbook*
- *I/O Features in Stratix IV Devices* chapter in the *Stratix IV Device Handbook*
- *I/O Features in Stratix V Devices* chapter in the *Stratix V Device Handbook*

Altera strongly recommends signal terminations to optimize signal integrity and timing margins, and to minimize unwanted emissions, reflections, and crosstalk.

**Note:** Simulate your design to check your termination scheme.

#### Related Information

- [I/O Features in Arria II Devices](#)
- [I/O Features in Arria V Devices](#)
- [Stratix III Device I/O Features](#)
- [I/O Features in Stratix IV Devices](#)
- [I/O Features in Stratix V Devices](#)

## Input to the FPGA from the RLDRAM Components

The RLDRAM II or RLDRAM 3 component drives the following input signals into the FPGA:

- Read data ( $D_Q$  on the bidirectional data signals for CIO RLDRAM II and CIO RLDRAM 3).
- Read clocks ( $Q_K/Q_K\#$ ).

Altera recommends that you use the FPGA parallel OCT to terminate the data on reads and read clocks.

## Outputs from the FPGA to the RLDRAM II and RLDRAM 3 Components

The following output signals are from the FPGA to the RLDRAM II and RLDRAM 3 components:

- Write data ( $D_Q$  on the bidirectional data signals for CIO RLDRAM II and RLDRAM 3)
- Data mask ( $D_M$ )
- Address, bank address
- Command ( $C_S$ ,  $W_E$ , and  $R_E F$ )
- Clocks ( $C_K/C_K\#$  and  $D_K/D_K\#$ )

For point-to-point single-ended signals requiring external termination, Altera recommends that you place a fly-by termination by terminating at the end of the transmission line after the receiver to avoid unterminated stubs. The guideline is to place the fly-by termination within 100 ps propagation delay of the receiver.

Although not recommended, you can place the termination before the receiver, which leaves an unterminated stub. The stub delay is critical because the stub between the termination and the receiver is effectively unterminated, causing additional ringing and reflections. Stub delays should be less than 50 ps.

Altera recommends that the differential clocks,  $C_K$ ,  $C_K\#$  and  $D_K$ ,  $D_K\#$  (RLDRAM II) and  $C_K$ ,  $C_K\#$  (RLDRAM 3), use a differential termination at the end of the trace at the external memory component. Alternatively, you can terminate each clock output with a parallel termination to VTT.

## RLDRAM II Termination Schemes

The following table lists the recommended termination schemes for major CIO RLDRAM II memory interface signals. These signals include data (DQ), data mask (DM), clocks (CK, CK#, DK, DK#, QK, and QK#), address, bank address, and command (WE#, REF#, and CS#).

**Table 5-2: RLDRAM II Termination Recommendations for Arria II GZ, Arria V, Stratix III, Stratix IV, and Stratix V Devices**

Signal Type	HSTL 15/18 Standard <sup>(1)</sup> <sup>(2)</sup> <sup>(3)</sup> <sup>(4)</sup>	Memory End Termination
DK/DK# Clocks	Class I R50 NO CAL	100 -ohm Differential
QK/QK# Clocks	Class I P50 CAL	ZQ50
Data (Write)	Class I R50 CAL	ODT
Data (Read)	Class I P50 CAL	ZQ50
Data Mask	Class I R50 CAL	ODT
CK/CK# Clocks	Class I R50 NO CAL	$\times 1 = 100\text{-ohm Differential}$ <sup>(9)</sup> $\times 2 = 200\text{-ohm Differential}$ <sup>(10)</sup>
Address/Bank Address <sup>(5)</sup> <sup>(6)</sup>	Class I Max Current	50 -ohm Parallel to V <sub>TT</sub>
Command (WE#, REF#) <sup>(5)</sup> <sup>(6)</sup>	Class I Max Current	50 -ohm Parallel to V <sub>TT</sub>
Command (CS#) <sup>(5)</sup> <sup>(6)</sup> <sup>(7)</sup>	Class I Max Current	50 -ohm Parallel to V <sub>TT</sub> or Pull-up to V <sub>DD</sub>
QVLD <sup>(8)</sup>	Class I P50 CAL	ZQ50

Signal Type	HSTL 15/18 Standard <sup>(1) (2) (3) (4)</sup>	Memory End Termination
Notes to Table:		
1.	R is effective series output impedance.	
2.	P is effective parallel input impedance.	
3.	CAL is OCT with calibration.	
4.	NO CAL is OCT without calibration.	
5.	For width expansion configuration, the address and control signals are routed to 2 devices. Recommended termination is 50 -ohm parallel to V <sub>TT</sub> at the trace split of a balanced T or Y routing topology. Use a clamshell placement of the two RLDRAM II components to achieve minimal stub delays and optimum signal integrity. Clamshell placement is when two devices overlay each other by being placed on opposite sides of the PCB.	
6.	The UniPHY default IP setting for this output is Max Current. A Class I 50 -ohm output with calibration output is typically optimal in single load topologies.	
7.	Altera recommends that you use a 50 -ohm parallel termination to V <sub>TT</sub> if your design meets the power sequencing requirements of the RLDRAM II component. Refer to the RLDRAM II data sheet for further information.	
8.	QVLD is not used in the RLDRAM II Controller with UniPHY implementations.	
9.	×1 is a single-device load.	
10.	×2 is a double-device load. An alternative option is to use a 100 -ohm differential termination at the trace split.	

**Note:** Altera recommends that you simulate your specific design for your system to ensure good signal integrity.

## RLDRAM 3 Termination Schemes

The following table lists the recommended termination schemes for major CIO RLDRAM 3 memory interface signals. These signals include data (DQ), data mask (DM), clocks (CK, CK#, DK, DK#, QK, and QK#), address, bank address, and command (WE#, REF#, and CS#).

**Table 5-3: RLDRAM 3 Termination Recommendations for Arria V GZ and Stratix V Devices**

Signal Type	Memory End Termination Option in the Chip (ODT)	Recommended On-Board Terminations
Data Read (DQ, QK)	40, 60 (series)	None
Data Write (DQ, DM, DK)	40, 60, 120 (parallel)	None
Address/Bank Address/ Command (WE#, REF#, CS#) <sup>(1) (2) (3)</sup>	None	50-ohm Parallel to V <sub>TT</sub>
CK/CK#	None	100 ohm Differential

Signal Type	Memory End Termination Option in the Chip (ODT)	Recommended On-Board Terminations
-------------	---	-----------------------------------

Notes to Table:

1. For width expansion configuration, the address and control signals are routed to 2 devices. Recommended termination is 50-ohm parallel to V<sub>TT</sub> at the trace split of a balanced T or Y routing topology. Use a clamshell placement of the two RLDRAM 3 components to achieve minimal stub delays and optimum signal integrity. Clamshell placement is when two devices overlay each other by being placed on opposite sides of the PCB.
2. The UniPHY default IP setting for this output is Max Current. A Class I 50-ohm output with calibration output is typically optimal in single load topologies.
3. Altera recommends that you use a 50-ohm parallel termination to V<sub>TT</sub> if your design meets the power sequencing requirements of the RLDRAM 3 component. Refer to the RLDRAM 3 data sheet for further information.
4. QVLD is not used in the RLDRAM 3 Controller with UniPHY implementations.
5. For information on the I/O standards and on-chip termination (OCT) resistance values supported for RLDRAM 3, refer to the *I/O Features* chapter of the appropriate device handbook.

Altera recommends that you simulate your specific design for your system to ensure good signal integrity.

## PCB Layout Guidelines

Altera recommends that you create your project in the Quartus® Prime software with a fully implemented RLDRAM II Controller with UniPHY interface, or RLDRAM 3 with UniPHY IP, and observe the interface timing margins to determine the actual margins for your design.

Although the recommendations in this chapter are based on simulations, you can apply the same general principles when determining the best termination scheme, drive strength setting, and loading style to any board designs. Altera recommends that you perform simulations, either using IBIS or HSPICE models, to determine the quality of signal integrity on your designs, and that you get accurate time base skew numbers when you simulate your specific implementation.

- Note:**
1. The following layout guidelines include several +/- length-based rules. These length-based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristics of your PCB implementation. They do not include any margin for crosstalk.
  2. To reliably close timing to and from the periphery of the device, signals to and from the periphery should be registered before any further logic is connected.

### Related Information

[Altera Power Distribution Network \(PDN\) Design Tool](#)

## General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the slew rate and propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

**Table 5-4: General Layout Guidelines**

Parameter	Guidelines
Impedance	<ul style="list-style-type: none"><li>• All unused via pads must be removed, because they cause unwanted capacitance.</li><li>• Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.</li></ul>
Decoupling Parameter	<ul style="list-style-type: none"><li>• Use 0.1 uF in 0402 size to minimize inductance</li><li>• Make VTT voltage decoupling close to pull-up resistors</li><li>• Connect decoupling caps between VTT and ground</li><li>• Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin</li><li>• Verify the capacitive decoupling using the Altera Power Distribution Network Design Tool</li></ul>
Power	<ul style="list-style-type: none"><li>• Route GND and V<sub>CC</sub> as planes</li><li>• Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation</li><li>• Route VTT as islands or 250-mil (6.35-mm) power traces</li><li>• Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces</li></ul>
General Routing	<p>All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Altera recommend that signals from the same net group always be routed on the same layer.</p> <ul style="list-style-type: none"><li>• Use 45° angles (<i>not</i> 90° corners)</li><li>• Avoid T-Junctions for critical nets or clocks</li><li>• Avoid T-junctions greater than 250 mils (6.35 mm)</li><li>• Disallow signals across split planes</li><li>• Restrict routing other signals close to system reset signals</li><li>• Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks</li></ul>

**Related Information**[Power Distribution Network Design Tool](#)

## RLDRAM II and RLDRAM 3 Layout Approach

Using the layout guidelines in the preceding table, Altera recommends the following layout approach:

1. If the RLDRAM II interface has multiple DQ groups ( $\times 18$  or  $\times 36$  RLDRAM II/RLDRAM 3 component or width expansion configuration), match all the  $\text{DK}/\text{DK}\#$  and  $\text{QK}, \text{QK}\#$  clocks as tightly as possible to optimize the timing margins in your design.
2. Route the  $\text{DK}/\text{DK}\#$  write clock and  $\text{QK}/\text{QK}\#$  read clock associated with a DQ group on the same PCB layer. Match these clock pairs to within  $\pm 5$  ps.
3. Set the  $\text{DK}/\text{DK}\#$  or  $\text{QK}/\text{QK}\#$  clock as the target trace propagation delay for the associated data and data mask signals.
4. Route the data and data mask signals for the DQ group ideally on the same layer as the associated  $\text{QK}/\text{QK}\#$  and  $\text{DK}/\text{DK}\#$  clocks to within  $\pm 10$  ps skew of the target clock.
5. Route the  $\text{CK}/\text{CK}\#$  clocks and set as the target trace propagation delays for the address/command signal group. Match the  $\text{CK}/\text{CK}\#$  clock to within  $\pm 50$  ps of all the  $\text{DK}/\text{DK}\#$  clocks.
6. Route the address/control signal group (address, bank address,  $\text{CS}$ ,  $\text{WE}$ , and  $\text{REF}$ ) ideally on the same layer as the  $\text{CK}/\text{CK}\#$  clocks, to within  $\pm 20$  ps skew of the  $\text{CK}/\text{CK}\#$  traces.

**Note:** It is important to match the delays of  $\text{CK}$  vs.  $\text{DK}$ , and  $\text{CK}$  vs.  $\text{Addr-Cmd}$  as much as possible.

This layout approach provides a good starting point for a design requirement of the highest clock frequency supported for the RLDRAM II and RLDRAM 3 interfaces.

For details on pin planning, refer to the *Planning Pin and FPGA Resources* chapter in the *External Memory Interface Handbook*.

### Related Information

[Planning Pin and FPGA Resources](#) on page 1-1

## RLDRAM II and RLDRAM 3 Layout Guidelines

The following table lists the RLDRAM II and RLDRAM 3 general routing layout guidelines.

**Table 5-5: RLDRAM II and RLDRAM 3 Layout Guidelines**

Parameter	Guidelines
General Routing	<ul style="list-style-type: none"> <li>• If you must route signals of the same net group on different layers with the same impedance characteristic, simulate your worst case PCB trace tolerances to ascertain actual propagation delay differences. Typical layer to layer trace delay variations are of 15 ps/inch order.</li> <li>• Avoid T-junctions greater than 150 ps.</li> <li>• Match all signals within a given DQ group with a maximum skew of <math>\pm 10</math> ps and route on the same layer.</li> </ul>

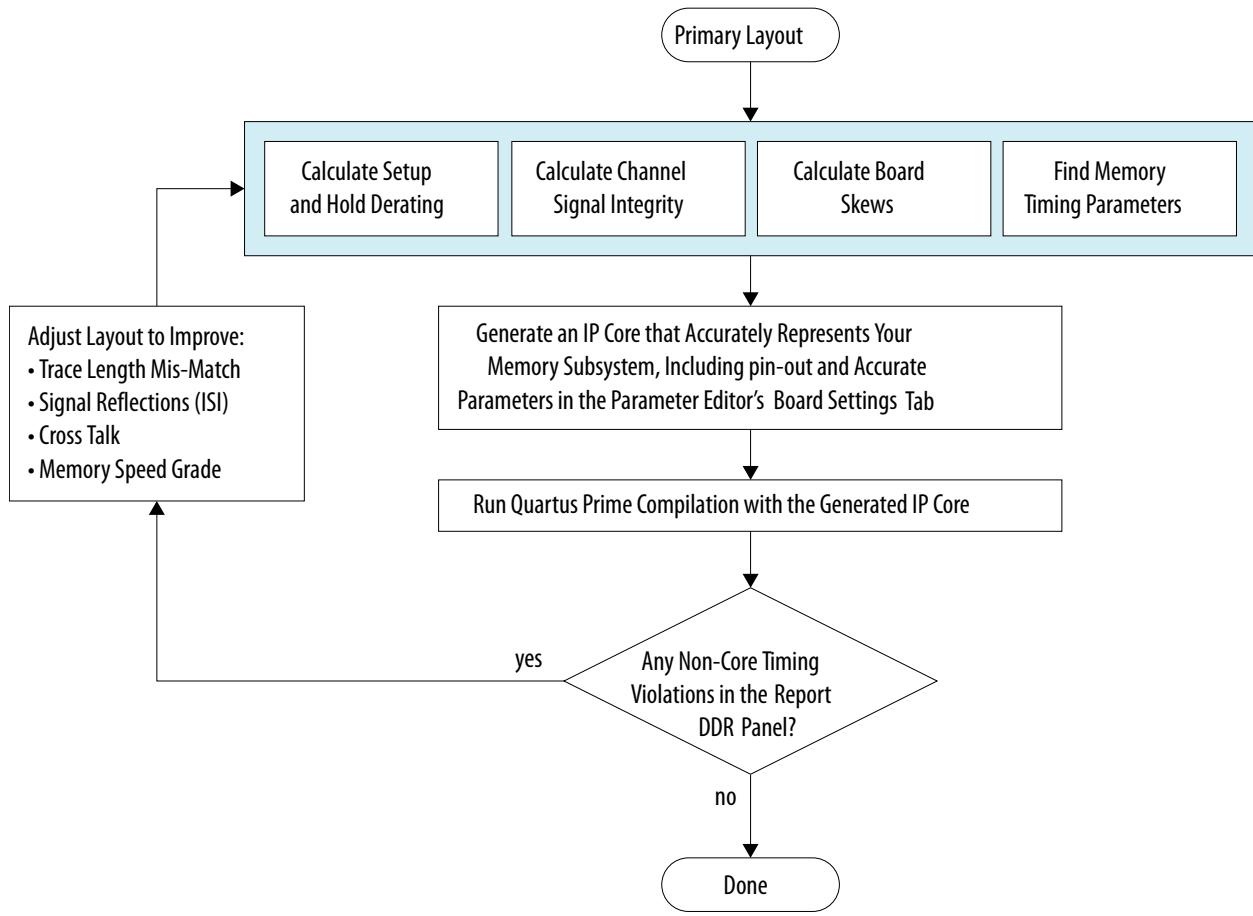
Parameter	Guidelines
Clock Routing	<ul style="list-style-type: none"><li>Route clocks on inner layers with outer-layer run lengths held to under 150 ps.</li><li>These signals should maintain a 10-mil (0.254 mm) spacing from other nets.</li><li>Clocks should maintain a length-matching between clock pairs of <math>\pm 5</math> ps.</li><li>Differential clocks should maintain a length-matching between P and N signals of <math>\pm 2</math> ps.</li><li>Space between different clock pairs should be at least three times the space between the traces of a differential pair.</li></ul>
Address and Command Routing	<ul style="list-style-type: none"><li>To minimize crosstalk, route address, bank address, and command signals on a different layer than the data and data mask signals.</li><li>Do not route the differential clock signals close to the address signals.</li><li>Keep the distance from the pin on the RLDRAM II or RLDRAM 3 component to the stub termination resistor (<math>V_{TT}</math>) to less than 50 ps for the address/command signal group.</li><li>Keep the distance from the pin on the RLDRAM II or RLDRAM 3 component to the fly-by termination resistor (<math>V_{TT}</math>) to less than 100 ps for the address/command signal group.</li></ul>
External Memory Routing Rules	<ul style="list-style-type: none"><li>Apply the following parallelism rules for the RLDRAM II or RLDRAM 3 data/address/command groups:<ul style="list-style-type: none"><li>4 mils for parallel runs &lt; 0.1 inch (approximately <math>1\times</math> spacing relative to plane distance).</li><li>5 mils for parallel runs &lt; 0.5 inch (approximately <math>1\times</math> spacing relative to plane distance).</li><li>10 mils for parallel runs between 0.5 and 1.0 inches (approximately <math>2\times</math> spacing relative to plane distance).</li><li>15 mils for parallel runs between 1.0 and 3.3 inch (approximately <math>3\times</math> spacing relative to plane distance).</li></ul></li></ul>
Maximum Trace Length	<ul style="list-style-type: none"><li>Keep the maximum trace length of all signals from the FPGA to the RLDRAM II or RLDRAM 3 components to 600 ps.</li></ul>

## Layout Approach

For all practical purposes, you can regard the TimeQuest timing analyzer's report on your memory interface as definitive for a given set of memory and board timing parameters.

You will find timing under **Report DDR** in TimeQuest and on the **Timing Analysis** tab in the parameter editor.

The following flowchart illustrates the recommended process to follow during the board design phase, to determine timing margin and make iterative improvements to your design.



## Setup and Hold Derating

For information on calculating derating parameters, refer to *Implementing and Parameterizing Memory IP*, in the *External Memory Interface Handbook*.

## Channel Signal Integrity

For information on determining channel signal integrity for Stratix V and earlier products, refer to the wiki page: [http://www.alterawiki.com/wiki/Measuring\\_Channel\\_Signal\\_Integrity](http://www.alterawiki.com/wiki/Measuring_Channel_Signal_Integrity).

For information on Arria 10 board timing and layout approach, refer to: [http://www.alterawiki.com/wiki/Arria\\_10\\_EMIF\\_Simulation\\_Guidance](http://www.alterawiki.com/wiki/Arria_10_EMIF_Simulation_Guidance).

## Board Skew

For information on calculating board skew parameters, refer to *Implementing and Parameterizing Memory IP*, in the *External Memory Interface Handbook*.

The Board Skew Parameter Tool is an interactive tool that can help you calculate board skew parameters if you know the absolute delay values for all the memory related traces.

## Memory Timing Parameters

For information on the memory timing parameters to be entered into the parameter editor, refer to the datasheet for your external memory device.

### Related Information

[Board Skew Parameter Tool](#)

## Arria V and Stratix V Board Setting Parameters

The following guidelines apply to the Board Setting parameters for Arria V and Stratix V devices.

### Setup and Hold Derating

For information on calculating derating parameters, refer to *Implementing and Parameterizing Memory IP*, in the *External Memory Interface Handbook*.

### Channel Signal Integrity

For information on determining channel signal integrity for Stratix V and earlier products, refer to the wiki page: [http://www.alterawiki.com/wiki/Measuring\\_Channel\\_Signal\\_Integrity](http://www.alterawiki.com/wiki/Measuring_Channel_Signal_Integrity).

### Board Skew

For information on calculating board skew parameters, refer to *Implementing and Parameterizing Memory IP*, in the *External Memory Interface Handbook*.

The Board Skew Parameter Tool is an interactive tool that can help you calculate board skew parameters if you know the absolute delay values for all the memory related traces.

### Memory Timing Parameters

For information on the memory timing parameters to be entered into the parameter editor, refer to the datasheet for your external memory device.

## Arria 10 Board Setting Parameters

For Board Setting and layout approach information for Arria 10 devices, refer to the Altera wiki at the address below.

Arria 10 Board Setting and layout approach information: [http://www.alterawiki.com/wiki/Arria\\_10\\_EMIF\\_Simulation\\_Guidance](http://www.alterawiki.com/wiki/Arria_10_EMIF_Simulation_Guidance).

## Package Deskew for RLDRAM II and RLDRAM 3

You should follow Altera's package deskew guidance if you are using Stratix V or Arria 10 devices.

For more information on package deskew, refer to *Package Deskew*.

### Related Information

[Package Deskew](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	<ul style="list-style-type: none"> <li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> </ul>
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	<ul style="list-style-type: none"> <li>Revised <i>RLDRAM 3 Termination Recommendations for Arria V GZ and Stratix V Devices</i> table.</li> <li>Removed millimeter approximations from lengths expressed in picoseconds in <i>RLDRAM II and RLDRAM 3 Layout Guidelines</i> table.</li> <li>Minor formatting fixes in <i>RLDRAM II and RLDRAM 3 Layout Guidelines</i> table.</li> <li>Added <i>Layout Approach</i> section.</li> </ul>
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>Added note about byteenable support to <i>Signal Descriptions</i> section.</li> <li>Consolidated General Layout Guidelines.</li> </ul>
November 2012	3.2	Added content supporting RLDRAM 3 and updated RLDRAM II standards.
June 2012	3.1	Added Feedback icon.
November 2011	3.0	Added Arria V information.
June 2011	2.0	Added Stratix V information.
December 2010	1.0	Initial release.

# QDR II and QDR-IV SRAM Board Design Guidelines

6

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The following topics provide guidelines for you to improve your system's signal integrity and layout guidelines to help successfully implement a QDR II or QDR II+ SRAM interface in your system.

The QDR II and QDR II+ SRAM Controller with UniPHY intellectual property (IP) enables you to implement QDR II and QDR II+ interfaces with Arria® II GX, Arria V, Stratix® III, Stratix IV, and Stratix V devices.

**Note:** In the following topics, QDR II SRAM refers to both QDR II and QDR II+ SRAM unless stated otherwise.

The following topics focus on the following key factors that affect signal integrity:

- I/O standards
- QDR II SRAM configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

## I/O Standards

QDR II SRAM interface signals use one of the following JEDEC I/O signalling standards:

- HSTL-15—provides the advantages of lower power and lower emissions.
- HSTL-18—provides increased noise immunity with slightly greater output voltage swings.

To select the most appropriate standard for your interface, refer to the *Arria II GX Devices Data Sheet: Electrical Characteristics* chapter in the *Arria II Device Handbook*, *Stratix III Device Datasheet: DC and Switching Characteristics* chapter in the *Stratix III Device Handbook*, or the *Stratix IV Device Datasheet DC and Switching Characteristics* chapter in the *Stratix IV Device Handbook*.

Altera® QDR II SRAM Controller with UniPHY IP defaults to HSTL 1.5 V Class I outputs and HSTL 1.5 V inputs.

## Related Information

- [Arria II GX Devices Data Sheet: Electrical Characteristics](#)
- [Stratix III Device Datasheet: DC and Switching Characteristics](#)
- [Stratix IV Device Datasheet DC and Switching Characteristics](#)

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

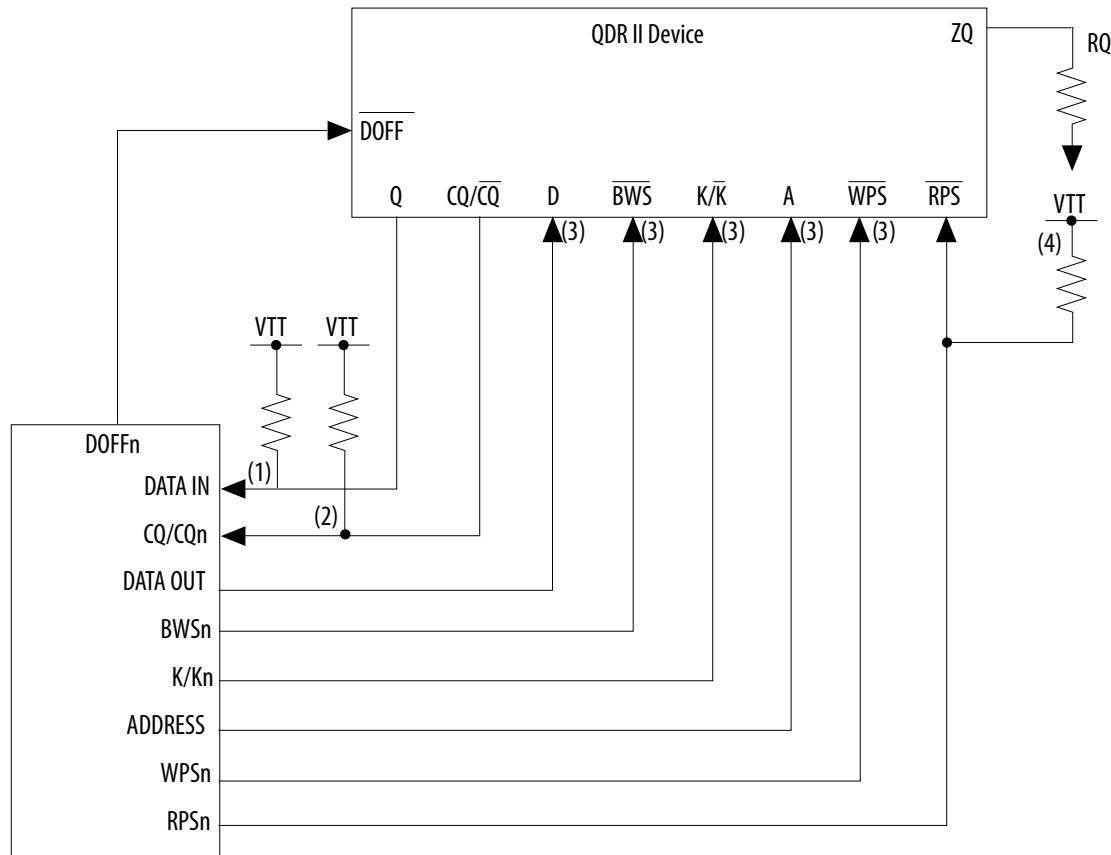
ISO  
9001:2008  
Registered

## QDR II SRAM Configurations

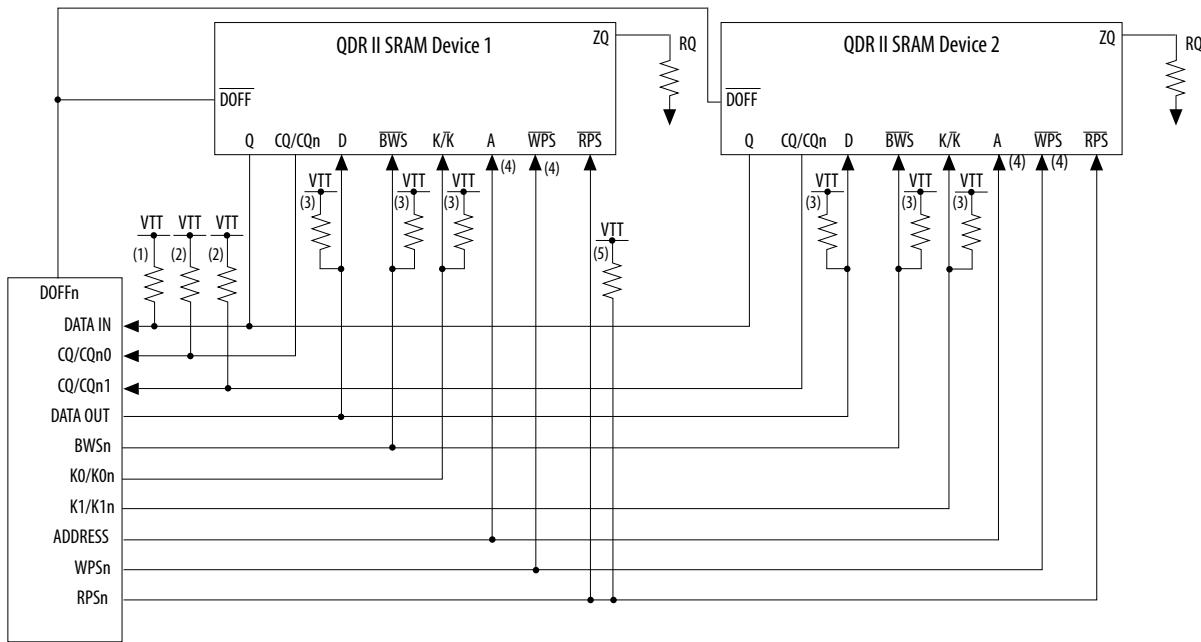
The QDR II SRAM Controller with UniPHY IP supports interfaces with a single device, and two devices in a width expansion configuration up to maximum width of 72 bits.

The following figure shows the main signal connections between the FPGA and a single QDR II SRAM component.

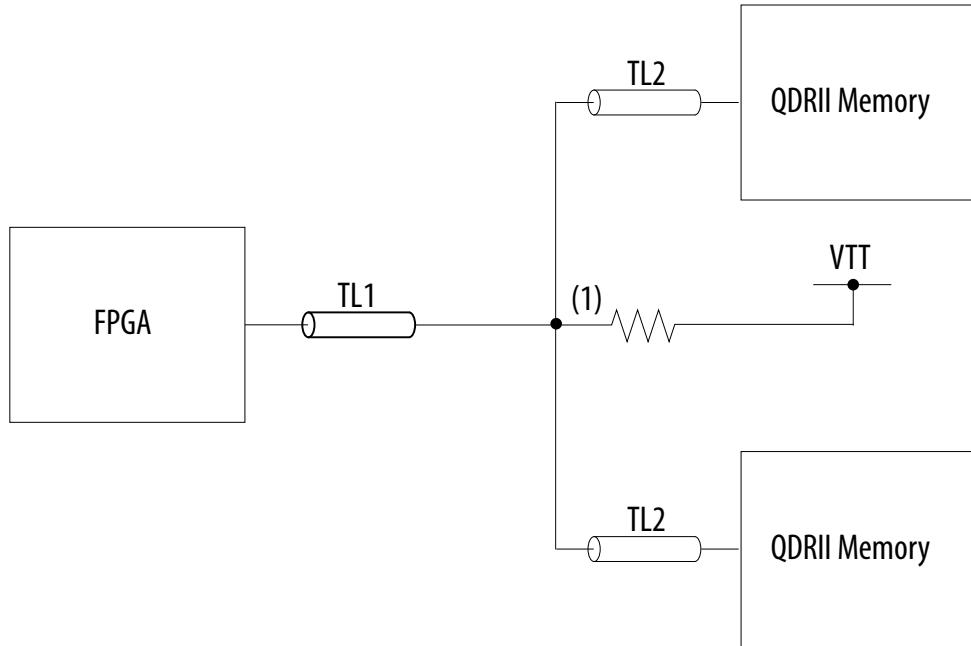
**Figure 6-1: Configuration With A Single QDR II SRAM Component**



The following figure shows the main signal connections between the FPGA and two QDR II SRAM components in a width expansion configuration.

**Figure 6-2: Configuration With Two QDR II SRAM Components In A Width Expansion Configuration**

The following figure shows the detailed balanced topology recommended for the address and command signals in the width expansion configuration.

**Figure 6-3: External Parallel Termination for Balanced Topology**

## Signal Terminations

Arria II GX, Stratix III and Stratix IV devices offer on-chip termination (OCT) technology.

The following table summarizes the extent of OCT support for each device.

**Table 6-1: On-Chip Termination Schemes<sup>(1)</sup>**

Termination Scheme	HSTL-15 and HSTL-18	FPGA Device					
		Arria II GX		Arria II GZ, Stratix III, and Stratix IV		Arria V and Stratix V	
		Column I/O	Row I/O	Column I/O	Row I/O	Column I/O	Row I/O
On-Chip Series Termination without Calibration	Class I	50	50	50	50	—	—
On-Chip Series Termination with Calibration	Class I	50	50	50	50	—	—
On-Chip Parallel Termination with Calibration	Class I	—	—	50	50	50	50

Note to Table:

1. This table provides information about HSTL-15 and HSTL-18 standards because these are the supported I/O standards for QDR II SRAM memory interfaces by Altera FPGAs.

On-chip series ( $R_S$ ) termination is supported only on output and bidirectional buffers, while on-chip parallel ( $R_T$ ) termination is supported only on input and bidirectional buffers. Because QDR II SRAM interfaces have unidirectional data paths, dynamic OCT is not required.

For Arria II GX, Stratix III and Stratix IV devices, the HSTL Class I I/O calibrated terminations are calibrated against 50-ohm 1% resistors connected to the  $R_{UP}$  and  $R_{DN}$  pins in an I/O bank with the same VCCIO as the QDRII SRAM interface. The calibration occurs at the end of the device configuration.

QDR II SRAM controllers have a ZQ pin which is connected via a resistor RQ to ground. Typically the QDR II SRAM output signal impedance is  $0.2 \times R_Q$ . Refer to the QDR II SRAM device data sheet for more information.

For information about OCT, refer to the *I/O Features in Arria II GX Devices* chapter in the *Arria II GX Device Handbook*, *I/O Features in Arria V Devices* chapter in the *Arria V Device Handbook*, *Stratix III Device I/O Features* chapter in the *Stratix III Device Handbook*, *I/O Features in Stratix IV Devices* chapter

in the *Stratix IV Device Handbook*, and the *I/O Features in Stratix V Devices* chapter in the *Stratix V Device Handbook*.

#### Related Information

- [I/O Features in Arria II GX Devices](#)
- [I/O Features in Arria V Devices](#)
- [Stratix III Device I/O Features](#)
- [I/O Features in Stratix IV Devices](#)
- [I/O Features in Stratix V Devices](#)

## Output from the FPGA to the QDR II SRAM Component

The following output signals are from the FPGA to the QDR II SRAM component:

- write data
- byte write select ( $\text{BWSn}$ )
- address
- control ( $\text{WPSn}$  and  $\text{RPSn}$ )
- clocks,  $\text{k}/\text{k}\#$

Altera recommends that you terminate the write clocks,  $\text{k}$  and  $\text{k}\#$ , with a single-ended fly-by 50-ohm parallel termination to  $V_{TT}$ . However, simulations show that you can consider a differential termination if the clock pair is well matched and routed differentially.

Altera strongly recommends signal terminations to optimize signal integrity and timing margins, and to minimize unwanted emissions, reflections, and crosstalk.

For point-to-point signals, Altera recommends that you place a fly-by termination by terminating at the end of the transmission line after the receiver to avoid unterminated stubs. The guideline is to place the fly-by termination within 100 ps propagation delay of the receiver.

Although not recommended, you can place the termination before the receiver, which leaves an unterminated stub. The stub delay is critical because the stub between the termination and the receiver is effectively unterminated, causing additional ringing and reflections. Stub delays should be less than 50 ps.

**Note:** Simulate your design to ensure correct functionality.

## Input to the FPGA from the QDR II SRAM Component

The QDR II SRAM component drives the following input signals into the FPGA:

- read data
- echo clocks,  $\text{CQ}/\text{CQ}\#$

For point-to-point signals, Altera recommends that you use the FPGA parallel OCT wherever possible. For devices that do not support parallel OCT (Arria II GX), and for  $\times 36$  emulated configuration  $\text{CQ}/\text{CQ}\#$  termination, Altera recommends that you use a fly-by 50-ohm parallel termination to  $V_{TT}$ . Although not recommended, you can use parallel termination with a short stub of less than 50 ps propagation delay as an alternative option. The input echo clocks,  $\text{CQ}$  and  $\text{CQ}\#$  must not use a differential termination.

## Termination Schemes

The following tables list the recommended termination schemes for major QDR II SRAM memory interface signals.

These signals include write data (D), byte write select (BWS), read data (Q), clocks (K, K#, CQ, and CQ#), address and command (WPS and RPS).

**Table 6-2: Termination Recommendations for Arria II GX Devices**

Signal Type	HSTL 15/18 Standard (1) (2)	FPGA End Discrete Termination	Memory End Termination
K/K# Clocks	Class I R50 CAL	—	50-ohm Parallel to V <sub>TT</sub>
Write Data	Class I R50 CAL	—	50-ohm Parallel to V <sub>TT</sub>
BWS	Class I R50 CAL	—	50-ohm Parallel to V <sub>TT</sub>
Address (3) (4)	Class I Max Current	—	50-ohm Parallel to V <sub>TT</sub>
WPS, RPS (3) (4)	Class I Max Current	—	50-ohm Parallel to V <sub>TT</sub>
CQ/CQ#	Class I	50-ohm Parallel to V <sub>TT</sub>	ZQ50
CQ/CQ# ×36 emulated (5)	Class I	50-ohm Parallel to V <sub>TT</sub>	ZQ50
Read Data (Q)	Class I	50-ohm Parallel to V <sub>TT</sub>	ZQ50
QVLD (6)	—	—	ZQ50

Notes to Table:

1. R is effective series output impedance.
2. CAL is calibrated OCT.
3. For width expansion configuration, the address and control signals are routed to 2 devices. Recommended termination is 50 -ohm parallel to V<sub>TT</sub> at the trace split of a balanced T or Y routing topology. For 400 MHz burst length 2 configurations where the address signals are double data rate, it is recommended to use a clamshell placement of the two QDR II SRAM components to achieve minimal stub delays and optimum signal integrity. Clamshell placement is when two devices overlay each other by being placed on opposite sides of the PCB.
4. A Class I 50 -ohm output with calibration output is typically optimal in double load topologies.
5. For ×36 emulated mode, the recommended termination for the CQ/CQ# signals is a 50 -ohm parallel termination to V<sub>TT</sub> at the trace split. Altera recommends that you use this termination when ×36 DQ/DQS groups are not supported in the FPGA.
6. QVLD is not used in the QDR II or QDR II+ SRAM with UniPHY implementations.

**Table 6-3: Termination Recommendations for Arria V, Stratix III, Stratix IV, and Stratix V Devices**

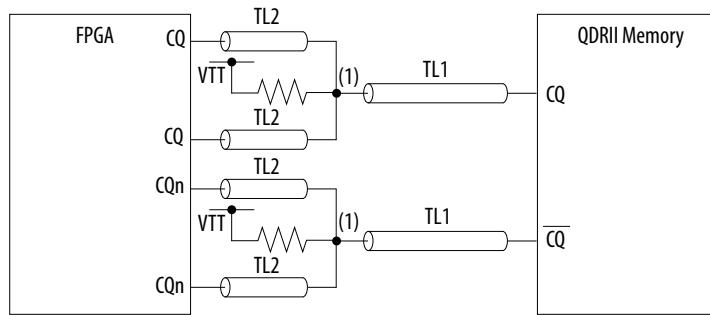
Signal Type	HSTL 15/18 Standard (1) (2) (3)	FPGA End Discrete Termination	Memory End Termination
K/K# Clocks	DIFF Class I R50 NO CAL	—	Series 50 -ohm Without Calibration
Write Data	Class I R50 CAL	—	50 -ohm Parallel to V <sub>TT</sub>
BWS	Class I R50 CAL	—	50 -ohm Parallel to V <sub>TT</sub>
Address <sup>(4)</sup> <sup>(5)</sup>	Class I Max Current	—	50 -ohm Parallel to V <sub>TT</sub>
WPS, RPS <sup>(4)</sup> <sup>(5)</sup>	Class I Max Current	—	50 -ohm Parallel to V <sub>TT</sub>
CQ/CQ#	Class I P50 CAL	—	ZQ50
CQ/CQ# ×36 emulated <sup>(6)</sup>	—	50 -ohm Parallel to V <sub>TT</sub>	ZQ50
Read Data (Q)	Class I P50 CAL	—	ZQ50
QVLD <sup>(7)</sup>	Class I P50 CAL	—	ZQ50

Notes to Table:

1. R is effective series output impedance.
2. P is effective parallel input impedance.
3. CAL is calibrated OCT.
4. For width expansion configuration, the address and control signals are routed to 2 devices. Recommended termination is 50-ohm parallel to V<sub>TT</sub> at the trace split of a balanced T or Y routing topology. For 400 MHz burst length 2 configurations where the address signals are double data rate, it is recommended to use a "clam shell" placement of the two QDR II SRAM components to achieve minimal stub delays and optimum signal integrity. "Clam shell" placement is when two devices overlay each other by being placed on opposite sides of the PCB.
5. The UniPHY default IP setting for this output is Max Current. A Class 1 50-ohm output with calibration output is typically optimal in single load topologies.
6. For ×36 emulated mode, the recommended termination for the CQ/CQ# signals is a 50-ohm parallel termination to V<sub>TT</sub> at the trace split. Altera recommends that you use this termination when ×36 DQ/DQS groups are not supported in the FPGA.
7. QVLD is not used in the QDR II or QDR II+ SRAM Controller with UniPHY implementations.

**Note:** Altera recommends that you simulate your specific design for your system to ensure good signal integrity.

For a ×36 QDR II SRAM interface that uses an emulated mode of two ×18 DQS groups in the FPGA, there are two CQ/CQ# connections at the FPGA and a single CQ/CQ# output from the QDR II SRAM device. Altera recommends that you use a balanced T topology with the trace split close to the FPGA and a parallel termination at the split, as shown in the following figure.

**Figure 6-4: Emulated ×36 Mode CQ/CQn Termination Topology**

For more information about ×36 emulated modes, refer to the “Exceptions for ×36 Emulated QDR II and QDR II+ SRAM Interfaces in Arria II GX, Stratix III, and Stratix IV Devices” section in the *Planning Pin and Resources* chapter.

#### Related Information

[Planning Pin and FPGA Resources](#) on page 1-1

## General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Altera recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the slew rate and propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

**Table 6-4: General Layout Guidelines**

Parameter	Guidelines
Impedance	<ul style="list-style-type: none"> <li>• All unused via pads must be removed, because they cause unwanted capacitance.</li> <li>• Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.</li> </ul>
Decoupling Parameter	<ul style="list-style-type: none"> <li>• Use 0.1 uF in 0402 size to minimize inductance</li> <li>• Make VTT voltage decoupling close to pull-up resistors</li> <li>• Connect decoupling caps between VTT and ground</li> <li>• Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin</li> <li>• Verify the capacitive decoupling using the Altera Power Distribution Network Design Tool</li> </ul>

Parameter	Guidelines
Power	<ul style="list-style-type: none"> <li>Route GND and V<sub>CC</sub> as planes</li> <li>Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation</li> <li>Route VTT as islands or 250-mil (6.35-mm) power traces</li> <li>Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces</li> </ul>
General Routing	<p>All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Altera recommend that signals from the same net group always be routed on the same layer.</p> <ul style="list-style-type: none"> <li>Use 45° angles (<i>not</i> 90° corners)</li> <li>Avoid T-Junctions for critical nets or clocks</li> <li>Avoid T-junctions greater than 250 mils (6.35 mm)</li> <li>Disallow signals across split planes</li> <li>Restrict routing other signals close to system reset signals</li> <li>Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks</li> </ul>

**Related Information**[Power Distribution Network Design Tool](#)

## QDR II Layout Guidelines

The following table summarizes QDR II and QDR II SRAM general routing layout guidelines.

- Note:**
1. The following layout guidelines include several +/- length based rules. These length based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristics of your PCB implementation. They do not include any margin for crosstalk.
  2. Altera recommends that you get accurate time base skew numbers when you simulate your specific implementation.
  3. To reliably close timing to and from the periphery of the device, signals to and from the periphery should be registered before any further logic is connected.

**Table 6-5: QDR II and QDR II+ SRAM Layout Guidelines**

Parameter	Guidelines
General Routing	<ul style="list-style-type: none"> <li>If signals of the same net group must be routed on different layers with the same impedance characteristic, you must simulate your worst case PCB trace tolerances to ascertain actual propagation delay differences. Typical later to later trace delay variations are of 15 ps/inch order.</li> <li>Avoid T-junctions greater than 150 ps.</li> </ul>

Parameter	Guidelines
Clock Routing	<ul style="list-style-type: none"> <li>Route clocks on inner layers with outer-layer run lengths held to under 150 ps.</li> <li>These signals should maintain a 10-mil (0.254 mm) spacing from other nets.</li> <li>Clocks should maintain a length-matching between clock pairs of <math>\pm 5</math> ps.</li> <li>Complementary clocks should maintain a length-matching between P and N signals of <math>\pm 2</math> ps.</li> <li>Keep the distance from the pin on the QDR II SRAM component to stub termination resistor (<math>V_{TT}</math>) to less than 50 ps for the <math>k, k\#</math> clocks.</li> <li>Keep the distance from the pin on the QDR II SRAM component to fly-by termination resistor (<math>V_{TT}</math>) to less than 100 ps for the <math>k, k\#</math> clocks.</li> <li>Keep the distance from the pin on the FPGA component to stub termination resistor (<math>V_{TT}</math>) to less than 50 ps for the echo clocks, <math>c_Q, c_Q\#</math>, if they require an external discrete termination.</li> <li>Keep the distance from the pin on the FPGA component to fly-by termination resistor (<math>V_{TT}</math>) to less than 100 ps for the echo clocks, <math>c_Q, c_Q\#</math>, if they require an external discrete termination.</li> </ul>
External Memory Routing Rules	<ul style="list-style-type: none"> <li>Keep the distance from the pin on the QDR II SRAM component to stub termination resistor (<math>V_{TT}</math>) to less than 50 ps for the write data, byte write select and address/command signal groups.</li> <li>Keep the distance from the pin on the QDR II SRAM component to fly-by termination resistor (<math>V_{TT}</math>) to less than 100 ps for the write data, byte write select and address/command signal groups.</li> <li>Keep the distance from the pin on the FPGA (Arria II GX) to stub termination resistor (<math>V_{TT}</math>) to less than 50 ps for the read data signal group.</li> <li>Keep the distance from the pin on the FPGA (Arria II GX) to fly-by termination resistor (<math>V_{TT}</math>) to less than 100 ps for the read data signal group.</li> <li>Parallelism rules for the QDR II SRAM data/address/command groups are as follows: <ul style="list-style-type: none"> <li>4 mils for parallel runs &lt; 0.1 inch (approximately 1x spacing relative to plane distance).</li> <li>5 mils for parallel runs &lt; 0.5 inch (approximately 1x spacing relative to plane distance).</li> <li>10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2x spacing relative to plane distance).</li> <li>15 mils for parallel runs between 1.0 and 6.0 inch (approximately 3x spacing relative to plane distance).</li> </ul> </li> </ul>
Maximum Trace Length	<ul style="list-style-type: none"> <li>Keep the maximum trace length of all signals from the FPGA to the QDR II SRAM components to 6 inches.</li> </ul>

**Related Information**[Altera Power Distribution Network \(PDN\) Design tool](#)

## QDR II SRAM Layout Approach

Using the layout guidelines in the above table, Altera recommends the following layout approach:

1. Route the K/K# clocks and set the clocks as the target trace propagation delays for the output signal group.
2. Route the write data output signal group (`write data`, `byte write select`), ideally on the same layer as the K/K# clocks, to within  $\pm 10$  ps skew of the K/K# traces.
3. Route the address/control output signal group (`address`, `RPS`, `WPS`), ideally on the same layer as the K/K# clocks, to within  $\pm 20$  ps skew of the K/K# traces.
4. Route the CQ/CQ# clocks and set the clocks as the target trace propagation delays for the input signal group.
5. Route the read data output signal group (`read data`), ideally on the same layer as the CQ/CQ# clocks, to within  $\pm 10$  ps skew of the CQ/CQ# traces.
6. The output and input groups do not need to have the same propagation delays, but they must have all the signals matched closely within the respective groups.

The following tables list the typical margins for QDR II and QDR II+ SRAM interfaces, with the assumption that there is zero skew between the signal groups.

**Table 6-6: Typical Worst Case Margins for QDR II SRAM Interfaces of Burst Length 2**

Device	Speed Grade	Frequency (MHz)	Typical Margin Address/ Command (ps)	Typical Margin Write Data (ps)	Typical Margin Read Data (ps)
Arria II GX	I5	250	$\pm 240$	$\pm 80$	$\pm 170$
Arria II GX $\times 36$ emulated	I5	200	$\pm 480$	$\pm 340$	$\pm 460$
Stratix IV	—	350	—	—	—
Stratix IV $\times 36$ emulated	C2	300	$\pm 320$	$\pm 170$	$\pm 340$

**Table 6-7: Typical Worst Case Margins for QDR II+ SRAM Interfaces of Burst Length 4**

Device	Speed Grade	Frequency (MHz)	Typical Margin Address/ Command (ps) <sup>(1)</sup>	Typical Margin Write Data (ps)	Typical Margin Read Data (ps)
Arria II GX	I5	250	$\pm 810$	$\pm 150$	$\pm 130$
Arria II GX $\times 36$ emulated	I5	200	$\pm 1260$	$\pm 410$	$\pm 420$
Stratix IV	C2	400	$\pm 550$	$\pm 10$	$\pm 80$
Stratix IV $\times 36$ emulated	C2	300	$\pm 860$	$\pm 180$	$\pm 300$

Device	Speed Grade	Frequency (MHz)	Typical Margin Address/ Command (ps) <sup>(1)</sup>	Typical Margin Write Data (ps)	Typical Margin Read Data (ps)
--------	-------------	-----------------	--	--------------------------------	-------------------------------

Note to Table:

1. The QDR II+ SRAM burst length of 4 designs have greater margins on the address signals because they are single data rate.

Other devices and speed grades typically show higher margins than the ones in the above tables.

**Note:** Altera recommends that you create your project with a fully implemented QDR II or QDR II+ SRAM Controller with UniPHY interface, and observe the interface timing margins to determine the actual margins for your design.

Although the recommendations in this chapter are based on simulations, you can apply the same general principles when determining the best termination scheme, drive strength setting, and loading style to any board designs. Even armed with this knowledge, it is still critical that you perform simulations, either using IBIS or HSPICE models, to determine the quality of signal integrity on your designs.

## Package Deskeew for QDR II and QDR-IV

You should follow Altera's package deskeew guidance if you are using Stratix V or Arria 10 devices.

For more information on package deskeew, refer to *Package Deskeew*.

## QDR-IV Layout Recommendations

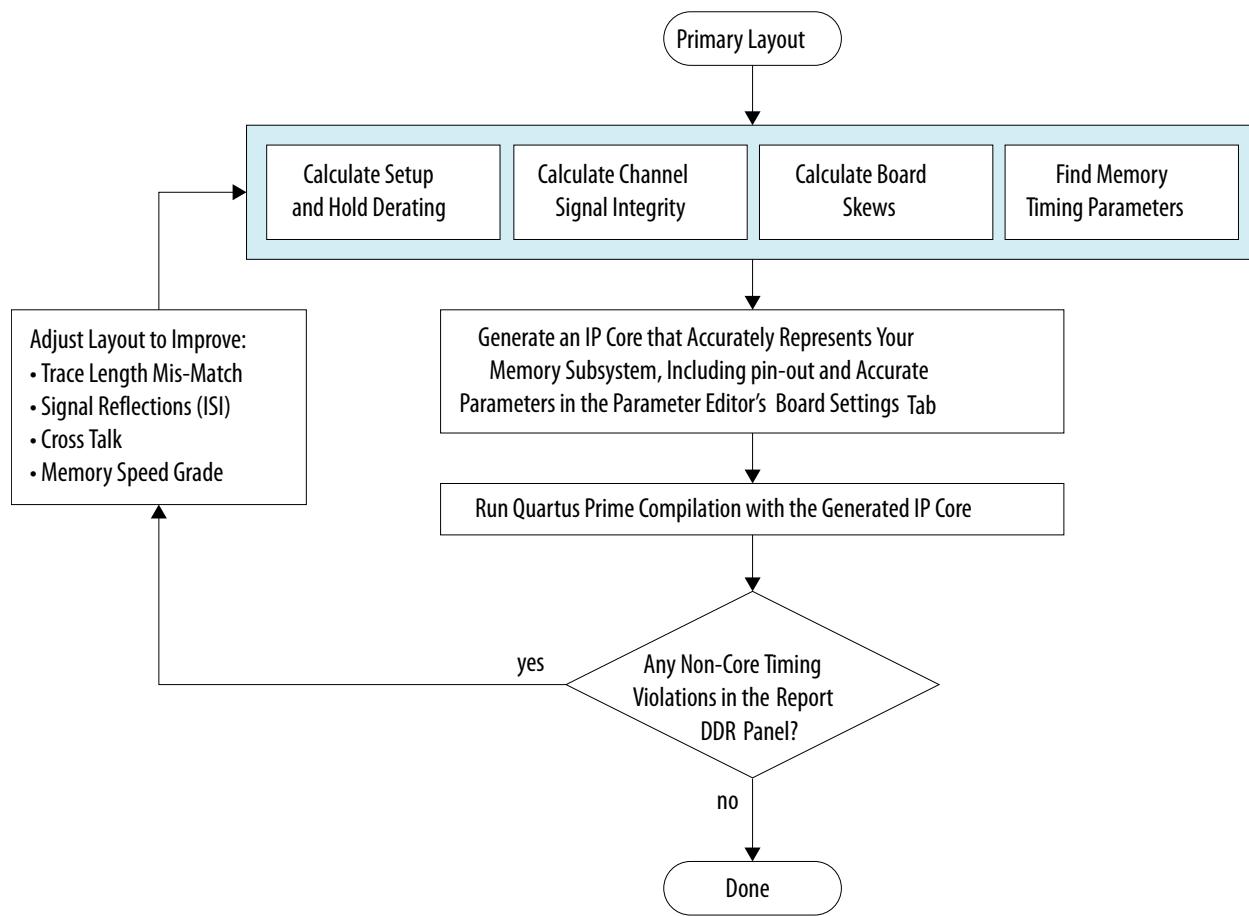
Altera recommends the following layout guidelines for QDR-IV on Arria 10 devices.

1. For port B only: Swap the polarity of the `QKB` and `QKB#` signals with respect to the polarity of the differential buffer inputs on the FPGA. Connect the positive leg of the differential input buffer on the FPGA to QDR-IV `QKB#` (negative) pin and vice-versa. Note that the port names at the top-level of the IP already reflect this swap (that is, `mem_qkb` is assigned to the negative buffer leg, and `mem_qkb_n` is assigned to the positive buffer leg).
2. For each port, set the `DK/DK#` or `QK/QK#` clock as the target trace propagation delay for the associated data signals (`DQ`).
3. For each port, route the data (`DQ`) signals for the `DQ` group ideally on the same layer as the associated `QK/QK#` and `DK/DK#` clocks to within  $\pm 10$  ps skew of the target clock.
4. Route the `mem_ck` (`CK/CK#`) clocks and set as the target trace propagation delays for the address/command signal group. Match the `CK/CK#` clock to within  $\pm 50$  ps of all the `DK/DK#` clocks for both ports.
5. Route the `address/control` signal group ideally on the same layer as the `mem_ck` (`CK/CK#`) clocks, to within  $\pm 10$  ps skew of the `mem_ck` (`CK/CK#`) traces.

## QDR-IV Layout Approach

For all practical purposes, you can regard the TimeQuest timing analyzer's report on your memory interface as definitive for a given set of memory and board timing parameters. You will find timing under Report DDR in TimeQuest and on the Timing Analysis tab in the parameter editor.

The following flowchart illustrates the recommended process to follow during the design phase, to determine timing margin and make iterative improvements to your design.



For more detailed simulation guidance for Arria 10, refer to the Altera wiki: [http://www.alterawiki.com/wiki/Arria\\_10\\_EMIF\\_Simulation\\_Guidance](http://www.alterawiki.com/wiki/Arria_10_EMIF_Simulation_Guidance)

### Intersymbol Interference/Crosstalk

For information on intersymbol interference and crosstalk, refer to the Altera wiki: [http://www.alterawiki.com/wiki/Arria\\_10\\_EMIF\\_Simulation\\_Guidance](http://www.alterawiki.com/wiki/Arria_10_EMIF_Simulation_Guidance)

### Board Skew

For information on calculating board skew parameters, refer to

If you know the absolute delays for all the memory related traces, the interactive **Board Skew Parameter Tool** can help you calculate the necessary parameters.

### Memory Timing Parameters

You can find the memory timing parameters to enter in the parameter editor, in your memory vendor's datasheet.

## QDR-IV Layout Guidelines

Observe the following layout guidelines for your QDR-IV interface.

Parameter	Guidelines
General Routing	<ul style="list-style-type: none"> <li>If you must route signals of the same net group on different layers with the same impedance characteristic, simulate your worst case PCB trace tolerances to determine actual propagation delay differences. Typical layer-to-layer trace delay variations are on the order of 15 ps/inch.</li> <li>Avoid T-junctions greater than 150 ps.</li> <li>Match all signals within a given DQ group with a maximum skew of <math>\pm 10</math> ps and route on the same layer.</li> </ul>
Clock Routing	<ul style="list-style-type: none"> <li>Route clocks on inner layers with outer-layer run lengths held to less than 150 ps.</li> <li>Clock signals should maintain a 10-mil (0.254 mm) spacing from other nets.</li> <li>Clocks should maintain a length-matching between clock pairs of <math>\pm 5</math> ps.</li> <li>Differential clocks should maintain a length-matching between P and N signals of <math>\pm 2</math> ps.</li> <li>Space between different clock pairs should be at least three times the space between the traces of a differential pair.</li> </ul>
Address and Command Routing	<ul style="list-style-type: none"> <li>- To minimize crosstalk, route address, bank address, and command signals on a different layer than the data signals.</li> <li>Do not route the differential clock signals close to the address signals.</li> <li>Keep the distance from the pin on the QDR-IV component to the stub termination resistor (VTT) to less than 50 ps for the address/command signal group.</li> <li>- Route the mem_ck (CK/CK#) clocks and set as the target trace propagation delays for the address/command signal group. Match the CK/CK# clock to within <math>\pm 50</math> ps of all the DK/DK# clocks for both ports.</li> <li>- Route the address/control signal group ideally on the same layer as the mem_ck (CK/CK#) clocks, to within <math>\pm 20</math> ps skew of the mem_ck (CK/CK#) traces.</li> </ul>
Data Signals	<ul style="list-style-type: none"> <li>For port B only: Swap the polarity of the QKB and QKB# signals with respect to the polarity of the differential buffer inputs on the FPGA. Connect the positive leg of the differential input buffer on the FPGA to QDR-IV QKB# (negative) pin and vice-versa. Note that the port names at the top-level of the IP already reflect this swap (that is, mem_qkb is assigned to the negative buffer leg, and mem_qkb_n is assigned to the positive buffer leg).</li> <li>For each port, route the DK/DK# write clock and QK/QK# read clock associated with a DQ group on the same PCB layer. Match these clock pairs to within <math>\pm 5</math> ps.</li> <li>For each port, set the DK/DK# or QK/QK# clock as the target trace propagation delay for the associated data signals (DQ).</li> <li>For each port, route the data (DQ) signals for the DQ group ideally on the same layer as the associated QK/QK# and DK/DK# clocks to within <math>\pm 10</math> ps skew of the target clock.</li> </ul>
Maximum Trace Length	<ul style="list-style-type: none"> <li>Keep the maximum trace length of all signals from the FPGA to the RLDRAM II or RLDRAM 3 components to 600 ps.</li> </ul>

Parameter	Guidelines
Spacing Guidelines	<ul style="list-style-type: none"><li>Avoid routing two signal layers next to each other. Always make sure that the signals related to memory interface are routed between appropriate GND or power layers.</li><li>For Data and Data Strobe traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace.</li><li>For Address/Command/Control traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace.</li><li>For Clock (<code>mem_CK</code>) traces: Maintain at least 5H spacing between two clock pair or a clock pair and any other memory interface trace, where H is the vertical distance to the closest return path for that particular trace.</li></ul>

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	In the first guideline of the <i>QDR-IV Layout Recommendations</i> and the <i>Data Signals</i> section of the <i>QDR-IV Layout Guidelines</i> , revised the information for <i>Port B only</i> .
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	<ul style="list-style-type: none"><li>Change to <i>K/K# Clocks</i> row in <i>Termination Recommendations for Arria V, Stratix III, Stratix IV, and Stratix V Devices</i> table.</li><li>Removed millimeter approximations from lengths expressed in picoseconds in <i>QDR II</i> and <i>QDR II+ SRAM Layout Guidelines</i> table.</li><li>Minor formatting fixes in <i>QDR II</i> and <i>QDR II+ SRAM Layout Guidelines</i> table.</li></ul>
December 2013	2013.12.16	Consolidated General Layout Guidelines.
November 2012	4.2	Changed chapter number from 7 to 8.
June 2012	4.1	Added Feedback icon.
November 2011	4.0	Added Arria V information.
June 2011	3.0	Added Stratix V information.
December 2010	2.0	Maintenance release.

Date	Version	Changes
July 2010	1.0	Initial release.

# Implementing and Parameterizing Memory IP

7

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The following topics describe the general overview of the Altera® IP core design flow to help you quickly get started with any Altera IP core.

The Altera IP Library is installed as part of the Quartus® Prime installation process. You can select and parameterize any Altera IP core from the library. Altera provides an integrated parameter editor that allows you to customize IP cores to support a wide variety of applications. The parameter editor guides you through the setting of parameter values and selection of optional ports. The following section describes the general design flow and use of Altera IP cores.

**Note:** Information for Arria 10 External Memory Interface IP also applies to Arria 10 External Memory Interface for HPS IP unless stated otherwise.

## Related Information

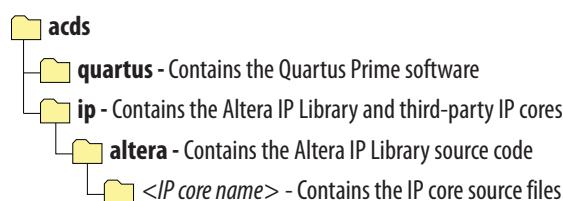
### Altera Design Store

Design Example for Arria 10 DDR3 External Memory Interface is available in the Altera Design Store.

## Licensing IP Cores

The Altera® IP Library provides many useful IP core functions for your production use without purchasing an additional license. Some Altera MegaCore® IP functions require that you purchase a separate license for production use. However, the OpenCore® feature allows evaluation of any Altera IP core in simulation and compilation in the software. After you are satisfied with functionality and performance, visit the Self Service Licensing Center to obtain a license number for any Altera product.

**Figure 7-1: IP Core Installation Path**



**Note:** The default IP installation directory on Windows is `<drive>:\altera\<version number>`; on Linux the IP installation directory is `<home directory>/altera/ <version number>`.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

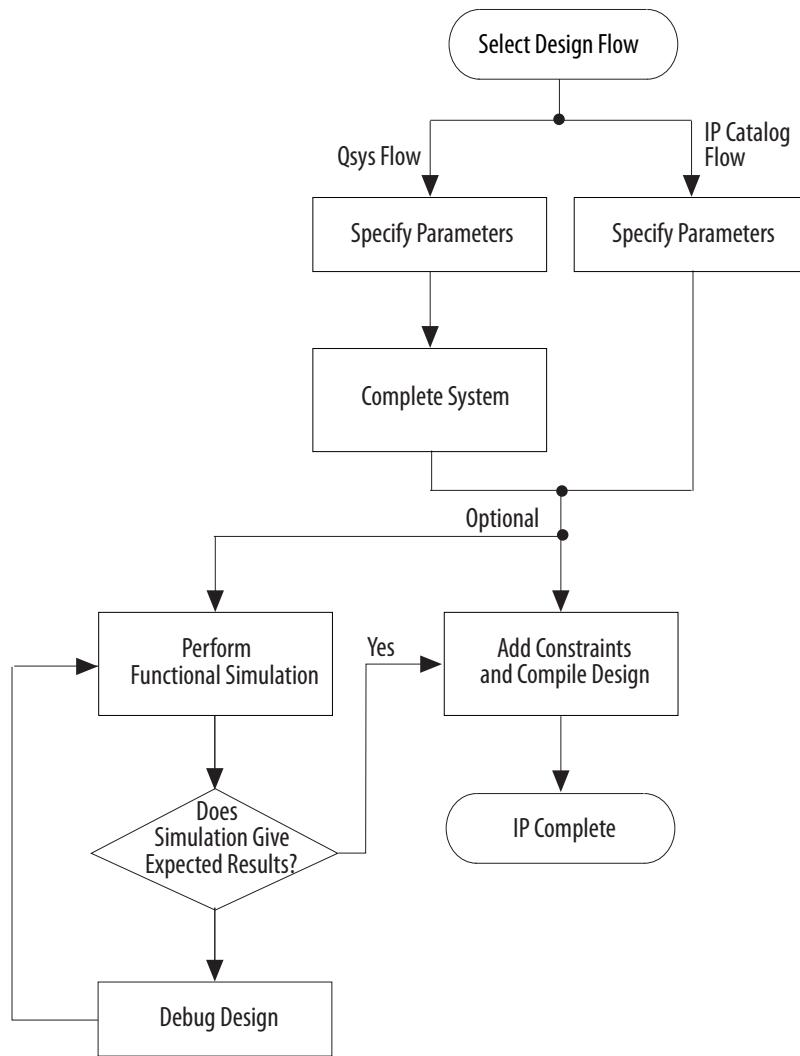
## Design Flow

You can implement the external memory interface IP using the following flows:

- IP Catalog flow
- Qsys flow

The following figure shows the stages for creating a system in the Quartus Prime software using the available flows.

**Figure 7-2: Design Flows**



Note to Figure:

The IP Catalog design flow is suited for simple designs where you want to manually instantiate the external memory interface IP into a larger component. The Qsys design flow is recommended for more complex system designs where you want the tool to manage the instantiation process.

## IP Catalog Design Flow

The IP Catalog design flow allows you to customize the external memory interface IP, and manually integrate the function into your design.

### IP Catalog and Parameter Editor

The IP Catalog displays the installed IP cores available for your design. Double-click any IP core to launch the parameter editor and generate files representing your IP variation. Use the following features to help you quickly locate and select an IP core:

- Filter IP Catalog to **Show IP for active device family** or **Show IP for all device families**. If you have no project open, select the **Device Family** in IP Catalog.
- Type in the Search field to locate any full or partial IP core name in IP Catalog.
- Right-click an IP core name in IP Catalog to display details about supported devices, open the IP core's installation folder, and click links to IP documentation.
- Click **Search for Partner IP**, to access partner IP information on the Altera website.

The parameter editor prompts you to specify an IP variation name, optional ports, and output file generation options. The parameter editor generates a top-level Qsys system file (.qsys) or IP file (.qip) representing the IP core in your project. You can also parameterize an IP variation without an open project.

The IP Catalog is also available in Qsys (**View > IP Catalog**). The Qsys IP Catalog includes exclusive system interconnect, video and image processing, and other system-level IP that are not available in the IP Catalog. For more information about using the Qsys IP Catalog, refer to *Creating a System with Qsys* in Volume 1 of the *Handbook*.

#### Related Information

[Creating a System with Qsys](#)

### Specifying Parameters for the IP Catalog Flow

To specify parameters with the IP Catalog design flow, perform the following steps:

1. In the Quartus Prime software, create a Quartus Prime project using the **New Project Wizard** available from the File menu.
2. Launch the **IP Catalog** from the **Tools** menu.
3. Select an external memory interface IP from the **Memory Interfaces and Controllers** folder in the **Library** list.

**Note:** The availability of external memory interface IP depends on the device family your design is using.

4. Depending on the window which appears, proceed as follows:
  - New IP Instance Window: Specify the Top-level Name and Device Settings, and click **Ok**.
  - Save IP Variation window: Specify the IP variation file name and IP variation file type, and click **Ok**.
5. In the **Presets** window, select the preset matching your design requirement, and click **Apply**.

**Tip:** If none of the presets match your design requirements, you can apply the closest preset available and then change the parameters manually. This method may be faster than entering all the parameters manually, and reduces the chance of having incorrect settings.

6. Specify the parameters on all tabs.

- Note:**
- For detailed explanation of the parameters, refer to *Parameterizing Memory Controllers with UniPHY IP* and *Parameterizing Memory Controllers with Arria 10 External Memory Interface IP*.
  - Although you have applied presets, you may need to modify some of the preset parameters depending on the frequency of operation. A typical list of parameters which you might need to change includes the Memory CAS Latency setting, the Memory CAS Write Latency setting, and the tWTR, tFAW, tRRD, and tRTP settings.

- Tip:**
- As a good practice, review any warning messages displayed in the **Messages Window** and correct any errors before making further changes.
  - To simplify future work, you might want to store the current configuration by saving your own presets. To create, modify, or remove your own custom presets, click **New**, **Update**, or **Delete** at the bottom of the **Presets** list.
  - If you want to generate an example design for your current configuration, click **Example Design** at the top-right corner of the parameter editor, specify a path for the example design, and click **Ok**.

7. Depending on which external memory interface IP is selected, perform the following steps to complete the IP generation:

- For Arria 10 External Memory Interface IP:
  - Click **Finish**. Your configuration is saved as a **.qsys** file.
  - Click **Yes** when you are prompted to generate your IP.
  - Set **Create HDL design files for synthesis** to **Verilog** or **VHDL**.

**Tip:** If you want to do RTL simulation of your design, you should set **Create simulation model** to either **Verilog** or **VHDL**. Some RTL simulation-related files, including simulator-specific scripts, are generated only if you specify this parameter.

**Note:** For Arria 10 External Memory Interface IP, the synthesis and simulation model files are identical. However, there are some differences in file types when generating for VHDL. For synthesis files, only the top-level wrapper is generated in VHDL; the other files are generated in System Verilog. For simulation files, all the files are generated as a Mentor-tagged encrypted IP for VHDL-only simulator support.

- Click **Generate**.
  - When generation has completed, click **Finish**.
- For UniPHY-based IP:

- Click the **Finish** button.

**Note:** The **Finish** button may be unavailable until you have corrected all parameterization errors listed in the **Messages** window.

- If prompted, specify whether you want to generate an example design by checking or unchecking **Generate Example Design**, and then click **Generate**.

**Caution:** If you have already generated an example design, uncheck **Generate Example Design** to prevent your previously generated files from being overwritten.

- When generation is completed, click **Exit**.
- Click **Yes** if you are prompted to add the **.qip** to the current Quartus Prime project. You can also turn on **Automatically add Quartus Prime IP Files to all projects**.

**Tip:** Always read the generated **readme.txt** file, which contains information and guidelines specific to your configuration.

- You can now integrate your custom IP core instance in your design, simulate, and compile. While integrating your IP core instance into your design, you must make appropriate pin assignments. You

can create a virtual pin to avoid making specific pin assignments for top-level signals while you are simulating and not ready to map the design to hardware.

**Note:** For information about the Quartus Prime software, including virtual pins and the IP Catalog and Qsys, refer to Quartus Prime Help.

#### Related Information

- [Simulating Altera Designs](#)
- [Quartus Prime Help](#)

## Using Example Designs

When you generate your IP, you can instruct the system to produce an example design consisting of an external memory interface IP of your configuration, together with a traffic generator.

For synthesis, the example design includes a project for which you can specify pin locations and a target device, compile in the Quartus Prime software, verify timing closure, and test on your board using the programming file generated by the Quartus Prime assembler. For simulation, the example design includes an example memory model with which you can run simulation and evaluate the result.

For a UniPHY-based external memory interface, click **Example Design** in the parameter editor, or enable **Generate Example Design**. The system produces an example design for synthesis in the **example\_project** directory, and generation scripts for simulation in the **simulation** directory. To generate the complete example design for RTL simulation, follow the instructions in the **readme.txt** file in the **simulation** directory.

For Arria 10 External Memory Interface IP, click **Example Design** in the parameter editor. The system produces generation scripts in the directory path that you specify. To create a complete example design for synthesis or RTL simulation, follow the instructions in the generated **<variation\_name>/altera\_emif\_arch\_nf\_140/<synth|sim>/<variation\_name>\_altera\_emif\_arch\_nf\_140\_<unique ID>\_readme.txt** file.

To compile an example design, open the **.qpf** file for the project and follow the standard design flow, including constraining the design prior to full compilation. If necessary, change the example project device to match the device in your project.

For more information about example designs, refer to *Functional Description—Example Top Level project* in Volume 3 of the *External Memory Interface Handbook*. For more information about simulating an example design, refer to *Simulating the Example Design* in the *Simulating Memory IP* chapter.

## Constraining the Design

For Arria 10 External Memory Interface IP for HPS, pin location assignments are predefined in the Quartus Prime IP file (**.qip**). In UniPHY-based and non-HPS Arria 10 external memory interfaces, you must make your own location assignments.

**Note:** You should not overconstrain any EMIF IP-related registers unless you are advised to do so by Altera, or you fully understand the effect on the external memory interface operation. Also, ensure that any wildcards in your user logic do not accidentally target EMIF IP-related registers.

For more information about timing constraints and analysis, refer to *Analyzing Timing of Memory IP*.

#### Related Information

- [Analyzing Timing of Memory IP](#) on page 9-1

## Adding Pins and DQ Group Assignments

The assignments defined in the **<variation\_name>\_pin\_assignments.tcl** script (for UniPHY-based IP) and the Quartus Prime IP file (**.qip**) (for Arria 10 EMIF IP) help you to set up the I/O standards and the input/

output termination for the external memory interface IP. These assignments also help to relate the DQ pin groups together for the Quartus Prime Fitter to place them correctly.

- For UniPHY-based external memory interfaces, run the `<variation_name>_pin_assignments.tcl` script to apply the input and output termination, I/O standards, and DQ group assignments to your design. To run the pin assignment script, follow these steps:
  - On the Processing menu, point to **Start**, and click **Start Analysis and Synthesis**. Allow **Analysis and Synthesis** to finish without errors before proceeding to step 2.
  - On the Tools menu click **Tcl Scripts**.
  - Specify the `pin_assignments.tcl` and click **Run**.

The pin assignment script does not create a PLL reference clock for the design. You must create a clock for the design and provide pin assignments for the signals of both the example driver and testbench that the MegaCore variation generates.

**Note:** For some UniPHY-based IP configurations, the `afi_clk` clock does not have a global signal assignment constraint. In this case, you should add a suitable assignment for your design. For example, for a UniPHY-based DDR3 IP targeting a Stratix IV device, `if0|p110|up11_memphy|auto_generated|clk[0]` does not have a global signal assignment and you should consider adding either a global clock or a dual regional clock assignment to your project for this clock.

- For Arria 10 External Memory Interface IP, the Quartus Prime software automatically reads assignments from the `.qip` file during compilation, so it is not necessary to apply assignments to your design manually.

- Note:**
- If you must overwrite the default assignments, ensure that you make your changes in the Quartus Prime Settings File (`.qsf`) and not the `.qip` file. Assignments in the `.qsf` file take precedence over assignments in the `.qip` file. Note also, that if you rerun the `<variation_name>_pin_assignments.tcl` file, it overwrites your changes.
  - If the PLL input reference clock pin does not have the same I/O standard as the memory interface I/Os, a no-fit might occur because incompatible I/O standards cannot be placed in the same I/O bank.
  - If you are upgrading your memory IP from an earlier Quartus Prime version, rerun the `pin_assignments.tcl` script in the later Quartus Prime revision.
  - If you encounter a shortage of clock resources, the AFI clock domain can be moved between regional, dual-regional, and global. Moving any other clock domain can result in fit errors or timing closure problems.

## Compiling the Design

After constraining your design, compile your design in the Quartus Prime software to generate timing reports to verify whether timing has been met.

To compile the design, on the Processing menu, click **Start Compilation**.

After you have compiled the top-level file, you can perform RTL simulation or program your targeted Altera device to verify the top-level file in hardware.

**Note:** In UniPHY-based memory controllers, the `derive_pll_clocks` command can affect timing closure if it is called before the memory controller files are loaded. Ensure that the Quartus Prime IP File (`.qip`) appears in the file list before any Synopsys Design Constraint Files (`.sdc`) files that contain `derive_pll_clocks`.

For more information about simulating the memory IP, refer to *Simulating Memory IP*.

#### Related Information

[Simulating Memory IP](#) on page 8-1

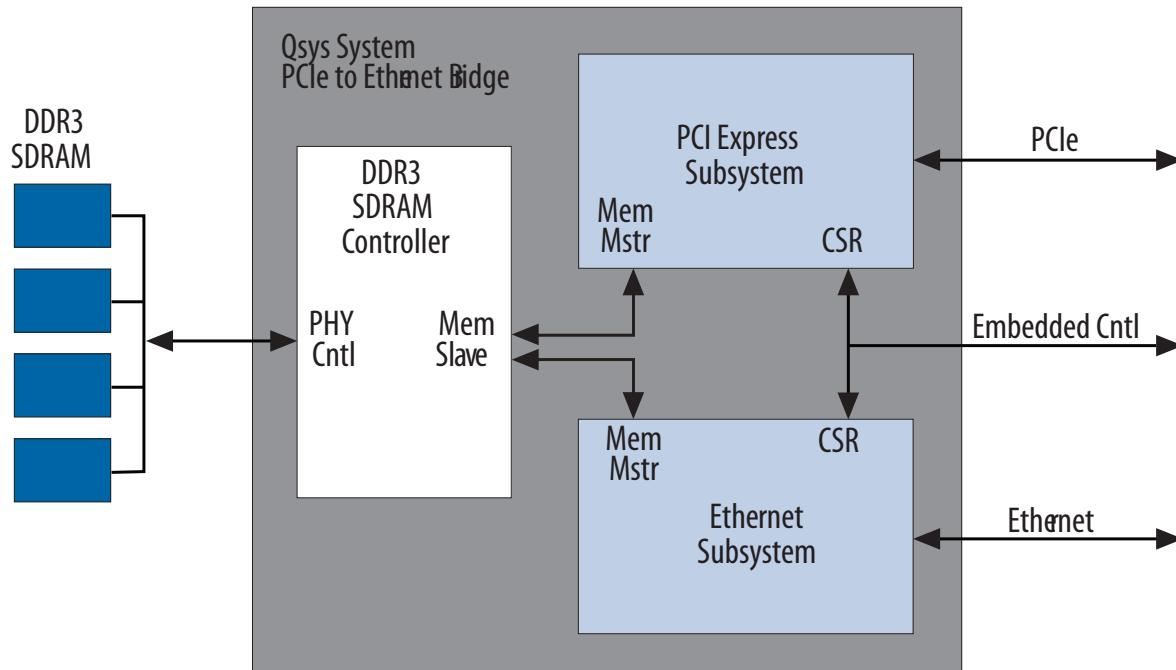
## Qsys System Integration Tool Design Flow

You can use the Qsys system integration tool to build a system that includes your customized IP core.

You easily can add other components and quickly create a Qsys system. Qsys automatically generates HDL files that include all of the specified components and interconnections. In Qsys, you specify the connections you want. The HDL files are ready to be compiled by the Quartus Prime software to produce output files for programming an Altera device. Qsys generates Verilog HDL simulation models for the IP cores that comprise your system.

The following figure shows a high level block diagram of an example Qsys system.

**Figure 7-3: Example Qsys System**



For more information about the Qsys system interconnect, refer to the *Qsys Interconnect* chapter in volume 1 of the *Quartus Prime Handbook* and to the *Avalon Interface Specifications*.

For more information about the Qsys tool and the Quartus Prime software, refer to the *System Design with Qsys* section in volume 1 of the *Quartus Prime Handbook* and to Quartus Prime Help.

#### Related Information

- [Qsys Interconnect](#)
- [Avalon Interface Specifications](#)
- [System Design with Qsys](#)

## Specify Parameters for the Qsys Flow

To specify parameters for your IP core using the Qsys flow, follow these steps:

1. In the Quartus Prime software, create a new Quartus Prime project using the **New Project Wizard** available from the **File** menu.
2. On the Tools menu, click **Qsys**.

**Note:** Qsys automatically sets device parameters based on your Quartus Prime project settings. To set device parameters manually, use the **Device Family** tab.

3. In the **IP Catalog**, select the available external memory interface IP from the **Memory Interfaces and Controllers** folder in the **Library** list. (For Arria 10 EMIF for HPS, select the external memory interface IP from the **Hard Processor Components** folder.) The relevant parameter editor appears.

**Note:** The availability of external memory interface IP depends on the device family your design is using. To use Arria 10 External Memory Interface for HPS IP, your design must target a device containing at least one HPS CPU core.

4. From the **Presets** list, select the preset matching your design requirement, and click **Apply**.

**Tip:** If none of the presets match your design requirements, you can apply the closest preset available and then change the inappropriate parameters manually. This method may be faster than entering all the parameters manually, and reduces the chance of having incorrect settings.

5. Specify the parameters on all tabs.

**Note:**

- For detailed explanation of the parameters, refer to *Parameterizing Memory Controllers with UniPHY IP* and *Parameterizing Memory Controllers with Arria 10 External Memory Interface IP*.
- Although you have applied presets, you may need to modify some of the preset parameters depending on the frequency of operation. A typical list of parameters which you might need to change includes the Memory CAS Latency setting, the Memory CAS Write Latency setting, and the tWTR, tFAW, tRRD, and tRTP settings.
- For UniPHY-based IP, turn on **Generate power-of-2 bus widths for Qsys or SOPC Builder** on the **Controller Settings** tab.

**Tip:**

- As a good practice, review any warning messages displayed in the **Messages Window** and correct any errors before making further changes.
- To simplify future work, you might want to store the current configuration by saving your own presets. To create, modify, or remove your own custom presets, click **New**, **Update**, or **Delete** at the bottom of the **Presets** list.
- If you want to generate an example design for your current configuration, click **Example Design** at the top-right corner of the parameter editor, specify a path for the example design, and click **Ok**.

6. Click **Finish** to complete the external memory interface IP instance and add it to the system.

**Note:** The **Finish** button may be unavailable until you have corrected all parameterization errors listed in the **Messages** window.

## Completing the Qsys System

To complete the Qsys system, follow these steps:

1. Add and parameterize any additional components.
2. Connect the components using the Connection panel on the **System Contents** tab.
3. In the **Export** column, enter the name of any connections that should be a top-level Qsys system port.

**Note:** Ensure that the memory and oct interfaces are exported to the top-level Qsys system port. If these interfaces are already exported, take care not to accidentally rename or delete either of them in the **Export** column of the **System Contents** tab.

4. Click **Finish**.
5. Specify the **File Name** and click **Save**.
6. When you are prompted to generate now, click **Yes**.
7. Set **Create HDL design files for synthesis** to either Verilog or VHDL.

**Tip:** If you want to do RTL simulation of your design, you should set **Create simulation model** to either **Verilog** or **VHDL**. Some RTL simulation-related files, including simulator-specific scripts, are generated only if you specify this parameter.

**Note:** For Arria 10 External Memory Interface IP, the synthesis and simulation model files are identical. However, there are some differences in file types when generating for VHDL. For synthesis files, only the top-level wrapper is generated in VHDL; the other files are generated in System Verilog. For simulation files, all the files are generated as a Mentor-tagged encrypted IP for VHDL-only simulator support.

8. Click **Generate**.
9. When generation has completed, click **Finish**.
10. If you are prompted to add the **.qip** file to the current Quartus Prime project, click **Yes** (If you want, you can turn on **Automatically Add Quartus Prime IP Files to all projects**).

**Tip:** Always read the generated **readme.txt** file, because it contains information and guidelines specific to your configuration.

You can now simulate and compile your design. But before compilation, you must make appropriate pin assignments. You can create a virtual pin to avoid making specific pin assignments for top-level signals during simulation and not yet ready to map the design to hardware.

For information about the Quartus Prime software, including virtual pins and the IP Catalog and Qsys, refer to the Quartus Prime Help.

## UniPHY-Based External Memory Interface IP

This section contains information about parameterizing UniPHY-based external memory interfaces.

### Qsys Interfaces

The following tables list the signals available for each interface in Qsys, and provide a description and guidance on connecting those interfaces.

### DDR2 SDRAM Controller with UniPHY Interfaces

The following table lists the DDR2 SDRAM with UniPHY signals available for each interface in Qsys and provides a description and guidance on how to connect those interfaces.

**Table 7-1: DDR2 SDRAM Controller with UniPHY Interfaces**

Signals in Interface	Interface Type	Description/How to Connect
<b>pll_ref_clk interface</b>		
pll_ref_clk	Clock input	PLL reference clock input.

Signals in Interface	Interface Type	Description/How to Connect
<b>global_reset interface</b>		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
<b>soft_reset interface</b>		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
<b>afi_reset interface</b>		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. The controller asserts this interface when the PLL loses lock or the PHY is reset.
<b>afi_reset_export interface</b>		
afi_reset_export_n	Reset output (PLL master/no sharing)	This interface is a copy of the afi_reset interface. It is intended to be connected to PLL sharing slaves.
<b>afi_reset_in interface</b>		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the afi_reset_export_n output of an identically configured memory interface in PLL master mode.
<b>afi_clk interface</b>		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_clk_in interface</b>		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, you must connect this afi_clk input to the afi_clk output of an identically configured memory interface in PLL master mode.
<b>afi_half_clk interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL master or no sharing modes, this interface is a clock output.

**afi\_half\_clk\_in interface**

afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL slave mode, this is a clock input that you must connect to the afi_half_clk output of an identically configured memory interface in PLL master mode.
--------------	-------------------------	---

**memory interface (DDR2 SDRAM)**

mem_a	Conduit	Interface signals between the PHY and the memory device.
mem_ba		
mem_ck		
mem_ck_n		
mem_cke		
mem_cs_n		
mem_dm		
mem_ras_n		
mem_cas_n		
mem_we_n		
mem_dq		
mem_dqs		
mem_dqs_n		
mem_odt		
mem_ac_parity		
mem_err_out_n		
mem_parity_error_n		

Signals in Interface	Interface Type	Description/How to Connect
<b>memory interface (LPDDR2)</b>		
mem_ca	Conduit	Interface signals between the PHY and the memory device.
mem_ck		
mem_ck_n		
mem_cke		
mem_cs_n		
mem_dm		
mem_dq		
mem_dqs		
mem_dqs_n		
<b>avl interface</b>		
avl_ready	Avalon-MM Slave	Avalon-MM interface signals between the memory interface and user logic.
avl_burst_begin		
avl_addr		
avl_rdata_valid		
avl_rdata		
avl_wdata		
avl_be		
avl_read_req		
avl_write_req		
avl_size		
<b>status interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
local_init_done	Conduit	
local_cal_success		Memory interface status signals.
local_cal_fail		
<b>oct interface</b>		
rup (Stratix® III/IV, Arria® II GZ)	Conduit	
rdn (Stratix III/IV, Arria II GZ)		OCT reference resistor pins for rup/rdn or rzqin.
rzq (Stratix V, Arria V, Cyclone V)		
<b>local_powerdown interface</b>		
local_powerdn_ack	Conduit	This powerdown interface for the controller is enabled only when you turn on <b>Enable Auto Powerdown</b> .
<b>pll_sharing interface</b>		
pll_mem_clk	Conduit	
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set <b>PLL sharing mode</b> to master or slave.
pll_config_clk		
pll_hr_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
pll_dr_clk		
<b>dll_sharing interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set <b>DLL sharing mode</b> to master or slave.
dll_pll_locked		
<b>oct_sharing interface</b>		
serieterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set <b>OCT sharing mode</b> to master or slave.
parallelterminationcontrol		
<b>autoprecharge_req interface</b>		
local_autopch_req	Conduit	Precharge interface for connection to a custom control block. This interface is enabled only when you turn on <b>Auto precharge Control</b> .
<b>user_refresh interface</b>		
local_refresh_req	Conduit	User refresh interface for connection to a custom control block. This interface is enabled only when you turn on <b>User Auto-Refresh Control</b> .
local_refresh_chip		
local_refresh_ack		
<b>self_refresh interface</b>		
local_self_rfsh_req	Conduit	Self refresh interface for connection to a custom control block. This interface is enabled only when you turn on <b>Self-refresh Control</b> .
local_self_rfsh_chip		
local_self_rfsh_ack		
<b>ecc_interrupt interface</b>		
ecc_interrupt	Conduit	ECC interrupt signal for connection to a custom control block. This interface is enabled only when you turn on <b>Error Detection and Correction Logic</b> .
<b>csr interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
csr_write_req	Avalon-MM Slave	Configuration and status register signals for the memory interface, for connection to an Avalon_MM master. This interface is enabled only when you turn on <b>Configuration and Status Register</b> .
csr_read_req		
csr_waitrequest		
csr_addr		
csr_be		
csr_wdata		
csr_rdata		
csr_rdata_valid		

### Hard Memory Controller MPFE FIFO Clock Interface

mp_cmd_clk	Conduit	When you enable the Hard Memory Interface, three FIFO buffers (command, read data, and write data) are created in the MPFE. Each FIFO buffer has its own clock and reset port.  This interface is enabled when you turn on the Enable Hard Memory Interface.
mp_rfifo_clk		
mp_wfifo_clk		
mp_cmd_reset		
mp_rfifo_reset		
mp_wfifo_reset		

### Hard Memory Controller Bonding Interface

bonding_in_1	Conduit	Bonding interface to bond two controllers to expand the bandwidth. This interface is enabled when you turn on the Export bonding interface.
bonding_in_2		
bonding_in_3		
bonding_out_1		
bonding_out_2		
bonding_out_3		

Note to Table:

1. Signals available only in DLL master mode.

## DDR3 SDRAM Controller with UniPHY Interfaces

The following table lists the DDR3 SDRAM with UniPHY signals available for each interface in Qsys and provides a description and guidance on how to connect those interfaces.

**Table 7-2: DDR3 SDRAM Controller with UniPHY Interfaces**

Signals in Interface	Interface Type	Description/How to Connect
<b>pll_ref_clk interface</b>		
pll_ref_clk	Clock input	PLL reference clock input.
<b>global_reset interface</b>		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
<b>soft_reset interface</b>		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
<b>afi_reset interface</b>		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. This interface is asserted when the PLL loses lock or the PHY is reset.
<b>afi_reset_export interface</b>		
afi_reset_export_n	Reset output (PLL master/no sharing)	This interface is a copy of the afi_reset interface. It is intended to be connected to PLL sharing slaves.
<b>afi_reset_in interface</b>		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the afi_reset_export_n output of an identically configured memory interface in PLL master mode.
<b>afi_clk interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_clk_in interface</b>		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, this is a clock input that you must connect to the <code>afi_clk</code> output of an identically configured memory interface in PLL master mode.
<b>afi_half_clk interface</b>		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of <code>afi_clk</code> . When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_half_clk_in interface</b>		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of the <code>afi_clk</code> . When the interface is in PLL slave mode, you must connect this <code>afi_half_clk</code> input to the <code>afi_half_clk</code> output of an identically configured memory interface in PLL master mode.
<b>memory interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
mem_a	Conduit	
mem_ba		
mem_ck		
mem_ck_n		
mem_cke		
mem_cs_n		
mem_dm		
mem_ras_n		
mem_cas_n		
mem_we_n		
mem_dq		
mem_dqs		
mem_dqs_n		
mem_odt		
mem_reset_n		
mem_ac_parity		
mem_err_out_n		
mem_parity_error_n		
<b>avl interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
avl_ready	Avalon-MM Slave	Avalon-MM interface signals between the memory interface and user logic.
avl_burst_begin		
avl_addr		
avl_rdata_valid		
avl_rdata		
avl_wdata		
avl_be		
avl_read_req		
avl_write_req		
avl_size		
<b>status interface</b>		
local_init_done	Conduit	Memory interface status signals.
local_cal_success		
local_cal_fail		
<b>oct interface</b>		
rup (Stratix III/IV, Arria II GZ)	Conduit	OCT reference resistor pins for rup/rdn or rzqin.
rdn (Stratix III/IV, Arria II GZ)		
rzq (Stratix V, Arria v, Cyclone V)		
<b>local_powerdown interface</b>		
local_powerdn_ack	Conduit	This powerdown interface for the controller is enabled only when you turn on <b>Enable Auto Power Down</b> .
<b>pll_sharing interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set <b>PLL sharing mode</b> to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_hr_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
pll_dr_clk		
<b>dll_sharing interface</b>		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set <b>DLL sharing mode</b> to master or slave.
dll_pll_locked		
<b>oct_sharing interface</b>		
serieterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set <b>OCT sharing mode</b> to master or slave.
parallelterminationcontrol		
<b>autoprecharge_req interface</b>		
local_autopch_req	Conduit	Precharge interface for connection to a custom control block. This interface is enabled only when you turn on <b>Auto-precharge Control</b> .
<b>user_refresh interface</b>		
local_refresh_req	Conduit	User refresh interface for connection to a custom control block. This interface is enabled only when you turn on <b>User Auto Refresh Control</b> .
local_refresh_chip		
local_refresh_ack		

Signals in Interface	Interface Type	Description/How to Connect
<b>self_refresh interface</b>		
local_self_rfsh_req		
local_self_rfsh_chip	Conduit	Self refresh interface for connection to a custom control block. This interface is enabled only when you turn on <b>Self-refresh Control</b> .
local_self_rfsh_ack		
<b>ecc_interrupt interface</b>		
ecc_interrupt	Conduit	ECC interrupt signal for connection to a custom control block. This interface is enabled only when you turn on <b>Error Detection and Correction Logic</b> .
<b>csr interface</b>		
csr_write_req		
csr_read_req		
csr_waitrequest		
csr_addr	Avalon-MM Slave	Configuration and status register signals for the memory interface, for connection to an Avalon_MM master. This interface is enabled only when you turn on <b>Configuration and Status Register</b> .
csr_be		
csr_wdata		
csr_rdata		
csr_rdata_valid		
<b>Hard Memory Controller MPFE FIFO Clock Interface</b>		
mp_cmd_clk		
mp_rfifo_clk		
mp_wfifo_clk		
mp_cmd_reset_n	Conduit	When you enable the Hard Memory Interface, three FIFO buffers (command, read data, and write data) are created in the MPFE. Each FIFO buffer has its own clock and reset port.
mp_rfifo_reset_n		This interface is enabled when you turn on the Enable Hard Memory Interface.
mp_wfifo_reset_n		

Signals in Interface	Interface Type	Description/How to Connect
<b>Hard Memory Controller Bonding Interface</b>		
bonding_in_1	Conduit	Use bonding interface to bond two controllers to expand the bandwidth. This interface is enabled when you turn on the Export bonding interface.
bonding_in_2		
bonding_in_3		
bonding_out_1		
bonding_out_2		
bonding_out_3		

Note to Table:

1. Signals available only in DLL master mode.

## LPDDR2 SDRAM Controller with UniPHY Interfaces

The following table lists the LPDDR2 SDRAM signals available for each interface in Qsys and provides a description and guidance on how to connect those interfaces.

**Table 7-3: LPDDR2 SDRAM Controller with UniPHY Interfaces**

Signals in Interface	Interface Type	Description/How to Connect
<b>pll_ref_clk interface</b>		
pll_ref_clk	Clock input	PLL reference clock input.
<b>global_reset interface</b>		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
<b>soft_reset interface</b>		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
<b>afi_reset interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. The controller asserts this interface when the PLL loses lock or the PHY is reset.
<b>afi_reset_export interface</b>		
afi_reset_export_n	Reset output (PLL master/no sharing)	This interface is a copy of the afi_reset interface. It is intended to be connected to PLL sharing slaves.
<b>afi_reset_in interface</b>		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the afi_reset_export_n output of an identically configured memory interface in PLL master mode.
<b>afi_clk interface</b>		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_clk_in interface</b>		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, you must connect this afi_clk input to the afi_clk output of an identically configured memory interface in PLL master mode.
<b>afi_half_clk interface</b>		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL master or no sharing modes, this interface is a clock output.

Signals in Interface	Interface Type	Description/How to Connect
<b>afi_half_clk_in interface</b>		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL slave mode, this is a clock input that you must connect to the afi_half_clk output of an identically configured memory interface in PLL master mode.
<b>Memory interface</b>		
mem_ca	Conduit	Interface signals between the PHY and the memory device.
mem_ck		
mem_ck_n		
mem_cke		
mem_cs_n		
mem_dm		
mem_dq		
mem_dqs		
mem_dqs_n		
<b>avl interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
avl_ready	Avalon-MM Slave	Avalon-MM interface signals between the memory interface and user logic.
avl_burst_begin		
avl_addr		
avl_rdata_valid		
avl_rdata		
avl_wdata		
avl_be		
avl_read_req		
avl_write_req		
avl_size		
<b>status interface</b>		
local_init_done	Conduit	Memory interface status signals.
local_cal_success		
local_cal_fail		
<b>oct interface</b>		
rzq	Conduit	OCT reference resistor pins for rzqin.
<b>local_powerdown interface</b>		
local_powerdn_ack	Conduit	This powerdown interface for the controller is enabled only when you turn on Enable Auto Powerdown.
<b>local_deep_powerdn interface</b>		
local_deep_powerdn_ack	Conduit	Deep power down interface for the controller to enable deep power down. This interface is enable when turn on Enable Deep Power-Down Controls.
local_deep_powerdn_chip		
local_deep_powerdn_req		
<b>pll_sharing interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set PLL sharing mode to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_mem_phy_clk		
afi_phy_clk		
pll_write_clk_pre_phy_clk		
<b>dll_sharing interface</b>		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set DLL sharing mode to master or slave.
dll_pll_locked		
<b>oct_sharing interface</b>		
seriesterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set OCT sharing mode to master or slave.
parallelterminationcontrol		
<b>autoprecharge_req interface</b>		
local_autopch_req	Conduit	Precharge interface for connection to a custom control block. This interface is enabled only when you turn on Auto-precharge Control.
<b>user_refresh interface</b>		
local_refresh_req	Conduit	User refresh interface for connection to a custom control block. This interface is enabled only when you turn on User Auto-Refresh Control.
local_refresh_chip		
local_refresh_ack		
<b>self_refresh interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
local_self_rfsh_req	Conduit	Self refresh interface for connection to a custom control block. This interface is enabled only when you turn on Self-refresh Control.
local_self_rfsh_chip		
local_self_rfsh_ack		
<b>ecc_interrupt interface</b>		
ecc_interrupt	Conduit	ECC interrupt signal for connection to a custom control block. This interface is enabled only when you turn on Error Detection and Correction Logic.
<b>csr interface</b>		
csr_write_req	Avalon-MM Slave	Configuration and status register signals for the memory interface, for connection to an Avalon_MM master. This interface is enabled only when you turn on Configuration and Status Register.
csr_read_req		
csr_waitrequest		
csr_addr		
csr_be		
csr_wdata		
csr_rdata		
csr_rdata_valid		
<b>Local_rdata_error interface</b>		
Local_rdata_error	Conduit	Indicates read data error when Error Detection and Correction logic is enabled.
<b>Hard Memory Controller MPFE FIFO Clock Interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
mp_cmd_clk	Conduit	
mp_rfifo_clk		When you enable the Hard Memory Interface, three FIFO buffers (command, read data, and write data) are created in the MPFE. Each FIFO buffer has its own clock and reset port.
mp_wfifo_clk		
mp_cmd_reset_n		This interface is enabled when you turn on the Enable Hard Memory Interface.
mp_rfifo_reset_n		
mp_wfifo_reset_n		

### Hard Memory Controller Bonding Interface

bonding_in_1	Conduit	
bonding_in_2		
bonding_in_3		
bonding_out_1		Bonding interface to bond two controllers to expand the bandwidth. This interface is enabled when you turn on the Export bonding interface.
bonding_out_2		
bonding_out_3		

## QDR II and QDR II+ SRAM Controller with UniPHY Interfaces

The following table lists the QDR II and QDR II+ SRAM signals available for each interface in Qsys and provides a description and guidance on how to connect those interfaces.

**Table 7-4: QDR II and QDR II+ SRAM Controller with UniPHY Interfaces**

Signals in Interface	Interface Type	Description/How to Connect
<b>pll_ref_clk interface</b>		
pll_ref_clk	Clock input	PLL reference clock input.
<b>global_reset interface</b>		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
<b>soft_reset interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
<b>afi_reset interface</b>		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. This interface is asserted when the PLL loses lock or the PHY is reset.
<b>afi_reset_export interface</b>		
afi_reset_export_n	Reset output (PLL master/no sharing)	This interface is a copy of the afi_reset interface. It is intended to be connected to PLL sharing slaves.
<b>afi_reset_in interface</b>		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the afi_reset_export_n output of an identically configured memory interface in PLL master mode.
<b>afi_clk interface</b>		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_clk_in interface</b>		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, this is a clock input that you must connect to the afi_clk output of an identically configured memory interface in PLL master mode.
<b>afi_half_clk interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_half_clk_in interface</b>		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL slave mode, you must connect this afi_half_clk input to the afi_half_clk output of an identically configured memory interface in PLL master mode.
<b>memory interface</b>		
mem_a	Conduit	Interface signals between the PHY and the memory device.  The sequencer holds mem_doff_n low during initialization to ensure that internal PLL and DLL circuits in the memory device do not lock until clock signals have stabilized.
mem_cqn		
mem_bws_n		
mem_cq		
mem_d		
mem_k		
mem_k_n		
mem_q		
mem_wps_n		
mem_rps_n		
mem_doff_n		
<b>avl_r interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
avl_r_read_req	Avalon-MM Slave	Avalon-MM interface between memory interface and user logic for read requests.
avl_r_ready		
avl_r_addr		
avl_r_size		
avl_r_rdata_valid		
avl_r_rdata		
<b>avl_w interface</b>		
avl_w_write_req	Avalon-MM Slave	Avalon-MM interface between memory interface and user logic for write requests.
avl_w_ready		
avl_w_addr		
avl_w_size		
avl_w_wdata		
avl_w_be		
<b>status interface</b>		
local_init_done	Conduit	Memory interface status signals.
local_cal_success		
local_cal_fail		
<b>oct interface</b>		
rup (Stratix III/IV, Arria II GZ, Arria II GX)	Conduit	OCT reference resistor pins for rup/rdn or rzqin.
rdn (Stratix III/IV, Arria II GZ, Arria II GX)		
rzq (Stratix V, Arria V, Cyclone V)		
<b>pll_sharing interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set <b>PLL sharing mode</b> to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_hr_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
pll_dr_clk		
<b>dll_sharing interface</b>		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set <b>DLL sharing mode</b> to master or slave.
dll_pll_locked		
<b>oct_sharing interface</b>		
seriesterminationcontrol(Stratix III/IV/V, Arria II GZ, Arria V, Cyclone V)	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set <b>OCT sharing mode</b> to master or slave.
parallelterminationcontrol (Stratix III/IV/V, Arria II GZ, Arria V, Cyclone V)		
terminationcontrol (Arria II GX)		

Note to Table:

1. Signals available only in DLL master mode.

## RLDRAM II Controller with UniPHY Interfaces

The following table lists the RLDRAM II signals available for each interface in Qsys and provides a description and guidance on how to connect those interfaces.

**Table 7-5: RLDRAM II Controller with UniPHY Interfaces**

Interface Name	Interface Type	Description
<b>pll_ref_clk interface</b>		
pll_ref_clk	Clock input.	PLL reference clock input.
<b>global_reset interface</b>		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
<b>soft_reset interface</b>		
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
<b>afi_reset interface</b>		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. This interface is asserted when the PLL loses lock or the PHY is reset.
<b>afi_reset_export interface</b>		
afi_reset_export_n	Reset output (PLL master/no sharing)	This interface is a copy of the afi_reset interface. It is intended to be connected to PLL sharing slaves.
<b>afi_reset_in interface</b>		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the afi_reset_export_n output of an identically configured memory interface in PLL master mode.
<b>afi_clk interface</b>		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.

Interface Name	Interface Type	Description
<b>afi_clk_in interface</b>		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, you must connect this <code>afi_clk</code> input to the <code>afi_clk</code> output of an identically configured memory interface in PLL master mode.
<b>afi_half_clk interface</b>		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of the <code>afi_clk</code> . When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_half_clk_in interface</b>		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of the <code>afi_clk</code> . When the interface is in PLL slave mode, you must connect this <code>afi_half_clk</code> input to the <code>afi_half_clk</code> output of an identically configured memory interface in PLL master mode.
<b>memory interface</b>		

Interface Name	Interface Type	Description
mem_a	Conduit	Interface signals between the PHY and the memory device.
mem_ba		
mem_ck		
mem_ck_n		
mem_cs_n		
mem_dk		
mem_dk_n		
mem_dm		
mem_dq		
mem_qk		
mem_qk_n		
mem_ref_n		
mem_we_n		

**avl interface**

avl_size	Avalon-MM Slave	Avalon-MM interface between memory interface and user logic.
avl_wdata		
avl_rdata_valid		
avl_rdata		
avl_ready		
avl_write_req		
avl_read_req		
avl_addr		

**status interface**

Interface Name	Interface Type	Description
local_init_done		
local_cal_success	Conduit	Memory interface status signals.
local_cal_fail		
<b>oct interface</b>		
rup (Stratix III/IV, Arria II GZ)		
rdn (Stratix III/IV, Arria II GZ)	Conduit	OCT reference resistor pins for rup/rdn or rzqin.
rzq (Stratix V)		
<b>pll_sharing interface</b>		
pll_mem_clk		
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set <b>PLL sharing mode</b> to master or slave.
pll_config_clk		
pll_hr_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
pll_dr_clk		
<b>dll_sharing interface</b>		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set <b>DLL sharing mode</b> to master or slave.
<b>oct_sharing interface</b>		

Interface Name	Interface Type	Description
seriesterminationcontrol		OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set <b>OCT sharing mode</b> to master or slave.
parallelterminationcontrol	Conduit	
<b>parity_error_interrupt interface</b>		
parity_error	Conduit	Parity error interrupt conduit for connection to custom control block. This interface is enabled only if you turn on <b>Enable Error Detection Parity</b> .
<b>user_refresh interface</b>		
ref_req	Conduit	User refresh interface for connection to custom control block. This interface is enabled only if you turn on <b>Enable User Refresh</b> .
ref_ba		
ref_ack		
<b>reserved interface</b>		
reserved	Conduit	Reserved interface required for certain pin configurations when you select the Nios® II-based sequencer.

Note to Table:

1. Signals available only in DLL master mode.

## RLDRAM 3 UniPHY Interface

The following table lists the RLDRAM 3 signals available for each interface in Qsys and provides a description and guidance on how to connect those interfaces.

**Table 7-6: RLDRAM 3 UniPHY Interface**

Signals in Interface	Interface Type	Description/How to Connect
<b>pll_ref_clk interface</b>		
pll_ref_clk	Clock input	PLL reference clock input.
<b>global_reset interface</b>		
global_reset_n	Reset input	Asynchronous global reset for PLL and all logic in PHY.
<b>soft_reset interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
soft_reset_n	Reset input	Asynchronous reset input. Resets the PHY, but not the PLL that the PHY uses.
<b>afi_reset interface</b>		
afi_reset_n	Reset output (PLL master/no sharing)	When the interface is in PLL master or no sharing modes, this interface is an asynchronous reset output of the AFI interface. The controller asserts this interface when the PLL loses lock or the PHY is reset.
<b>afi_reset_export interface</b>		
afi_reset_export_n	Reset output (PLL master/no sharing)	This interface is a copy of the afi_reset interface. It is intended to be connected to PLL sharing slaves.
<b>afi_reset_in interface</b>		
afi_reset_n	Reset input (PLL slave)	When the interface is in PLL slave mode, this interface is a reset input that you must connect to the afi_reset_export_n output of an identically configured memory interface in PLL master mode.
<b>afi_clk interface</b>		
afi_clk	Clock output (PLL master/no sharing)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_clk_in interface</b>		
afi_clk	Clock input (PLL slave)	This AFI interface clock can be a full-rate or half-rate memory clock frequency based on the memory interface parameterization. When the interface is in PLL slave mode, you must connect this afi_clk input to the afi_clk output of an identically configured memory interface in PLL master mode.

Signals in Interface	Interface Type	Description/How to Connect
<b>afi_half_clk interface</b>		
afi_half_clk	Clock output (PLL master/no sharing)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL master or no sharing modes, this interface is a clock output.
<b>afi_half_clk_in interface</b>		
afi_half_clk	Clock input (PLL slave)	The AFI half clock that is half the frequency of afi_clk. When the interface is in PLL slave mode, this is a clock input that you must connect to the afi_half_clk output of an identically configured memory interface in PLL master mode.
<b>memory interface</b>		
mem_a	Conduit	Interface signals between the PHY and the memory device.
mem_ba		
mem_ck		
mem_ck_n		
mem_cs_n		
mem_dk		
mem_dk_n		
mem_dm		
mem_dq		
mem_qk		
mem_qk_n		
mem_ref_n		
mem_we_n		
mem_reset_n		
<b>afi interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
afi_addr	Avalon-MM Slave	Altera PHY interface (AFI) signals between the PHY and controller.
afi_ba		
afi_cs_n		
afi_we_n		
afi_ref_n		
afi_wdata_valid		
afi_wdata		
afi_dm		
afi_rdata		
afi_rdata_en		
afi_rdata_en_full		
afi_rdata_valid		
afi_RST_n		
afi_cal_success		
afi_cal_fail		
afi_wlat		
afi_rlat		
<b>oct interface</b>		
oct_rzqin	Conduit	OCT reference resistor pins for rzqin.
<b>pll_sharing interface</b>		

Signals in Interface	Interface Type	Description/How to Connect
pll_mem_clk	Conduit	Interface signals for PLL sharing, to connect PLL masters to PLL slaves. This interface is enabled only when you set PLL sharing mode to master or slave.
pll_write_clk		
pll_addr_cmd_clk		
pll_locked		
pll_avl_clk		
pll_config_clk		
pll_mem_phy_clk		
afi_phy_clk		
pll_write_clk_pre_phy_clk		
pll_p2c_read_clk		
pll_c2p_write_clk		
<b>dll_sharing interface</b>		
dll_delayctrl	Conduit	DLL sharing interface for connecting DLL masters to DLL slaves. This interface is enabled only when you set DLL sharing mode to master or slave.
dll_pll_locked		
<b>oct_sharing interface</b>		
serieterminationcontrol	Conduit	OCT sharing interface for connecting OCT masters to OCT slaves. This interface is enabled only when you set OCT sharing mode to master or slave.
parallelterminationcontrol		

## Generated Files for Memory Controllers with the UniPHY IP

When you complete the IP generation flow, there are generated files created in your project directory. The directory structure created varies somewhat, depending on the tool used to parameterize and generate the IP.

**Note:** The PLL parameters are statically defined in the `<variation_name>_parameters.tcl` at generation time. To ensure timing constraints and timing reports are correct, when you edit the PLL parameters, apply those changes to the PLL parameters in this file.

The following table lists the generated directory structure and key files created with the IP Catalog and Qsys.

**Table 7-7: Generated Directory Structure and Key Files for the IP Catalog Synthesis Files**

Directory	File Name	Description
<code>&lt;working_dir&gt;/</code>	<code>&lt;variation_name&gt;.qip</code>	Quartus Prime IP file which refers to all generated files in the synthesis fileset. Include this file in your Quartus Prime project.
<code>&lt;working_dir&gt;/</code>	<code>&lt;variation_name&gt;.v</code> or <code>&lt;variation_name&gt;.vhd</code>	Top-level wrapper synthesis files. .v is IEEE Encrypted Verilog. .vhd is generated VHDL.
<code>&lt;working_dir&gt;/&lt;variation_name&gt;/</code>	<code>&lt;variation_name&gt;_0002.v</code>	UniPHY top-level wrapper.
<code>&lt;working_dir&gt;/&lt;variation_name&gt;/</code>	<code>*.v, *.sv, *.tcl, *.sdc, *.ppf</code>	RTL and constraints files for synthesis.
<code>&lt;working_dir&gt;/&lt;variation_name&gt;/</code>	<code>&lt;variation_name&gt;_p0_pin_assignments.tcl</code>	Pin constraints script to be run after synthesis.

**Table 7-8: Generated Directory Structure and Key Files for the IP Catalog Simulation Files**

Directory	File Name	Description
<code>&lt;working_dir&gt;/&lt;variation_name&gt;_sim/</code>	<code>&lt;variation_name&gt;.v</code>	Top-level wrapper simulation files for both Verilog and VHDL.
<code>&lt;working_dir&gt;/&lt;variation_name&gt;_sim/&lt;subcomponent_module&gt;/</code>	<code>*.v, *.sv, *.vhd, *.vho, *.hex, *.mif</code>	RTL and constraints files for simulation. .v and .sv files are IEEE Encrypted Verilog. .vhd and .vho are generated VHDL.

**Table 7-9: Generated Directory Structure and Key Files for the IP Catalog—Example Design Fileset Synthesis Files**

Directory	File Name	Description
<code>&lt;variation_name&gt;_example_design/example_project/</code>	<code>&lt;variation_name&gt;_example.qip</code>	Quartus Prime IP file that refers to all generated files in the synthesizable project.
<code>&lt;variation_name&gt;_example_design/example_project/</code>	<code>&lt;variation_name&gt;_example.qpf</code>	Quartus Prime project for synthesis flow.

Directory	File Name	Description
<code>&lt;variation_name&gt;_example_design/example_project/</code>	<code>&lt;variation_name&gt;_example.qsf</code>	Quartus Prime project for synthesis flow.
<code>&lt;variation_name&gt;_example_design/example_project/ &lt;variation_name&gt;_example/</code>	<code>&lt;variation_name&gt;_example.v</code>	Top-level wrapper.
<code>&lt;variation_name&gt;_example_design/example_project/ &lt;variation_name&gt;_example/ submodules/</code>	<code>*.v, *.sv, *.tcl, *.sdc, *.ppf</code>	RTL and constraints files.
<code>&lt;variation_name&gt;_example_design/example_project/ &lt;variation_name&gt;_example/ submodules/</code>	<code>&lt;variation_name&gt;_example_if0_p0_pin_assignments.tcl</code>	Pin constraints script to be run after synthesis.  _if0 and _p0 are instance names.

**Table 7-10: Generated Directory Structure and Key Files for the IP Catalog—Example Design Fileset Simulation Files**

Directory	File Name	Description
<code>&lt;variation_name&gt;_example_design/simulation/</code>	<code>generate_sim_verilog_example_design.tcl</code>	Run this file to generate the Verilog simulation example design.
<code>&lt;variation_name&gt;_example_design/simulation/</code>	<code>generate_sim_vhdl_example_design.tcl</code>	Run this file to generate the VHDL simulation example design.
<code>&lt;variation_name&gt;_example_design/simulation/</code>	<code>README.txt</code>	A text file with instructions about how to generate and run the simulation example design.
<code>&lt;variation_name&gt;_example_design/simulation/verilog/mentor</code>	<code>run.do</code>	ModelSim script to simulate the generated Verilog example design.
<code>&lt;variation_name&gt;_example_design/simulation/vhdl/mentor</code>	<code>run.do</code>	ModelSim script to simulate the generated VHDL example design.

Directory	File Name	Description
<code>&lt;variation_name&gt;_example_design/simulation/verilog/&lt;variation_name&gt;_sim/</code>	<code>&lt;variation_name&gt;_example_sim.v</code>	Top-level wrapper (Testbench) for Verilog.
<code>&lt;variation_name&gt;_example_design/simulation/vhdl/&lt;variation_name&gt;_sim/</code>	<code>&lt;variation_name&gt;_example_sim.vhd</code>	Top-level wrapper (Testbench) for VHDL.
<code>&lt;variation_name&gt;_example_design/simulation/&lt;variation_name&gt;_sim/verilog/submodules/</code>	<code>*.v, *.sv, *.hex, *.mif</code>	RTL and ROM data for Verilog.
<code>&lt;variation_name&gt;_example_design/simulation/&lt;variation_name&gt;_sim/vhdl/submodules/</code>	<code>*.vhd, *.vho, *.hex, *.mif</code>	RTL and ROM data for VHDL.

Table 7-11: Generated Directory Structure and Key Files for Qsys

Directory	File Name	Description
<code>&lt;working_dir&gt;/&lt;system_name&gt;/synthesis/</code>	<code>&lt;system_name&gt;.qip</code>	Quartus Prime IP file that refers to all the generated files in the synthesis fileset.
<code>&lt;working_dir&gt;/&lt;system_name&gt;/synthesis/</code>	<code>&lt;system_name&gt;.v</code>	System top-level RTL for synthesis.
<code>&lt;working_dir&gt;/&lt;system_name&gt;/simulation/</code>	<code>&lt;system_name&gt;.v or &lt;variation_name&gt;.vhd</code>	System top-level RTL for simulation. .v file is IEEE Encrypted Verilog. .vhd file is generated VHDL.
<code>&lt;working_dir&gt;/&lt;system_name&gt;/synthesis/submodules/</code>	<code>*.v, *.sv, *.tcl, *.sdc, *.ppf</code>	RTL and constraints files for synthesis.
<code>&lt;working_dir&gt;/&lt;system_name&gt;/simulation/submodules/</code>	<code>*.v, *.sv, *.hex, *.mif</code>	RTL and ROM data for simulation.

The following table lists the prefixes or instance names of submodule files within the memory interface IP. These instances are concatenated to form unique synthesis and simulation filenames.

**Table 7-12: Prefixes of Submodule Files**

Prefixes	Description
_c0	Specifies the controller.
_d0	Specifies the driver or traffic generator.
_dll0	Specifies the DLL.
_e0	Specifies the example design.
_ifo	Specifies the memory Interface.
_m0	Specifies the AFI mux.
_oct0	Specifies the OCT.
_p0	Specifies the PHY.
_pll0	Specifies the PLL.
_s0	Specifies the sequencer.
_t0	Specifies the traffic generator status checker.

## Parameterizing Memory Controllers

This section describes the parameters you can set for various UniPHY-based memory controllers.

### Parameterizing Memory Controllers with UniPHY IP

The **Parameter Settings** page in the parameter editor allows you to parameterize the following settings for the LPDDR2, DDR2, DDR3 SDRAM, QDR II, QDR II+ SRAM, RLDRAM II, and RLDRAM 3 memory controllers with the UniPHY IP:

- PHY Settings
- Memory Parameters
- Memory Timing
- Board Settings
- Controller Settings
- Diagnostics

The messages window at the bottom of the parameter editor displays information about the memory interface, warnings, and errors if you are trying to create something that is not supported.

### Enabling the Hard Memory Interface

For Arria V and Cyclone V devices, enable the hard memory interface by turning on **Interface Type > Enable Hard Memory Interface** in the parameter editor. The hard memory interface uses the hard memory controller and hard memory PHY blocks in the Arria V and Cyclone V devices.

The half-rate bridge option is available only as an SOPC Builder component, **Avalon-MM DDR Memory Half-Rate Bridge**, for use in a Qsys project.

## PHY Settings for UniPHY IP

The following table lists the PHY parameters for UniPHY-based EMIF IP.

**Table 7-13: PHY Parameters**

Parameter	Description
<b>General Settings</b>	
<b>Speed Grade</b>	Specifies the speed grade of the targeted FPGA device that affects the generated timing constraints and timing reporting.
<b>Generate PHY only</b>	Turn on this option to generate the UniPHY core without a memory controller. When you turn on this option, the AFI interface is exported so that you can easily connect your own memory controller.  Not applicable to RLDRAM 3 UniPHY as no controller support for RLDRAM 3 UniPHY.
<b>Clocks</b>	
<b>Memory clock frequency</b>	The frequency of the clock that drives the memory device. Use up to 4 decimal places of precision.  To obtain the maximum supported frequency for your target memory configuration, refer to the External Memory Interface Spec Estimator page on the Altera website.
<b>Achieved memory clock frequency</b>	The actual frequency the PLL generates to drive the external memory interface (memory clock).
<b>PLL reference clock frequency</b>	The frequency of the input clock that feeds the PLL. Use up to 4 decimal places of precision.
<b>Rate on Avalon-MM interface</b>	The width of data bus on the Avalon-MM interface. <b>Full</b> results in a width of 2× the memory data width. <b>Half</b> results in a width of 4× the memory data width. <b>Quarter</b> results in a width of 8× the memory data width. Use <b>Quarter</b> for memory frequency 533 MHz and above.  To determine the Avalon-MM interface rate selection for other memories, refer to the local interface clock rate for your target device in the External Memory Interface Spec Estimator page on the Altera website.  <b>Note:</b> MAX 10 devices support only half-rate Avalon-MM interface.
<b>Achieved local clock frequency</b>	The actual frequency the PLL generates to drive the local interface for the memory controller (AFI clock).
<b>Enable AFI half rate clock</b>	Export the afi_half_rate clock which is running half of the AFI clock rate to the top level.

Parameter	Description
<b>Advanced PHY Settings</b>	
<b>Advanced clock phase control</b>	<p>Enables access to clock phases. Default value should suffice for most DIMMs and board layouts, but can be modified if necessary to compensate for larger address and command versus clock skews.</p> <p>This option is available for DDR, DDR2 and DDR3 SDRAM only.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
<b>Additional address and command clock phase</b>	<p>Allows you to increase or decrease the amount of phase shift on the address and command clock. The base phase shift center aligns the address and command clock at the memory device, which may not be the optimal setting under all circumstances. Increasing or decreasing the amount of phase shift can improve timing. The default value is 0 degrees.</p> <p>In DDR, DDR2, DDR3 SDRAM, and LPDDR2 SDRAM, you can set this value from -360 to 360 degrees. In QDRII/II+ SRAM and RLDRAM II, the available settings are -45, -22.5, 22.5, and 45.</p> <p>To achieve the optimum setting, adjust the value based on the address and command timing analysis results.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
<b>Additional phase for core-to-periphery transfer</b>	<p>Allows you to phase shift the latching clock of the core-to-periphery transfers. By delaying the latch clock, a positive phase shift value improves setup timing for transfers between registers in the core and the half-rate DDIO_OUT blocks in the periphery, respectively. Adjust this setting according to the core timing analysis.</p> <p>The default value is 0 degrees. You can set this value from -179 to 179 degrees.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>

Parameter	Description
<b>Additional CK/CK# phase</b>	<p>Allows you to increase or decrease the amount of phase shift on the CK/CK# clock. The base phase shift center aligns the address and command clock at the memory device, which may not be the optimal setting under all circumstances. Increasing or decreasing the amount of phase shift can improve timing. Increasing or decreasing the phase shift on CK/CK# also impacts the read, write, and leveling transfers, which increasing or decreasing the phase shift on the address and command clocks does not.</p> <p>To achieve the optimum setting, adjust the value based on the address and command timing analysis results. Ensure that the read, write, and write leveling timings are met after adjusting the clock phase. Adjust this value when there is a core timing failure after adjusting <b>Additional address and command clock phase</b>.</p> <p>The default value is 0 degrees. You can set this value from -360 to 360 degrees.</p> <p>This option is available for LPDDR2, DDR, DDR2, and DDR3 SDRAM only.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
<b>Supply voltage</b>	<p>The supply voltage and sub-family type of memory.</p> <p>This option is available for DDR3 SDRAM only.</p>
<b>I/O standard</b>	<p>The I/O standard voltage. Set the I/O standard according to your design's memory standard.</p>
<b>PLL sharing mode</b>	<p>When you select <b>No sharing</b>, the parameter editor instantiates a PLL block without exporting the PLL signals. When you select <b>Master</b>, the parameter editor instantiates a PLL block and exports the signals. When you select <b>Slave</b>, the parameter editor exposes a PLL interface and you must connect an external PLL master to drive the PLL slave interface signals.</p> <p>Select <b>No sharing</b> if you are not sharing PLLs, otherwise select <b>Master</b> or <b>Slave</b>.</p> <p>For more information about resource sharing, refer to “The DLL and PLL Sharing Interface” section in the <i>Functional Description—UniPHY</i> chapter of the <i>External Memory Interface Handbook</i>.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
<b>Number of PLL sharing interfaces</b>	<p>This option allows you to specify the number of PLL sharing interfaces to create, facilitating creation of many one-to-one connections in Qsys flow. In Megawizard, you can select one sharing interface and manually connect the master to all the slaves.</p> <p>This option is enabled when you set <b>PLL sharing mode</b> to <b>Master</b>.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>

Parameter	Description
<b>DLL sharing mode</b>	<p>When you select <b>No sharing</b>, the parameter editor instantiates a DLL block without exporting the DLL signals. When you select <b>Master</b>, the parameter editor instantiates a DLL block and exports the signals. When you select <b>Slave</b>, the parameter editor exposes a DLL interface and you must connect an external DLL master to drive the DLL slave signals.</p> <p>Select <b>No sharing</b> if you are not sharing DLLs, otherwise select <b>Master</b> or <b>Slave</b>.</p> <p>For more information about resource sharing, refer to “The DLL and PLL Sharing Interface” section in the <i>Functional Description—UniPHY</i> chapter of the <i>External Memory Interface Handbook</i>.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
<b>Number of DLL sharing interfaces</b>	<p>This option allows you to specify the number of DLL sharing interfaces to create, facilitating creation of many one-to-one connections in Qsys flow. In Megawizard, you can select one sharing interface and manually connect the master to all the slaves.</p> <p>This option is enabled when you set <b>PLL sharing mode</b> to <b>Master</b>.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
<b>OCT sharing mode</b>	<p>When you select <b>No sharing</b>, the parameter editor instantiates an OCT block without exporting the OCT signals. When you select <b>Master</b>, the parameter editor instantiates an OCT block and exports the signals. When you select <b>Slave</b>, the parameter editor exposes an OCT interface and you must connect an external OCT control block to drive the OCT slave signals.</p> <p>Select <b>No sharing</b> if you are not sharing OCT blocks, otherwise select <b>Master</b> or <b>Slave</b>.</p> <p>For more information about resource sharing, refer to “The OCT Sharing Interface” section in the <i>Functional Description—UniPHY</i> chapter of the <i>External Memory Interface Handbook</i>.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
<b>Number of OCT sharing interfaces</b>	<p>This option allows you to specify the number of OCT sharing interfaces to create, facilitating creation of many one-to-one connections in Qsys flow. In Megawizard, you can select one sharing interface and manually connect the master to all the slaves.</p> <p>This option is enabled when you set <b>PLL sharing mode</b> to <b>Master</b>.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
<b>Reconfigurable PLL location</b>	<p>When you set the PLL used in the UniPHY memory interface to be reconfigurable at run time, you must specify the location of the PLL. This assignment generates a PLL that can only be placed in the given sides.</p>

Parameter	Description
<b>Sequencer optimization</b>	<p>Select <b>Performance</b> to enable the Nios II-based sequencer, or <b>Area</b> to enable the RTL-based sequencer.</p> <p>Altera recommends that you enable the Nios-based sequencer for memory clock frequencies greater than 400 MHz and enable the RTL-based sequencer if you want to reduce resource utilization.</p> <p>This option is available for QDRII and QDR II+ SRAM, and RLDRAM II only.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>

**Related Information**

- [External Memory Interface Spec Estimator](#)
- [Functional Description–UniPHY](#)

**Memory Parameters for LPDDR2, DDR2 and DDR3 SDRAM for UniPHY IP**

The following table lists the memory parameters for LPDDR2, DDR2 and DDR3 SDRAM.

Use the **Memory Parameters** tab to apply the memory parameters from your memory manufacturer's data sheet.

**Table 7-14: Memory Parameters for LPDDR2, DDR2, and DDR3 SDRAM**

Parameter	Description
<b>Memory vendor</b>	The vendor of the memory device. Select the memory vendor according to the memory vendor you use. For memory vendors that are not listed in the setting, select JEDEC with the nearest memory parameters and edit the parameter values according to the values of the memory vendor that you use. However, if you select a configuration from the list of memory presets, the default memory vendor for that preset setting is automatically selected.
<b>Memory format</b>	The format of the memory device. Select <b>Discrete</b> if you are using just the memory device. Select <b>Unbuffered</b> or <b>Registered</b> for DIMM format. Use the DIMM format to turn on levelling circuitry for LPDDR2 support device only. DDR2 supports DIMM also.

Parameter	Description
<b>Number of clock enables per device/DIMM</b>	The number of clock enable pins per device or DIMM. This value also determines the number of ODT signals. (This parameter is available only when the selected memory format is <b>Registered</b> .)  <b>Note:</b> This parameter is not available for MAX 10 devices.
<b>Number of chip selects per device/DIMM</b>	The number of chip selects per device or DIMM. This value is not necessarily the same as the number of ranks for RDIMMs or LRDIMMs. This value must be 2 or greater for RDIMMs or LRDIMMs. (This parameter is available only when the selected memory format is <b>Registered</b> .)  <b>Note:</b> This parameter is not available for MAX 10 devices.
<b>Number of ranks per slot</b>	The number of ranks per DIMM slot. (This parameter is available only when the selected memory format is <b>Registered</b> .)  <b>Note:</b> This parameter is not available for MAX 10 devices.
<b>Number of slots</b>	The number of DIMM slots. (This parameter is available only when the selected memory format is <b>Registered</b> .)  <b>Note:</b> This parameter is not available for MAX 10 devices.
<b>Memory device speed grade</b>	The maximum frequency at which the memory device can run.
<b>Total interface width</b>	The total number of DQ pins of the memory device. Limited to 144 bits for DDR2 and DDR3 SDRAM (with or without leveling).  The total interface is depending on the rate on Avalon-MM interface because the maximum Avalon data width is 1024. If you select 144 bit for total interface width with Quarter-rate, the avalon data width is 1152 exceeding maximum avalon data width.
<b>DQ/DQS group size</b>	The number of DQ bits per DQS group.

Parameter	Description
<b>Number of DQS groups</b>	The number of DQS groups is calculated automatically from the Total interface width and the DQ/DQS group size parameters.
<b>Number of chip selects (DDR2 and DDR3 SDRAM device only)</b>	The number of chip-selects the IP core uses for the current device configuration. Specify the total number of chip-selects according to the number of memory device.
<b>Number of clocks</b>	The width of the clock bus on the memory interface.
<b>Row address width</b>	The width of the row address on the memory interface.
<b>Column address width</b>	The width of the column address on the memory interface.
<b>Bank-address width</b>	The width of the bank address bus on the memory interface.
<b>Enable DM pins</b>	Specifies whether the DM pins of the memory device are driven by the FPGA. You can turn off this option to avoid overusing FPGA device pins when using x4 mode memory devices.  When you are using x4 mode memory devices, turn off this option for DDR3 SDRAM.  You must turn on this option if you are using Avalon byte enable.
<b>DQS# Enable (DDR2)</b>	Turn on differential DQS signaling to improve signal integrity and system performance.  This option is available for DDR2 SDRAM only.

#### Related Information

- [Optimizing the Controller](#) on page 11-1
- [DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines](#) on page 2-1

#### Memory Initialization Options for DDR2

## Memory Initialization Options—DDR2

	<b>Address and command parity</b>	Enables address/command parity checking. This is required for Registered DIMM.
<b>Mode Register 0</b>	<b>Burst length</b>	Specifies the burst length.
	<b>Read burst type</b>	Specifies accesses within a given burst in sequential or interleaved order.  Specify sequential ordering for use with the Altera memory controller. Specify interleaved ordering only for use with an interleaved-capable custom controller, when the <b>Generate PHY only</b> parameter is enabled on the PHY Settings tab.
	<b>DLL precharge power down</b>	Determines whether the DLL in the memory device is in slow exit mode or in fast exit mode during precharge power down. For more information, refer to memory vendor data sheet.
	<b>Memory CAS latency setting</b>	Determines the number of clock cycles between the READ command and the availability of the first bit of output data at the memory device. For more information, refer to memory vendor data sheet speed bin table.  Set this parameter according to the target memory speed grade and memory clock frequency.

Memory Initialization Options—DDR2		
<b>Mode Register 1</b>	<b>Output drive strength setting</b>	Determines the output driver impedance setting at the memory device.  To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
	<b>Memory additive CAS latency setting</b>	Determines the posted CAS additive latency of the memory device.  Enable this feature to improve command and bus efficiency, and increase system bandwidth. For more information, refer to the <i>Optimizing the Controller</i> chapter.
	<b>Memory on-die termination (ODT) setting</b>	Determines the on-die termination resistance at the memory device.  To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
<b>Mode Register 2</b>	<b>SRT Enable</b>	Determines the selfrefresh temperature (SRT). Select <b>1x refresh rate</b> for normal temperature (0–85C) or select <b>2x refresh rate</b> for high-temperature (>85C).

### Memory Initialization Options for DDR3

Memory Initialization Options—DDR3	
<b>Mirror Addressing: 1 per chip select</b>	Specifies the mirror addressing for multiple rank DIMMs. Refer to memory vendor data sheet for more information. Enter ranks with mirrored addresses in this field. For example, for four chip selects, enter 1101 to mirror the address on chip select #3, #2, and #0.
<b>Address and command parity</b>	Enables address/command parity checking to detect errors in data transmission. This is required for registered DIMM (RDIMM).

## Memory Initialization Options—DDR3

	<b>Read burst type</b>	Specifies accesses within a given burst in sequential or interleaved order.  Specify sequential ordering for use with the Altera memory controller. Specify interleaved ordering only for use with an interleaved-capable custom controller, when the <b>Generate PHY only</b> parameter is enabled on the PHY Settings tab.
<b>Mode Register 0</b>	<b>DLL precharge power down</b>	Specifies whether the DLL in the memory device is off or on during precharge power-down.
	<b>Memory CAS latency setting</b>	The number of clock cycles between the read command and the availability of the first bit of output data at the memory device and also interface frequency. Refer to memory vendor data sheet speed bin table.  Set this parameter according to the target memory speed grade and memory clock frequency.
<b>Mode Register 1</b>	<b>Output drive strength setting</b>	The output driver impedance setting at the memory device.  To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.
	<b>Memory additive CAS latency setting</b>	The posted CAS additive latency of the memory device. Enable this feature to improve command and bus efficiency, and increase system bandwidth. For more information, refer to the <i>Optimizing the Controller</i> chapter.
	<b>ODT Rtt nominal value</b>	The on-die termination resistance at the memory device.  To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.

## Memory Initialization Options—DDR3

<b>Mode Register 2</b>	<b>Auto selfrefresh method</b>	Disable or enable auto selfrefresh.
	<b>Selfrefresh temperature</b>	Specifies the selfrefresh temperature as <b>Normal</b> or <b>Extended</b> .
	<b>Memory write CAS latency setting</b>	The number of clock cycles from the releasing of the internal write to the latching of the first data in, at the memory device and also interface frequency. Refer to memory vendor data sheet speed bin table and set according to the target memory speed grade and memory clock frequency.
	<b>Dynamic ODT (Rtt_WR) value</b>	<p>The mode of the dynamic ODT feature of the memory device. This is used for multi-rank configurations. Refer to <i>DDR2 and DDR3 SDRAM Board Layout Guidelines</i>.</p> <p>To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.</p>
	<b>DDR3 RDIMM/LRDIMM Control Words</b>	<p>The memory device features a set of control words of SSTE32882 registers. These 4-bit control words of serial presence-detect (SPD) information allow the controller to optimize device properties to match different DIMM net topologies.</p> <p>You can obtain the control words from the memory manufacturer's data sheet. You enter each word in hexadecimal, starting with RC15 on the left and ending with RC0 on the right.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>
	<b>LRDIMM Additional Control Words</b>	<p>The memory device features a set of control words of SSTE32882 registers. These 4-bit control words of serial presence-detect (SPD) information allow the controller to optimize device properties to match different DIMM net topologies.</p> <p>You can obtain the control words from the memory manufacturer's data sheet. You enter each word in hexadecimal, starting with SPD (77-72) or SPD(83-78) on the left and ending with SPD(71-69) on the right.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>

## Memory Initialization Options for LPDDR2

## Memory Initialization Options—LPDDR2

<b>Mode Register 1</b>	<b>Burst Length</b>	Specifies the burst length.
	<b>Read Burst Type</b>	<p>Specifies accesses within a given burst in sequential or interleaved order.</p> <p>Specify sequential ordering for use with the Altera memory controller.</p> <p>Specify interleaved ordering only for use with an interleaved-capable custom controller, when the <b>Generate PHY only</b> parameter is enabled on the PHY Settings tab.</p>
<b>Mode Register 2</b>	<b>Memory CAS latency setting</b>	<p>Determines the number of clock cycles between the READ command and the availability of the first bit of output data at the memory device.</p> <p>Set this parameter according to the target memory interface frequency. Refer to memory data sheet and also target memory speed grade.</p>
<b>Mode Register 3</b>	<b>Output drive strength settings</b>	<p>Determines the output driver impedance setting at the memory device.</p> <p>To obtain the optimum signal integrity performance, select the optimum setting based on the board simulation results.</p>

## Memory Parameters for QDR II and QDR II+ SRAM for UniPHY IP

The following table describes the memory parameters for QDR II and QDR II+ SRAM for UniPHY IP.

Use the **Memory Parameters** tab to apply the memory parameters from your memory manufacturer's data sheet.

Table 7-15: Memory Parameters for QDR II and QDR II+ SRAM

Parameter	Description
<b>Address width</b>	The width of the address bus on the memory device.
<b>Data width</b>	The width of the data bus on the memory device.
<b>Data-mask width</b>	The width of the data-mask on the memory device.
<b>CQ width</b>	The width of the CQ (read strobe) bus on the memory device.

Parameter	Description
<b>K width</b>	The width of the K (write strobe) bus on the memory device.
<b>Burst length</b>	The burst length supported by the memory device. For more information, refer to memory vendor data sheet.
<b>Topology</b>	
<b>x36 emulated mode</b>	Emulates a larger memory-width interface using smaller memory-width interfaces on the FPGA.  Turn on this option when the target FPGA do not support x36 DQ/DQS group. This option allows two x18 DQ/DQS groups to emulate 1 x36 read data group.
<b>Emulated write groups</b>	Number of write groups to use to form the x36 memory interface on the FPGA. Select 2 to use 2 x18 DQ/DQS group to form x36 write data group. Select 4 to use 4 x9 DQ/DQS group to form x36 write data group.
<b>Device width</b>	Specifies the number of memory devices used for width expansion.

### Memory Parameters for RLDRAM II for UniPHY IP

The following table describes the memory parameters for RLDRAM II.

Use the **Memory Parameters** tab to apply the memory parameters from your memory manufacturer's data sheet.

**Table 7-16: Memory Parameters for RLDRAM II**

Parameter	Description
<b>Address width</b>	The width of the address bus on the memory device.
<b>Data width</b>	The width of the data bus on the memory device.
<b>Bank-address width</b>	The width of the bank-address bus on the memory device.
<b>Data-mask width</b>	The width of the data-mask on the memory device,

Parameter	Description
<b>QK width</b>	The width of the QK (read strobe) bus on the memory device. Select <b>1</b> when data width is set to 9. Select <b>2</b> when data width is set to 18 or 36.
<b>DK width</b>	The width of the DK (write strobe) bus on the memory device. Select <b>1</b> when data width is set to 9 or 18. Select <b>2</b> when data width is set to 36.
<b>Burst length</b>	The burst length supported by the memory device. For more information, refer to memory vendor data sheet.
<b>Memory mode register configuration</b>	Configuration bits that set the memory mode. Select the option according to the interface frequency.
<b>Device impedance</b>	Select External (ZQ) to adjust the driver impedance using the external impedance resistor (RQ). The output impedance range is 25-60. You must connect the RQ resistor between ZQ pin and ground. The value of RQ must be 5 times the output impedance. For example, 60 output impedance requires 300 RQ. Set the value according to the board simulation.
<b>On-Die Termination</b>	Turn on this option to enable ODT in the memory to terminate the DQs and DM pins to Vtt. Dynamically switch off during read operation and switch on during write operation. Refer to memory vendor data sheet for more information.
<b>Topology</b>	
<b>Device width</b>	Specifies the number of memory devices used for width expansion.

## Memory Timing Parameters for DDR2, DDR3, and LPDDR2 SDRAM for UniPHY IP

The following table lists the memory timing parameters for DDR2, DDR3, and LPDDR2 SDRAM.

Use the **Memory Timing** tab to apply the memory timings from your memory manufacturer's data sheet.

**Table 7-17: Parameter Description**

Parameter	Protocol	Description
<b>tIS (base)</b>	DDR2, DDR3, LPDDR2	Address and control setup to CK clock rise. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tIH (base)</b>	DDR2, DDR3, LPDDR2	Address and control hold after CK clock rise. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDS (base)</b>	DDR2, DDR3, LPDDR2	Data setup to clock (DQS) rise. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDH (base)</b>	DDR2, DDR3, LPDDR2	Data hold after clock (DQS) rise. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDQSQ</b>	DDR2, DDR3, LPDDR2	DQS, DQS# to DQ skew, per access. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tQHS</b>	DDR2, LPDDR2	DQ output hold time from DQS, DQS# (absolute time value)
<b>tQH</b>	DDR3	DQ output hold time from DQS, DQS# (percentage of tCK). Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDQSCK</b>	DDR2, DDR3, LPDDR2	DQS output access time from CK/CK#. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDQSCK Delta Short</b>	LPDDR2	Absolute difference between any two tDQSCK measurements (within a byte lane) within a contiguous sequence of bursts in a 160ns rolling window. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDQSCK Delta Medium</b>	LPDDR2	Absolute difference between any two tDQSCK measurements (within a byte lane) within a contiguous sequence of bursts in a 1.6us rolling window. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDQSCK Delta Long</b>	LPDDR2	Absolute difference between any two tDQSCK measurements (within a byte lane) within a contiguous sequence of bursts in a 32ms rolling window. Set according to the memory speed grade and refer to the memory vendor data sheet.

Parameter	Protocol	Description
<b>tDQSS</b>	DDR2, DDR3, LPDDR2	First latching edge of DQS to associated clock edge (percentage of tCK). Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDQSH</b>	DDR2, LPDDR2	DQS Differential High Pulse Width (percentage of tCK). Specifies the minimum high time of the DQS signal received by the memory. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tQSH</b>	DDR3	
<b>tDSH</b>	DDR2, DDR3, LPDDR2	DQS falling edge hold time from CK (percentage of tCK). Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tDSS</b>	DDR2, DDR3, LPDDR2	DQS falling edge to CK setup time (percentage of tCK). Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tINIT</b>	DDR2, DDR3, LPDDR2	Memory initialization time at power-up. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tMRD</b>	DDR2, DDR3	Load mode register command period. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tMRW</b>	LPDDR2	
<b>tRAS</b>	DDR2, DDR3, LPDDR2	Active to precharge time. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tRCD</b>	DDR2, DDR3, LPDDR2	Active to read or write time. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tRP</b>	DDR2, DDR3, LPDDR2	Precharge command period. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tREFI</b>	DDR2, DDR3	Refresh command interval. (All banks only for LPDDR2.) Set according to the memory speed grade and temperature range. Refer to the memory vendor data sheet.
<b>tREFICab</b>	LPDDR2	
<b>tRFC</b>	DDR2, DDR3	Auto-refresh command interval. (All banks only for LPDDR2.) Set according to the memory device capacity. Refer to the memory vendor data sheet.
<b>tRFCab</b>	LPDDR2	

Parameter	Protocol	Description
<b>tWR</b>	DDR2, DDR3, LPDDR2	Write recovery time. Set according to the memory speed grade and refer to the memory vendor data sheet.
<b>tWTR</b>	DDR2, DDR3, LPDDR2	Write to read period. Set according to the memory speed grade and memory clock frequency. Refer to the memory vendor data sheet. Calculate the value based on the memory clock frequency.
<b>tFAW</b>	DDR2, DDR3, LPDDR2	Four active window time. Set according to the memory speed grade and page size. Refer to the memory vendor data sheet.
<b>tRRD</b>	DDR2, DDR3, LPDDR2	RAS to RAS delay time. Set according to the memory speed grade, page size and memory clock frequency. Refer to the memory vendor data sheet. Calculate the value based on the memory interface frequency and memory clock frequency.
<b>tRTP</b>	DDR2, DDR3, LPDDR2	Read to precharge time. Set according to memory speed grade. Refer to the memory vendor data sheet. Calculate the value based on the memory interface frequency and memory clock frequency.

### Memory Timing Parameters for QDR II and QDR II+ SRAM for UniPHY IP

The following table lists the memory timing parameters for QDR II and QDR II+ SRAM.

Use the **Memory Timing** tab to apply the memory timings from your memory manufacturer's data sheet.

**Table 7-18: Parameter Description**

Parameter	Description
<b>QDR II and QDR II+ SRAM</b>	
<b>tWL (cycles)</b>	The write latency. Set write latency 0 for burst length of 2, and set write latency to 1 for burst length of 4.
<b>tRL (cycles)</b>	The read latency. Set according to memory protocol. Refer to memory data sheet.
<b>tSA</b>	The address and control setup to K clock rise. Set according to memory protocol. Refer to memory data sheet.
<b>tHA</b>	The address and control hold after K clock rise. Set according to memory protocol. Refer to memory data sheet.
<b>tSD</b>	The data setup to clock (K/K#) rise. Set according to memory protocol. Refer to memory data sheet.

Parameter	Description
<b>tHD</b>	The data hold after clock (K/K#) rise. Set according to memory protocol. Refer to memory data sheet.
<b>tCQD</b>	Echo clock high to data valid. Set according to memory protocol. Refer to memory data sheet.
<b>tCQDOH</b>	Echo clock high to data invalid. Set according to memory protocol. Refer to memory data sheet.
<b>Internal jitter</b>	The QDRII/II+ internal jitter. Refer to memory data sheet.
<b>TCQHCQnH</b>	The CQ clock rise to CQ $n$ clock rise (rising edge to rising edge). Set according to memory speed grade. Refer to memory data sheet.
<b>TKHKnH</b>	The K clock rise to Kn clock rise (rising edge to rising edge). Set according to memory speed grade. Refer to memory data sheet.

### Memory Timing Parameters for RLDRAM II for UniPHY IP

The following table lists the memory timing parameters for RLDRAM II.

Use the **Memory Timing** tab to apply the memory timings from your memory manufacturer's data sheet.

**Table 7-19: Memory Timing Parameters**

Parameter	Description
<b>RLDRAM II</b>	
<b>Maximum memory clock frequency</b>	The maximum frequency at which the memory device can run. Set according to memory speed grade. Refer to memory data sheet.
<b>Refresh interval</b>	The refresh interval. Set according to memory speed grade. Refer to memory data sheet.
<b>tCKH (%)</b>	The input clock (CK/CK#) high expressed as a percentage of the full clock period. Set according to memory speed grade. Refer to memory data sheet.
<b>tQKH (%)</b>	The read clock (QK/QK#) high expressed as a percentage of tCKH. Set according to memory speed grade. Refer to memory data sheet.
<b>tAS</b>	Address and control setup to CK clock rise. Set according to memory speed grade. Refer to memory data sheet.
<b>tAH</b>	Address and control hold after CK clock rise. Set according to memory speed grade. Refer to memory data sheet.

Parameter	Description
<b>tDS</b>	Data setup to clock (CK/CK#) rise. Set according to memory speed grade. Refer to memory data sheet.
<b>tDH</b>	Data hold after clock (CK/CK#) rise. Set according to memory speed grade. Refer to memory data sheet.
<b>tQKQ_max</b>	QK clock edge to DQ data edge (in same group). Set according to memory speed grade. Refer to memory data sheet.
<b>tQKQ_min</b>	QK clock edge to DQ data edge (in same group). Set according to memory speed grade. Refer to memory data sheet.
<b>tCKDK_max</b>	Clock to input data clock (max). Set according to memory speed grade. Refer to memory data sheet.
<b>tCKDK_min</b>	Clock to input data clock (min). Set according to memory speed grade. Refer to memory data sheet.

### Memory Parameters for RLDRAM 3 for UniPHY IP

The following tables list the memory parameters for RLDRAM 3 for UniPHY IP.

Use the **Memory Timing** tab to apply the memory timings from your memory manufacturer's data sheet.

**Table 7-20: Memory Parameters for RLDRAM 3 for UniPHY**

Parameter	Description
<b>Enable data-mask pins</b>	Specifies whether the DM pins of the memory device are driven by the FPGA.
<b>Data-mask width</b>	The width of the data-mask on the memory device.
<b>Data width</b>	The width of the data bus on the memory device.
<b>QK width</b>	The width of the QK (read strobe) bus on the memory device. Select 2 when data width is set to 18. Select 4 when data width is set to 36.
<b>DK width</b>	The width of the DK (write strobe) bus on the memory device. For x36 device, DQ[8:0] and DQ[26:18] are referenced to DK0/DK0#, and DQ[17:9] and DQ[35:27] are referenced to DK1/DK1#.
<b>Address width</b>	The width of the address bus on the memory device.
<b>Bank-address width</b>	The width of the bank-address bus on the memory device.

Parameter	Description
<b>Burst length</b>	The burst length supported by the memory device. Refer to memory vendor data sheet.
<b>tRC</b>	Mode register bits that set the tRC. Set the tRC according to the memory speed grade and data latency. Refer to the tRC table in the memory vendor data sheet.
<b>Data Latency</b>	Mode register bits that set the latency. Set latency according to the interface frequency and memory speed grade. Refer to speed bin table in the memory data sheet.
<b>Output Drive</b>	Mode register bits that set the output drive impedance setting. Set the value according to the board simulation.
<b>ODT</b>	Mode register bits that set the ODT setting. Set the value according to the board simulation.
<b>AREF Protocol</b>	Mode register setting for refreshing memory content of a bank. Select <b>Multibank</b> to allow refresh 4 bank simultaneously. Select Bank Address Control to refresh a particular bank by setting the bank address.
<b>Write Protocol</b>	Mode register setting for write protocol. When multiple bank (dual bank or quad bank) is selected, identical data is written to multiple banks.

**Topology**

<b>Device width</b>	Specifies the number of memory devices used for width expansion.
---------------------	--

**Table 7-21: Memory Timing Parameters for RLDRAM 3 for UniPHY**

Parameter	Description
<b>Memory Device Timing</b>	
<b>Maximum memory clock frequency</b>	The maximum frequency at which the memory device can run.
<b>tDS (base)</b>	Base specification for data setup to DK/DK#. Set according to memory speed grade. Refer to memory data sheet.
<b>tDH (base)</b>	Base specification for data hold from DK/DK#. Set according to memory speed grade. Refer to memory data sheet.

Parameter	Description
<b>tQKQ_max</b>	QK/QK# clock edge to DQ data edge (in same group). Set according to memory speed grade. Refer to memory data sheet.
<b>tQH (% of CK)</b>	DQ output hold time from QK/QK#. Set according to memory speed grade. Refer to memory data sheet.
<b>tCKDK_max(% of CK)</b>	Clock to input data clock (max). Set according to memory speed grade. Refer to memory data sheet.
<b>tCKDK_min (% of CK)</b>	Clock to input data clock (min). Set according to memory speed grade. Refer to memory data sheet.
<b>tCKQK_max</b>	QK edge to clock edge skew (max). Set according to memory speed grade. Refer to memory data sheet.
<b>tIS (base)</b>	Base specification for address and control setup to CK. Set according to memory speed grade. Refer to memory data sheet.
<b>tIH (base)</b>	Base specification for address and control hold from CK. Set according to memory speed grade. Refer to memory data sheet.
<b>Controller Timing</b>	
<b>Read-to-Write NOP commands (min)</b>	Minimum number of no operation commands following a read command and before a write command. The value must be at least ((Burst Length/2) + RL - WL + 2). The value, along with other delay/skew parameters, are used by the "Bus Turnaround" timing analysis to determine if bus contention is an issue.  Set according to the controller specification.
<b>Write-to-Read NOP commands (min)</b>	Minimum number of no operation commands following a write command and before a read command. The value must be at least ((Burst Length/2) + WL - RL + 1). The value, along with other delay/skew parameters, are used by the "Bus Turnaround" timing analysis to determine if bus contention is an issue.  Set according to the controller specification.
<b>RLDRAM 3 Board Derate</b>	
<b>CK/CK# slew rate (differential)</b>	CK/CK# slew rate (differential).

Parameter	Description
<b>Address/Command slew rate</b>	Address and command slew rate.
<b>DK/DK# slew rate (Differential)</b>	DK/DK# slew rate (differential).
<b>DQ slew rate</b>	DQ slew rate.
<b>tIS</b>	Address/command setup time to CK.

## Board Settings

Use the **Board Settings** tab to model the board-level effects in the timing analysis.

The **Board Settings** tab allows you to specify the following settings:

- Setup and hold derating (For LPDDR2/DDR2/DDR3 SDRAM, RLDRAM 3 and RLDRAM II for UniPHY IP)
- Channel Signal Integrity
- Board skews (For UniPHY IP.)

**Note:** For accurate timing results, you must enter board settings parameters that are correct for your PCB.

The IP core supports single and multiple chip-select configurations. Altera has determined the effects on the output signaling of single-rank configurations for certain Altera boards, and included the channel uncertainties in the Quartus Prime timing models.

Because the Quartus Prime timing models hold channel uncertainties that are representative of specific Altera boards, you must determine the board-level effects of your board, including any additional channel uncertainty relative to Altera's reference design, and enter those values into the Board Settings panel in the parameter editor. You can use HyperLynx or a similar simulator to obtain values that are representative of your board.

For more information about how to include your board simulation results in the Quartus Prime software, refer to the following sections. For more information about how to assign pins using pin planners, refer to the design flow tutorials and design examples on the List of Designs Using Altera External Memory IP page of the Altera Wiki.

For more general information about timing deration methodology, refer to the *Timing Deration Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs* section in the *Analyzing Timing of Memory IP* chapter.

### Related Information

- [Timing Deration Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs](#) on page 9-41
- [Analyzing Timing of Memory IP](#)
- [List of Designs using Altera External Memory IP](#)

### Setup and Hold Derating for UniPHY IP

The slew rate of the output signals affects the setup and hold times of the memory device, and thus the write margin. You can specify the slew rate of the output signals to see their effect on the setup and hold

times of both the address and command signals and the DQ signals, or alternatively, you may want to specify the setup and hold times directly.

For RDIMMs, the slew rate is defined at the register on the RDIMM, instead of at the memory component. For LRDIMMs, the slew rate is defined at the buffer on the LRDIMM, instead of at the memory component.

**Note:** You should enter information derived during your PCB development process of prelayout (line) and postlayout (board) simulation.

The following table lists the setup and hold derating parameters.

**Table 7-22: Setup and Hold Derating Parameters**

Parameter	Description
<b>LPDDR2/DDR2/DDR3 SDRAM/RLDRAM 3</b>	
<b>Derating method</b>	Derating method. The default settings are based on Altera internal board simulation data. To obtain accurate timing analysis according to the condition of your board, Altera recommends that you perform board simulation and enter the slew rate in the Quartus Prime software to calculate the derated setup and hold time automatically or enter the derated setup and hold time directly.  For more information, refer to the “Timing Deration Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs” section in the <i>Analyzing Timing of Memory IP</i> chapter.
<b>CK/CK# slew rate (differential)</b>	CK/CK# slew rate (differential).
<b>Address/Command slew rate</b>	Address and command slew rate.
<b>DQS/DQS# slew rate (Differential)</b>	DQS and DQS# slew rate (differential).
<b>DQ slew rate</b>	DQ slew rate.
<b>tIS</b>	Address/command setup time to CK.
<b>tIH</b>	Address/command hold time from CK.
<b>tDS</b>	Data setup time to DQS.
<b>tDH</b>	Data hold time from DQS.
<b>RLDRAM II</b>	

Parameter	Description
<b>tAS Vref to CK/CK# Crossing</b>	For a given address/command and CK/CK# slew rate, the memory device data sheet provides a corresponding "tAS Vref to CK/CK# Crossing" value that can be used to determine the derated address/command setup time.
<b>tAS VIH MIN to CK/CK# Crossing</b>	For a given address/command and CK/CK# slew rate, the memory device data sheet provides a corresponding "tAS VIH MIN to CK/CK# Crossing" value that can be used to determine the derated address/command setup time.
<b>tAH CK/CK# Crossing to Vref</b>	For a given address/command and CK/CK# slew rate, the memory device data sheet provides a corresponding "tAH CK/CK# Crossing to Vref" value that can be used to determine the derated address/command hold time.
<b>tAH CK/CK# Crossing to VIH MIN</b>	For a given address/command and CK/CK# slew rate, the memory device data sheet provides a corresponding "tAH CK/CK# Crossing to VIH MIN" value that can be used to determine the derated address/command hold time.
<b>tDS Vref to CK/CK# Crossing</b>	For a given data and DK/DK# slew rate, the memory device data sheet provides a corresponding "tDS Vref to CK/CK# Crossing" value that can be used to determine the derated data setup time.
<b>tDS VIH MIN to CK/CK# Crossing</b>	For a given data and DK/DK# slew rate, the memory device data sheet provides a corresponding "tDS VIH MIN to CK/CK# Crossing" value that can be used to determine the derated data setup time.
<b>tDH CK/CK# Crossing to Vref</b>	For a given data and DK/DK# slew rate, the memory device data sheet provides a corresponding "tDH CK/CK# Crossing to Vref" value that can be used to determine the derated data hold time.
<b>tDH CK/CK# Crossing to VIH MIN</b>	For a given data and DK/DK# slew rate, the memory device data sheet provides a corresponding "tDH CK/CK# Crossing to VIH MIN" value that can be used to determine the derated data hold time.
<b>Derated tAS</b>	The derated address/command setup time is calculated automatically from the "tAS", the "tAS Vref to CK/CK# Crossing", and the "tAS VIH MIN to CK/CK# Crossing" parameters.

Parameter	Description
<b>Derated tAH</b>	The derated address/command hold time is calculated automatically from the "tAH", the "tAH CK/CK# Crossing to Vref", and the "tAH CK/CK# Crossing to VIH MIN" parameters.
<b>Derated tDS</b>	The derated data setup time is calculated automatically from the "tDS", the "tDS Vref to CK/CK# Crossing", and the "tDS VIH MIN to CK/CK# Crossing" parameters.
<b>Derated tDH</b>	The derated data hold time is calculated automatically from the "tDH", the "tDH CK/CK# Crossing to Vref", and the "tDH CK/CK# Crossing to VIH MIN" parameters.

**Related Information**

[Analyzing Timing of Memory IP](#) on page 9-1

**Intersymbol Interference Channel Signal Integrity for UniPHY IP**

Channel signal integrity is a measure of the distortion of the eye due to intersymbol interference or crosstalk or other effects.

Typically, when going from a single-rank configuration to a multi-rank configuration there is an increase in the channel loss, because there are multiple stubs causing reflections. Although the Quartus Prime timing models include some channel uncertainty, you must perform your own channel signal integrity simulations and enter the additional channel uncertainty, relative to the reference eye, into the parameter editor GUI.

For details about measuring channel loss parameters and entering channel signal integrity information into the parameter editor GUI, refer to the Altera Wiki: [http://www.alterawiki.com/wiki/Measuring\\_Channel\\_Signal\\_Integrity](http://www.alterawiki.com/wiki/Measuring_Channel_Signal_Integrity).

The following table lists intersymbol interference parameters.

**Table 7-23: ISI Parameters**

Parameter	Description
<b>Derating method</b>	Choose between default Altera settings (with specific Altera boards) or manually enter board simulation numbers obtained for your specific board.  This option is supported in LPDDR2/DDR2/DDR3 SDRAM only.

Parameter	Description
<b>Address and command eye reduction (setup)</b>	<p>The reduction in the eye diagram on the setup side (or left side of the eye) due to ISI on the address and command signals compared to a case when there is no ISI. (For single rank designs, ISI can be zero; in multirank designs, ISI is necessary for accurate timing analysis.)</p> <p>For more information about how to measure the ISI value for the address and command signals, refer to the “Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time” section in <i>Analyzing Timing of Memory IP</i>.</p>
<b>Address and command eye reduction (hold)</b>	<p>The reduction in the eye diagram on the hold side (or right side of the eye) due to ISI on the address and command signals compared to a case when there is no ISI.</p> <p>For more information about how to measure the ISI value for the address and command signals, refer to “Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time” section in <i>Analyzing Timing of Memory IP</i>.</p>
<b>DQ/ D eye reduction</b>	<p>The total reduction in the eye diagram due to ISI on DQ signals compared to a case when there is no ISI. Altera assumes that the ISI reduces the eye width symmetrically on the left and right side of the eye.</p> <p>For more information about how to measure the ISI value for the address and command signals, refer to “Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time” section in <i>Analyzing Timing of Memory IP</i>.</p>
<b>Delta DQS/Delta K/ Delta DK arrival time</b>	<p>The increase in variation on the range of arrival times of DQS compared to a case when there is no ISI. Altera assumes that the ISI causes DQS to further vary symmetrically to the left and to the right.</p> <p>For more information about how to measure the ISI value for the address and command signals, refer to “Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time” section in <i>Analyzing Timing of Memory IP</i>.</p>

#### Related Information

[Analyzing Timing of Memory IP](#) on page 9-1

#### Board Skews for UniPHY IP

PCB traces can have skews between them that can reduce timing margins. Furthermore, skews between different chip selects can further reduce the timing margin in multiple chip-select topologies.

The **Board Skews** section of the parameter editor allows you to enter parameters to compensate for these variations.

**Note:** You must ensure the timing margin reported in TimeQuest Report DDR is positive when the board skew parameters are correct for the PCB.

The following tables list the board skew parameters. For parameter equations containing delay values, delays should be measured as follows:

- Non-fly-by topology (Balanced Tree)
  - For discrete devices—all the delay (CK, Addr/Cmd, DQ and DQS) from the FPGA are right to every memory device
  - For UDIMMs—all the delay (CK, Addr/Cmd, DQ and DQS) from the FPGA to UDIMM connector for every memory device on the UDIMM. If UDIMM delay information is available, calculate delays to every memory device on the UDIMM.
  - For RDIMMs—the Addr/Cmd and CK delay are from the FPGA to the register on the RDIMM. The DQ and DQS delay are from FPGA to RDIMM connector for every memory device on the RDIMM.
  - For LRDIMMS—the delay from the FPGA to the register on the LRDIMM.
- Fly-by topology
  - For discrete devices—the Addr/Cmd and CK delay are from the FPGA to the first memory device. The DQ and DQS delay are from FPGA to every memory device.
  - For UDIMMs—the Addr/Cmd and CK delay are from the FPGA to the UDIMM connector. The DQ and DQS delay are from the FPGA to UDIMM connector for every memory device on the UDIMM.
  - For RDIMMs—the Addr/Cmd and CK delay are from the FPGA to the register on the RDIMM. The DQ and DQS delay are from FPGA to RDIMM connector for every memory device on the RDIMM.
  - For LRDIMMS—the delay from the FPGA to the buffer on the LRDIMM.

Equations apply to any given memory device, except when marked by the board or group qualifiers (*\_b* or *\_g*), where they apply to the particular device or group being iterated over.

Use the Board Skew Parameter Tool to help you calculate the board skews.

#### Related Information

##### [Board Skew Parameter Tool](#)

#### Board Skew Parameters for LPDDR2/DDR2/DDR3 SDRAM

The following table lists board skew parameters for LPDDR2, DDR2, and DDR3 interfaces.

**Table 7-24: Parameter Descriptions**

Parameter	Description
FPGA DQ/DQS Package Skews Deskewed on Board	Enable this parameter if you will deskew the FPGA package with your board traces on the DQ and DQS pins. This option increases the read capture and write margins. Enable this option when memory clock frequency is larger than 800 MHz. Enabling this option improves the read capture and write timing margin. You can also rely on the read capture and write timing margin in timing report to enable this option.  When this option is enabled, package skews are output on the DQ and DQS pins in the Pin-Out File (.pin) and package skew is not included in timing analysis. All of the other board delay and skew parameters related to DQ or DQS must consider the package and the board together. For more information, refer to <i>DDR2 and DDR3 Board Layout Guidelines</i> .

Parameter	Description
<b>Address/ Command Package Deskew</b>	<p>Enable this parameter if you will deskew the FPGA package with your board traces on the address and command pins. This option increases the address and command margins. Enable this option when memory clock frequency is larger than 800 MHz. Enabling this option will improve the address and command timing margin. You can also rely on the address and command margin in timing report to enable this option.</p> <p>When this option is enabled, package skews are output on the address and command pins in the Pin-Out File (.pin) and package skew is not included in timing analysis. All of the other board delay and skew parameters related to Address and Command must consider the package and the board together. For more information, refer to <i>DDR2 and DDR3 Board Layout Guidelines</i>.</p>
<b>Maximum CK delay to DIMM/ device</b>	<p>The delay of the longest CK trace from the FPGA to the memory device, whether on a DIMM or the same PCB as the FPGA is expressed by the following equation:</p> $\max_r[\max_n(CK_{n\_r} PathDelay)]$ <p>Where <math>n</math> is the number of memory clock and <math>r</math> is number rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clocks in each rank DIMM, the maximum CK delay is expressed by the following equation:</p> <p><math>\max(CK_1 PathDelay \text{ rank 1}, CK_2 Path Delay \text{ rank 1}, CK_1 Path Delay \text{ rank 2}, CK_2 Path Delay \text{ rank 2})</math></p>
<b>Maximum DQS delay to DIMM/ device</b>	<p>The delay of the longest DQS trace from the FPGA to the memory device, whether on a DIMM or the same PCB as the FPGA is expressed by the following equation:</p> $\max_r[\max_n(DQS_{n\_r} Path Delay)]$ <p>Where <math>n</math> is the number of DQS and <math>r</math> is number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 DQS in each rank DIMM, the maximum DQS delay is expressed by the following equation:</p> <p><math>\max(DQS_1 PathDelay \text{ rank 1}, DQS_2 Path Delay \text{ rank 1}, DQS_1 Path Delay \text{ rank 2}, DQS_2 Path Delay \text{ rank 2})</math></p>

Parameter	Description
<b>Minimum delay difference between CK and DQS</b>	<p>The minimum skew or smallest positive skew (or largest negative skew) between the CK signal and any DQS signal when arriving at the same DIMM/device over all DIMMs/devices is expressed by the following equation:</p> $\min_r \left[ \min_{n, m} \{ (CK_{n\_r} \text{Delay} - DQS_{m\_r} \text{Delay}) \} \right]$ <p>Where <math>n</math> is the number of memory clock, <math>m</math> is the number of DQS, and <math>r</math> is the number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clock and 4 DQS signals (two for each clock) for each rank DIMM, the minimum delay difference between CK and DQS is expressed by the following equation:</p> $\min \left\{ \begin{array}{l} (CK_{1\_1} \text{Delay} - DQS_{1\_1} \text{Delay}), (CK_{1\_1} \text{Delay} - DQS_{2\_1} \text{Delay}), (CK_{2\_1} \text{Delay} - DQS_{3\_1} \text{Delay}), (CK_{2\_1} \text{Delay} - DQS_{4\_1} \text{Delay}) \\ (CK_{1\_2} \text{Delay} - DQS_{1\_2} \text{Delay}), (CK_{1\_2} \text{Delay} - DQS_{2\_2} \text{Delay}), (CK_{2\_2} \text{Delay} - DQS_{3\_2} \text{Delay}), (CK_{2\_2} \text{Delay} - DQS_{4\_2} \text{Delay}) \end{array} \right\}$ <p>This parameter value affects the write leveling margin for DDR3 interfaces with leveling in multi-rank configurations. This parameter value also applies to non-leveling configurations of any number of ranks with the requirement that DQS must have positive margins in TimeQuest Report DDR.</p> <p>For multiple boards, the minimum skew between the CK signal and any DQS signal when arriving at the same DIMM over all DIMMs is expressed by the following equation, if you want to use the same design for several different boards:</p> $\min_b [ \min_g [ CK_{g-b} - DQS_{g-b} ] ]$ <p><b>Note:</b> If you are using a clamshell topology in a multirank/multi chip-select design with either DIMM or discrete devices, or using dual-die devices, the above calculations do not apply; you may use the default values in the GUI.</p>

Parameter	Description
<b>Maximum delay difference between CK and DQS</b>	<p>The maximum skew or smallest negative skew (or largest positive skew) between the CK signal and any DQS signal when arriving at the same DIMM/device over all DIMMs/devices is expressed by the following equation:</p> $\max_r \left[ \max_{n,m} \{ (CK_{n\_r} \text{Delay} - DQS_{m\_r} \text{Delay}) \} \right]$ <p>Where <math>n</math> is the number of memory clock, <math>m</math> is the number of DQS, and <math>r</math> is the number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clock and 4 DQS signals (two for each clock) for each rank DIMM, the maximum delay difference between CK and DQS is expressed by the following equation:</p> $\max \left\{ \begin{array}{l} (CK_{1\_1} \text{Delay} - DQS_{1\_1} \text{Delay}), (CK_{1\_1} \text{Delay} - DQS_{2\_1} \text{Delay}), (CK_{2\_1} \text{Delay} - DQS_{3\_1} \text{Delay}), (CK_{2\_1} \text{Delay} - DQS_{4\_1} \text{Delay}) \\ (CK_{1\_2} \text{Delay} - DQS_{1\_2} \text{Delay}), (CK_{1\_2} \text{Delay} - DQS_{2\_2} \text{Delay}), (CK_{2\_2} \text{Delay} - DQS_{3\_2} \text{Delay}), (CK_{2\_2} \text{Delay} - DQS_{4\_2} \text{Delay}) \end{array} \right\}$ <p>This value affects the write Leveling margin for DDR3 interfaces with leveling in multi-rank configurations. This parameter value also applies to non-leveling configurations of any number of ranks with the requirement that DQS must have positive margins in TimeQuest Report DDR.</p> <p>For multiple boards, the maximum skew (or largest positive skew) between the CK signal and any DQS signal when arriving at the same DIMM over all DIMMs is expressed by the following equation, if you want to use the same design for several different boards:</p> $\max_b [ \max_g [ CK_{g\_b} - DQS_{g\_b} ] ]$ <p><b>Note:</b> If you are using a clamshell topology in a multirank/multi chip-select design with either DIMM or discrete devices, or using dual-die devices, the above calculations do not apply; you may use the default values in the GUI.</p>
<b>Maximum skew within DQS group</b>	<p>The largest skew among DQ and DM signals in a DQS group. This value affects the read capture and write margins for DDR2 and DDR3 SDRAM interfaces in all configurations (single or multiple chip-select, DIMM or component).</p> <p>For multiple boards, the largest skew between DQ and DM signals in a DQS group is expressed by the following equation:</p> $\max_b [ \max_g [ \maxDQ_{g\_b} - \minDQ_{g\_b} ] ]$

Parameter	Description
<b>Maximum skew between DQS groups</b>	<p>The largest skew between DQS signals in different DQS groups. This value affects the resynchronization margin in memory interfaces without leveling such as DDR2 SDRAM and discrete-device DDR3 SDRAM in both single- or multiple chip-select configurations. For protocols or families that do not have read resynchronization analysis, this parameter has no effect.</p> <p>For multiple boards, the largest skew between DQS signals in different DQS groups is expressed by the following equation, if you want to use the same design for several different boards:</p> $\max_b \left[ \max_g [DQS_{g-b}] - \min_b \left[ \min_g [DQS_{g-b}] \right] \right]$
<b>Average delay difference between DQ and DQS</b>	<p>The average delay difference between each DQ signal and the DQS signal, calculated by averaging the longest and smallest DQ signal delay values minus the delay of DQS. The average delay difference between DQ and DQS is expressed by the following equation:</p> $\sum_{n=1}^n \left[ \left( \frac{\text{Longest DQ Path Delay in } DQS_n \text{ group} + \text{Shortest DQ Path Delay in } DQS_n \text{ group}}{2} \right) - DQS_n \text{ PathDelay} \right]$ <p>where n is the number of DQS groups. For multi-rank or multiple CS configuration, the equation is:</p> $\sum_{r=1}^r \left[ \text{Average delay differnt between DQ and DQS in rank r} \right]$
<b>Maximum skew within address and command bus</b>	<p>The largest skew between the address and command signals for a single board is expressed by the following equation:</p> $0.5[(MaxACdelay - MinCKdelay) - (MinACdelay - MaxCKdelay)]$ <p>For multiple boards, the largest skew between the address and command signals is expressed by the following equation, if you want to use the same design for several different boards:</p> $\frac{\max_b [(MaxAC_b - MinCK_b) - (MinAC_b - MaxCK_b)]}{2}$

Parameter	Description
Average delay difference between address and command and CK	<p>A value equal to the average of the longest and smallest address and command signal delay values, minus the delay of the CK signal. The value can be positive or negative. Positive values represent address and command signals that are longer than CK signals; negative values represent address and command signals that are shorter than CK signals. The average delay difference between address and command and CK is expressed by the following equation:</p> $\sum_{n=1}^n \left[ \left( \frac{\text{Longest AC Path Delay} + \text{Shortest AC Path Delay}}{2} \right) - CK_n \text{PathDelay} \right]$ <p>where n is the number of memory clocks. For multi-rank or multiple CS configuration, the equation is:</p> $\sum_{r=1}^r \left[ \text{Average delay differnt between AC and CK in rank r} \right]$ <p>The Quartus Prime software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins for DDR2 and DDR3 SDRAM interfaces. You should derive this value through board simulation.</p> <p>For multiple boards, the average delay difference between address and command and CK is expressed by the following equation, if you want to use the same design for several different boards:</p> $Avg_b^{boards} \left[ \left( \frac{\text{MaxAC}_b + \text{MinAC}_b}{2} \right) - \left( \frac{\text{MaxCK}_b + \text{MinCK}_b}{2} \right) \right]$

**Related Information**[DDR2, DDR3, and DDR4 SDRAM Board Design Guidelines](#) on page 2-1**Board Skew Parameters for QDR II and QDR II+**

The following table lists board skew parameters for QDR II and QDR II+ interfaces.

**Table 7-25: Parameter Descriptions**

Parameter	Description
Maximum delay difference between devices	<p>The maximum delay difference of data signals between devices is expressed by the following equation:</p> $Abs \left[ \left( \frac{\text{Longest device 1 delay} - \text{Shortest device 2 delay}}{2} \right) - \left( \frac{\text{Longest device 2 delay} - \text{Shortest device 1 delay}}{2} \right) \right]$ <p>For example, in a two-device configuration there is greater propagation delay for data signals going to and returning from the furthest device relative to the nearest device. This parameter is applicable for depth expansion. Set the value to 0 for non-depth expansion design.</p>

Parameter	Description
<b>Maximum skew within write data group (ie, K group)</b>	The maximum skew between D and BWS signals referenced by a common K signal.
<b>Maximum skew within read data group (ie, CQ group)</b>	The maximum skew between Q signals referenced by a common CQ signal.
<b>Maximum skew between CQ groups</b>	The maximum skew between CQ signals of different read data groups. Set the value to 0 for non-depth expansion designs.
<b>Maximum skew within address/command bus</b>	The maximum skew between the address/command signals. $0.5[(MaxACdelay - MinCKdelay) - (MinACdelay - MaxCKdelay)]$
<b>Average delay difference between address/command and K</b>	A value equal to the average of the longest and smallest address/command signal delay values, minus the delay of the K signal. The value can be positive or negative. The average delay difference between the address and command and K is expressed by the following equation: $\sum_{n=1}^n \left[ \left( \frac{\text{Longest AC Path Delay} + \text{Shortest AC Path Delay}}{2} \right) - K_n \text{PathDelay} \right]$ where n is the number of K clocks.
<b>Average delay difference between write data signals and K</b>	A value equal to the average of the longest and smallest write data signal delay values, minus the delay of the K signal. Write data signals include the D and BWS signals. The value can be positive or negative. The average delay difference between D and K is expressed by the following equation: $\sum_{n=1}^n \left[ \left( \frac{\text{Longest D Path Delay in } K_n \text{ group} + \text{Shortest D Path Delay in } K_n \text{ group}}{2} \right) - K_n \text{PathDelay} \right]$ where n is the number of DQS groups.

Parameter	Description
<b>Average delay difference between read data signals and CQ</b>	<p>A value equal to the average of the longest and smallest read data signal delay values, minus the delay of the CQ signal. The value can be positive or negative.</p> <p>The average delay difference between Q and CQ is expressed by the following equation:</p> $\sum_{n=1}^{n=n} \left[ \left( \frac{\text{Longest } Q \text{ Path Delay in } CQ_n \text{ group} + \text{Shortest } Q \text{ Path Delay in } CQ_n \text{ group}}{2} \right) - CQ_n \text{ PathDelay} \right]$ <p>where n is the number of CQ groups.</p>

**Board Skew parameters for RLDRAM II and RLDRAM 3**

The following table lists board skew parameters for RLDRAM II and RLDRAM 3 interfaces.

**Table 7-26: Parameter Descriptions**

Parameter	Description
<b>Maximum CK delay to device</b>	<p>The delay of the longest CK trace from the FPGA to any device/DIMM is expressed by the following equation:</p> $\max_n(CK_n \text{ PathDelay})$ <p>where n is the number of memory clocks. For example, the maximum CK delay for two pairs of memory clocks is expressed by the following equation:</p> $\max_2(CK_1 \text{ PathDelay}, CK_2 \text{ PathDelay})$
<b>Maximum DK delay to device</b>	<p>The delay of the longest DK trace from the FPGA to any device/DIMM is expressed by the following equation:</p> $\max_n(DK_n \text{ PathDelay})$ <p>where n is the number of DK. For example, the maximum DK delay for two DK is expressed by the following equation:</p> $\max_2(DK_1 \text{ PathDelay}, DK_2 \text{ PathDelay})$

Parameter	Description
<b>Minimum delay difference between CK and DK</b>	<p>The minimum delay difference between the CK signal and any DK signal when arriving at the memory device(s). The value is equal to the minimum delay of the CK signal minus the maximum delay of the DK signal. The value can be positive or negative.</p> <p>The minimum delay difference between CK and DK is expressed by the following equations:</p> $\min_{n,m} (CK_n \text{PathDelay} - DK_m \text{PathDelay})$ <p>where n is the number of memory clocks and m is the number of DK. For example, the minimum delay difference between CK and DK for two pairs of memory clocks and four DK signals (two DK signals for each clock) is expressed by the following equation:</p> $\min_{2,2} \{(Ck_1 \text{Delay} - DK_1 \text{Delay}), (Ck_1 \text{Delay} - DK_2 \text{Delay}), (Ck_2 \text{Delay} - DK_3 \text{Delay}), (Ck_2 \text{Delay} - DK_4 \text{Delay})\}$
<b>Maximum delay difference between CK and DK</b>	<p>The maximum delay difference between the CK signal and any DK signal when arriving at the memory device(s). The value is equal to the maximum delay of the CK signal minus the minimum delay of the DK signal. The value can be positive or negative.</p> <p>The maximum delay difference between CK and DK is expressed by the following equations:</p> $\max_{n,m} (CK_n \text{PathDelay} - DK_m \text{PathDelay})$ <p>where n is the number of memory clocks and m is the number of DK. For example, the maximum delay difference between CK and DK for two pairs of memory clocks and four DK signals (two DK signals for each clock) is expressed by the following equation:</p> $\max_{2,2} \{(Ck_1 \text{Delay} - DK_1 \text{Delay}), (Ck_1 \text{Delay} - DK_2 \text{Delay}), (Ck_2 \text{Delay} - DK_3 \text{Delay}), (Ck_2 \text{Delay} - DK_4 \text{Delay})\}$

Parameter	Description
<b>Maximum delay difference between devices</b>	The maximum delay difference of data signals between devices is expressed by the following equation: $Abs\left[\left(\frac{Longest\ device\ 1\ delay - Shortest\ device\ 2\ delay}{2}\right) - \left(\frac{Longest\ device\ 2\ delay - Shortest\ device\ 1\ delay}{2}\right)\right]$ For example, in a two-device configuration there is greater propagation delay for data signals going to and returning from the furthest device relative to the nearest device. This parameter is applicable for depth expansion. Set the value to 0 for non-depth expansion design.
<b>Maximum skew within QK group</b>	The maximum skew between the DQ signals referenced by a common QK signal.
<b>Maximum skew between QK groups</b>	The maximum skew between QK signals of different data groups.
<b>Maximum skew within address/command bus</b>	The maximum skew between the address/command signals. $0.5[(MaxACdelay - MinCKdelay) - (MinACdelay - MaxCKdelay)]$
<b>Average delay difference between address/command and CK</b>	A value equal to the average of the longest and smallest address/command signal delay values, minus the delay of the CK signal. The value can be positive or negative. The average delay difference between the address and command and CK is expressed by the following equation: $\sum_{n=1}^n \left[ \left( \frac{Longest\ AC\ Path\ Delay + Shortest\ AC\ Path\ Delay}{2} \right) - CK_n\ PathDelay \right]$ where n is the number of memory clocks.

Parameter	Description
<b>Average delay difference between write data signals and DK</b>	<p>A value equal to the average of the longest and smallest write data signal delay values, minus the delay of the DK signal. Write data signals include the DQ and DM signals. The value can be positive or negative.</p> <p>The average delay difference between DQ and DK is expressed by the following equation:</p> $\sum_{n=1}^n \left[ \left( \frac{\text{Longest DQ Path Delay in } DK_n \text{ group} + \text{Shortest DQ Path Delay in } DK_n \text{ group}}{2} \right) - DK_n \text{ PathDelay} \right]$ <p>where n is the number of DK groups.</p>
<b>Average delay difference between read data signals and QK</b>	<p>A value equal to the average of the longest and smallest read data signal delay values, minus the delay of the QK signal. The value can be positive or negative.</p> <p>The average delay difference between DQ and QK is expressed by the following equation:</p> $\sum_{n=1}^n \left[ \left( \frac{\text{Longest DQ Path Delay in } QK_n \text{ group} + \text{Shortest DQ Path Delay in } QK_n \text{ group}}{2} \right) - QK_n \text{ PathDelay} \right]$ <p>where n is the number of QK groups.</p>

## Controller Settings for UniPHY IP

Use the Controller Settings tab to apply the controller settings suitable for your design.

**Note:** This section describes parameters for the High Performance Controller II (HPC II) with advanced features first introduced in version 11.0 for designs generated in version 11.0 or later. Designs created in earlier versions and regenerated in version 11.0 or later do not inherit the new advanced features; for information on parameters for HPC II without the advanced features, refer to the External Memory Interface Handbook for Quartus II version 10.1, available on the *Literature: External Memory Interfaces* page of the Altera website.

**Table 7-27: Controller Settings for LPDDR2/DDR2/DDR3 SDRAM**

Parameter	Description
<b>Avalon Interface</b>	<p><b>Generate power-of-2 bus widths for SOPC Builder</b></p> <p>Rounds down the Avalon-MM side data bus to the nearest power of 2. You must enable this option for Qsys systems.</p> <p>This option is enabled, the Avalon data buses are truncated to 256 bits wide. One Avalon read-write transaction of 256 bit width maps to four memory beat transactions,</p>

Parameter	Description
	each of 72 bits (8 MSB bits are zero, while 64 LSB bits carry useful content). The four memory beats may comprise an entire burst length-of-4 transaction, or part of a burst-length-of-8 transaction.
<b>Generate SOPC Builder compatible resets</b>	This option is not required when using the IP Catalog or Qsys.
<b>Maximum Avalon-MM burst length</b>	Specifies the maximum burst length on the Avalon-MM bus. Affects the <code>AVL_SIZE_WIDTH</code> parameter.
<b>Enable Avalon-MM byte-enable signal</b>	When you turn on this option, the controller adds the byte enable signal ( <code>avl_be</code> ) for the Avalon-MM bus to control the data mask ( <code>mem_dm</code> ) pins going to the memory interface. You must also turn on <b>Enable DM pins</b> if you are turning on this option.  When you turn off this option, the byte enable signal ( <code>avl_be</code> ) is not enabled for the Avalon-MM bus, and by default all bytes are enabled. However, if you turn on <b>Enable DM pins</b> with this option turned off, all write words are written.
<b>Avalon interface address width</b>	The address width on the Avalon-MM interface.
<b>Avalon interface data width</b>	The data width on the Avalon-MM interface.

	Parameter	Description
<b>Low Power Mode</b>	<b>Enable self-refresh controls</b>	Enables the self-refresh signals on the controller top-level design. These controls allow you to control when the memory is placed into self-refresh mode.
	<b>Enable Deep Power-Down Controls</b>	Enables the Deep Power-Down signals on the controller top level. These signals control when the memory is placed in Deep Power-Down mode.  This parameter is available only for LPDDR2 SDRAM.
	<b>Enable auto-power down</b>	Allows the controller to automatically place the memory into (Precharge) power-down mode after a specified number of idle cycles. Specifies the number of idle cycles after which the controller powers down the memory in the auto-power down cycles parameter.
	<b>Auto power-down cycles</b>	The number of idle controller clock cycles after which the controller automatically powers down the memory. The legal range is from 1 to 65,535 controller clock cycles.
<b>Efficiency</b>	<b>Enable user auto refresh controls</b>	Enables the user auto-refresh control signals on the controller top level. These controller signals allow you to control when the controller issues memory autorefresh commands.
	<b>Enable auto precharge control</b>	Enables the autoprecharge control on the controller top level. Asserting the autoprecharge control signal while requesting a read or write burst allows you to specify whether the controller should close (autoprecharge) the currently open page at the end of the read or write burst.

Parameter	Description
<b>Local-to-memory address mapping</b>	Allows you to control the mapping between the address bits on the Avalon-MM interface and the chip, row, bank, and column bits on the memory.  Select <b>Chip-Row-Bank-Col</b> to improve efficiency with sequential traffic.  Select <b>Chip-Bank-Row-Col</b> to improve efficiency with random traffic.  Select <b>Row-Chip-Bank-Col</b> to improve efficiency with multiple chip select and sequential traffic.
<b>Command queue look ahead depth</b>	Selects a look-ahead depth value to control how many read or writes requests the look-ahead bank management logic examines. Larger numbers are likely to increase the efficiency of the bank management, but at the cost of higher resource usage. Smaller values may be less efficient, but also use fewer resources. The valid range is from 1 to 16.
<b>Enable reordering</b>	Allows the controller to perform command and data reordering that reduces bus turnaround time and row/bank switching time to improve controller efficiency.
<b>Starvation limit for each command</b>	Specifies the number of commands that can be served before a waiting command is served. The valid range is from 1 to 63.
<b>Configuration, Status, and Error Handling</b>	<b>Enable Configuration and Status Register Interface</b>  Enables run-time configuration and status interface for the memory controller. This option adds an additional Avalon-MM slave port to the memory controller top level, which you can use to change or read out the memory timing parameters, memory address sizes, mode register settings and controller status. If Error Detection and Correction Logic is enabled, the same slave port also allows you to control and retrieve the status of this logic.

Parameter	Description
	<p><b>CSR port host interface</b></p> <p>Specifies the type of connection to the CSR port. The port can be exported, internally connected to a JTAG Avalon Master, or both.</p> <p>Select <b>Internal (JTAG)</b> to connect the CSR port to a JTAG Avalon Master.</p> <p>Select <b>Avalon-MM Slave</b> to export the CSR port.</p> <p>Select <b>Shared</b> to export and connect the CSR port to a JTAG Avalon Master.</p>
	<p><b>Enable error detection and correction logic</b></p> <p>Enables ECC for single-bit error correction and double-bit error detection. Your memory interface must be a multiple of 16, 24, 40, or 72 bits wide to use ECC.</p>
	<p><b>Enable auto error correction</b></p> <p>Allows the controller to perform auto correction when a single-bit error is detected by the ECC logic.</p>
<b>Multiple Port Front End</b>	<p><b>Export bonding port</b></p> <p>Turn on this option to export bonding interface for wider avalon data width with two controllers. Bonding ports are exported to the top level.</p>
	<p><b>Number of ports</b></p> <p>Specifies the number of Avalon-MM Slave ports to be exported. The number of ports depends on the width and the type of port you selected. There are four 64-bit read FIFOs and four 64-bit write FIFOs in the multi-port front-end (MPFE) component. For example, If you select 256 bits width and bidirectional slave port, all the FIFOs are fully utilized, therefore you can only select one port.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>

Parameter	Description
<b>Width</b>	<p>Specifies the local data width for each Avalon-MM Slave port. The width depends on the type of slave port and also the number of ports selected. This is due to the limitation of the FIFO counts in the MPFE. There are four 64-bit read FIFOs and four 64-bit write FIFOs in the MPFE. For example, if you select one bidirectional slave port, you can select up to 256 bits to utilize all the read and write FIFOs.</p> <p>As a general guideline to choosing an optimum port width for your half-rate or quarter-rate design, apply the following equation:</p> $\text{port width} = 2 \times \text{DQ width} \times \text{Interface width multiplier}$ <p>where the interface width multiplier is 2 for half-rate interfaces and 4 for quarter-rate interfaces.</p>
<b>Priority</b>	Specifies the absolute priority for each Avalon-MM Slave port. Any transaction from the port with higher priority number will be served before transactions from the port with lower priority number.
<b>Weight</b>	Specifies the relative priority for each Avalon-MM Slave port. When there are two or more ports having the same absolute priority, the transaction from the port with higher (bigger number) relative weight will be served first. You can set the weight from a range of 0 to 32.
<b>Type</b>	Specifies the type of Avalon MM slave port to either a bidirectional port, read only port or write only port.

**Table 7-28: Controller Settings for QDR II/QDR II+ SRAM and RLDRAM II**

Parameter	Description
<b>Generate power-of-2 data bus widths for SOPC Builder</b>	Rounds down the Avalon-MM side data bus to the nearest power of 2. You must enable this option for Qsys systems.

Parameter	Description
<b>Generate SOPC Builder compatible resets</b>	This option is not required when using the IP Catalog or Qsys.
<b>Maximum Avalon-MM burst length</b>	Specifies the maximum burst length on the Avalon-MM bus.
<b>Enable Avalon-MM byte-enable signal</b>	<p>When you turn on this option, the controller adds a byte-enable signal (<code>avl_be_w</code>) for the Avalon-MM bus, in which controls the <code>bws_n</code> signal on the memory side to mask bytes during write operations.</p> <p>When you turn off this option, the <code>avl_be_w</code> signal is not available and the controller will always drive the memory <code>bws_n</code> signal so as to not mask any bytes during write operations.</p>
<b>Avalon interface address width</b>	Specifies the address width on the Avalon-MM interface.
<b>Avalon interface data width</b>	Specifies the data width on the Avalon-MM interface.
<b>Reduce controller latency by</b>	<p>Specifies the number of clock cycles by which to reduce controller latency.</p> <p>Lower controller latency results in lower resource usage and <math>f_{MAX}</math> while higher latency results in higher resource usage and <math>f_{MAX}</math>.</p>
<b>Enable user refresh</b>	<p>Enables user-controlled refresh. Refresh signals will have priority over read/write requests.</p> <p>This option is available for RLDRAM II only.</p>
<b>Enable error detection parity</b>	<p>Enables per-byte parity protection.</p> <p>This option is available for RLDRAM II only</p>

#### Related Information

[Literature: External Memory Interfaces](#)

## Diagnostics for UniPHY IP

The **Diagnostics** tab allows you to set parameters for certain diagnostic functions.

The following table describes parameters for simulation.

**Table 7-29: Simulation Options**

Parameter	Description	
<b>Simulation Options</b>		
<b>Auto-calibration mode</b>	<p>Specifies whether you want to improve simulation performance by reducing calibration. There is no change to the generated RTL. The following autocalibration modes are available:</p> <ul style="list-style-type: none"><li>• <b>Skip calibration</b>—provides the fastest simulation. It loads the settings calculated from the memory configuration and enters user mode.</li><li>• <b>Quick calibration</b>—calibrates (without centering) one bit per group before entering user mode.</li><li>• <b>Full calibration</b>—calibrates the same as in hardware, and includes all phases, delay sweeps, and centering on every data bit. You can use timing annotated memory models. Be aware that full calibration can take hours or days to complete.</li></ul> <p>To perform proper PHY simulation, select <b>Quick calibration</b> or <b>Full calibration</b>. For more information, refer to the “Simulation Options” section in the <i>Simulating Memory IP</i> chapter.</p> <p>For QDR II, QDR II+ SRAM, and RLDRAM II, the Nios II-based sequencer must be selected to enable the auto calibration modes selection.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>	
<b>Skip memory initialization delays</b>	<p>When you turn on this option, required delays between specific memory initialization commands are skipped to speed up simulation.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>	
<b>Enable verbose memory model output</b>	<p>Turn on this option to display more detailed information about each memory access during simulation.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>	
<b>Enable support for Nios II ModelSim® flow in Eclipse</b>	<p>Initializes the memory interface for use with the <b>Run as Nios II ModelSim</b> flow with Eclipse.</p> <p>This parameter is not available for QDR II and QDR II+ SRAM.</p> <p><b>Note:</b> This parameter is not available for MAX 10 devices.</p>	
<b>Debug Options</b>		
<b>Debug level</b>	Specifies the debug level of the memory interface.	
<b>Efficiency Monitor and Protocol Checker Settings</b>		

Parameter	Description
<b>Enable the Efficiency Monitor and Protocol Checker on the Controller Avalon Interface</b>	Enables efficiency monitor and protocol checker block on the controller Avalon interface.  This option is not available for QDR II and QDR II+ SRAM, or for the MAX 10 device family, or for Arria V or Cyclone V designs using the Hard Memory Controller.

## Arria 10 External Memory Interface IP

This section contains information about parameterizing Arria 10 External Memory Interface IP.

### Qsys Interfaces

The interfaces in the Arria 10 External Memory Interface IP each have signals that can be connected in Qsys. The following tables list the signals available for each interface and provide a description and guidance on how to connect those interfaces.

Listed interfaces and signals are available in all configurations unless stated otherwise in the description column. For Arria 10 External Memory Interface for HPS, the `global_reset_reset_sink`, `pll_ref_clk_clock_sink`, `hps_emif_conduit_end`, `oct_conduit_end` and `mem_conduit_end` interfaces are the only available interfaces regardless of your configuration.

### Arria 10 External Memory Interface IP Interfaces

**Table 7-30: Interface: afi\_clk\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
afi_clk	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3, RLDRAM 3, QDR IV</li> <li>Hard PHY only</li> </ul>	<p>The Altera PHY Interface (AFI) clock output signal. The clock frequency in relation to the memory clock frequency depends on the <b>Clock rate of user logic</b> value set in the parameter editor.</p> <p>Connect this interface to the (clock input) conduit of the custom AFI-based memory controller connected to the <code>afi_conduit_end</code> or any user logic block that requires the generated clock frequency.</p>

**Table 7-31: Interface: afi\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
afi_cal_success	Output		
afi_cal_fail	Output		
afi_cal_req	Input		
afi_rlat	Output		
afi_wlat	Output		
afi_addr	Input		
afi_RST_n	Input		
afi_wdata_valid	Input		
afi_wdata	Input		
afi_rdata_en_full	Input		
afi_rdata	Output		
afi_rdata_valid	Output		
afi_rrank	Input		
afi_wrrank	Input		
afi_BA	Input	<ul style="list-style-type: none"> <li>DDR3, DDR4, RLDRAM 3</li> <li>Hard PHY only</li> </ul>	The Altera PHY Interface (AFI) signals between the external memory interface IP and the custom AFI-based memory controller.
afi_CS_n	Input	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3, RLDRAM 3</li> <li>Hard PHY only</li> </ul>	Connect this interface to the AFI conduit of the custom AFI-based memory controller.
afi_CKE	Input		
afi_ODT	Input		
afi_DQS_burst	Input	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3</li> <li>Hard PHY only</li> </ul>	
afi_AP	Input		
afi_PE_n	Output		
afi_AINV	Input		
afi_LD_n	Input		
afi_RW_n	Input	<ul style="list-style-type: none"> <li>QDR IV</li> <li>Hard PHY only</li> </ul>	
afi_CFG_n	Input		
afi_LBK0_n	Input		
afi_LBK1_n	Input		

Signals in Interface	Direction	Availability	Description
afi_rdata_dinv	Output		The Altera PHY Interface (AFI) signals between the external memory interface IP and the custom AFI-based memory controller.
afi_wdata_dinv	Input	<ul style="list-style-type: none"> <li>• QDR IV</li> <li>• Hard PHY only</li> </ul>	Connect this interface to the AFI conduit of the custom AFI-based memory controller.
afi_we_n	Input	<ul style="list-style-type: none"> <li>• DDR3, RLDRAM 3</li> <li>• Hard PHY only</li> </ul>	
afi_dm	Input	<ul style="list-style-type: none"> <li>• DDR3, LPDDR3, RLDRAM 3</li> <li>• Hard PHY only</li> <li>• <b>Enable DM pins=True</b></li> </ul>	
afi_ras_n	Input	<ul style="list-style-type: none"> <li>• DDR3</li> </ul>	
afi_cas_n	Input	<ul style="list-style-type: none"> <li>• Hard PHY only</li> </ul>	
afi_rm	Input	<ul style="list-style-type: none"> <li>• DDR3</li> <li>• Hard PHY only</li> <li>• RDIMM/LRDIMM with <b>Number of rank multiplication pins &gt; 0</b></li> </ul>	The Altera PHY Interface (AFI) signals between the external memory interface IP and the custom AFI-based memory controller.
afi_par	Input	<ul style="list-style-type: none"> <li>• DDR3</li> <li>• Hard PHY only</li> <li>• RDIMM/LRDIMM</li> <li>• DDR4</li> <li>• Hard PHY only</li> <li>• <b>Enable alert_n/par pins = True</b></li> </ul>	Connect this interface to the AFI conduit of the custom AFI-based memory controller.  For more information, refer to the <a href="#">AFI 4.0 Specification</a> .
afi_bg	Input	<ul style="list-style-type: none"> <li>• DDR4</li> </ul>	
afi_act_n	Input	<ul style="list-style-type: none"> <li>• Hard PHY only</li> </ul>	
afi_dm_n	Input	<ul style="list-style-type: none"> <li>• DDR4</li> <li>• Hard PHY only</li> <li>• <b>Enable DM pins=True</b></li> </ul>	
afi_ref_n	Input	<ul style="list-style-type: none"> <li>• RLDRAM 3</li> <li>• Hard PHY only</li> </ul>	

**Table 7-32: Interface: afi\_half\_clk\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
afi_half_clk	Output	<ul style="list-style-type: none"><li>DDR3, DDR4, LPDDR3, RLDRAM 3, QDR IV</li><li>Hard PHY only</li></ul>	<p>The Altera PHY Interface (AFI) half clock output signal. The clock runs at half the frequency of the AFI clock (<code>afi_clk</code> clock).</p> <p>Connect this interface to the clock input conduit of the user logic block that needs to be clocked at the generated clock frequency.</p>

**Table 7-33: Interface: afi\_reset\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
afi_reset_n	Output	<ul style="list-style-type: none"><li>DDR3, DDR4, LPDDR3, RLDRAM 3, QDR IV</li><li>Hard PHY only</li></ul>	<p>The Altera PHY Interface (AFI) reset output signal. Asserted when the PLL becomes unlocked or when the PHY is reset. Asynchronous assertion and synchronous deassertion.</p> <p>Connect this interface to the reset input conduit of the custom AFI-based memory controller connected to the <code>afi_conduit_end</code> and all the user logic blocks that are under the AFI clock domain <code>afi_clk</code> or <code>afi_half_clk</code> clock).</p>

**Table 7-34: Interface: cal\_debug\_avalon\_slave**

Interface type: Avalon Memory-Mapped Slave

Signals in Interface	Direction	Availability	Description
cal_debug_waitrequest	Output		The Avalon-MM signals between the external memory interface IP and the external memory interface Debug Component.
cal_debug_read	Input		
cal_debug_write	Input		
cal_debug_addr	Input		
cal_debug_read_data	Output		
cal_debug_write_data	Input		
cal_debug_byteneable	Input		
cal_debug_read_data_valid	Output	<ul style="list-style-type: none"> <li>• EMIF Debug Toolkit</li> <li>• On-Chip Debug Port=Export</li> </ul>	<p>Connect this interface to the (<code>to_ioaux</code>) Avalon-MM master of the <b>Arria 10 EMIF Debug Component IP</b> or to (<code>cal_debug_out_avalon_master</code>) Avalon-MM master of the other external memory interface IP that has exported the interface. If you are not using the Altera EMIF Debug Toolkit, connect this interface to the Avalon-MM master of the custom debug logic.</p> <p>When in daisy-chaining mode, ensure one of the connected Avalon masters is either the <b>Arria 10 EMIF Debug Component IP</b> or the external memory interface IP with <b>EMIF Debug Toolkit/On-Chip Debug Port</b> set to <b>Add EMIF Debug Interface</b>.</p>

**Table 7-35: Interface: cal\_debug\_clk\_clock\_sink**

Interface type: Clock Input

Signals in Interface	Direction	Availability	Description
cal_debug_clk	Input	<ul style="list-style-type: none"> <li>• EMIF Debug Toolkit / On-Chip Debug Port=Export</li> </ul>	<p>The calibration debug clock input signal.</p> <p>Connect this interface to the (<code>avl_clk_out</code>) clock output of the <b>Arria 10 EMIF Debug Component IP</b> or to (<code>cal_debug_out_clk_clock_source</code>) clock input of the other external memory interface IP, depending on which IP the <code>cal_debug_avalon_slave</code> interface is connecting to. If you are not using the Altera EMIF Debug Toolkit, connect this interface to the clock output of the custom debug logic.</p>

**Table 7-36: Interface: cal\_debug\_out\_avalon\_master**

Interface type: Avalon Memory-Mapped Master

Signals in Interface	Direction	Availability	Description
cal_debug_out_waitrequest	Input	<ul style="list-style-type: none"> <li>• <b>EMIF Debug Toolkit / On-Chip Debug Port=Export</b></li> <li>• Add EMIF Debug Interface with <b>Enable Daisy-Chaining for EMIF Debug Toolkit/ On-Chip Debug Port=True</b></li> </ul>	<p>The Avalon-MM signals between the external memory interface IP and the other external memory interface IP.</p> <p>Connect this interface to the (<code>cal_debug_avalon_slave</code>) Avalon-MM Master of the external memory interface IP that has exported the interface .</p>
cal_debug_out_read	Output		
cal_debug_out_write	Output		
cal_debug_out_addr	Output		
cal_debug_out_read_data	Input		
cal_debug_out_write_data	Output		
cal_debug_out_bytewable	Output		
cal_debug_out_read_data_valid	Input		

**Table 7-37: Interface: cal\_debug\_out\_clk\_clock\_source**

Interface type: Clock Output

Signals in Interface	Direction	Availability	Description
cal_debug_out_clk	Output	<ul style="list-style-type: none"> <li>• <b>EMIF Debug Toolkit / On-Chip Debug Port=Export</b></li> <li>• Add EMIF Debug Interface with <b>Enable Daisy-Chaining for EMIF Debug Toolkit/ On-Chip Debug Port=True</b></li> </ul>	<p>The calibration debug clock output signal.</p> <p>For <b>EMIF Debug Toolkit/On-Chip Debug Port=Export</b> with <b>Enable Daisy-Chaining for EMIF Debug Toolkit/ On-Chip Debug Port=True</b>, the clock frequency follows the <code>cal_debug_clk</code> frequency. Otherwise, the clock frequency in relation to the memory clock frequency depends on the <b>Clock rate of user logic</b> value set in the parameter editor.</p> <p>Connect this interface to the (<code>cal_debug_out_reset_reset_source</code>) clock input of the other external memory interface IP where the <code>cal_debug_avalon_master</code> interface is being connected to or to any user logic block that needs to be clocked at the generated clock frequency.</p>

**Table 7-38: Interface: cal\_debug\_out\_reset\_reset\_source**

Interface type: Reset Output

Signals in Interface	Direction	Availability	Description
cal_debug_out_reset_n	Output	<ul style="list-style-type: none"> <li>• <b>EMIF Debug Toolkit / On-Chip Debug Port=Export</b></li> <li>• Add EMIF Debug Interface with <b>Enable Daisy-Chaining for EMIF Debug Toolkit/ On-Chip Debug Port=True</b></li> </ul>	<p>The calibration debug reset output signal. Asynchronous assertion and synchronous deassertion.</p> <p>Connect this interface to the (cal_debug_reset_reset_sink) reset input of the other external memory interface IP where the cal_debug_avalon_master interface being connected to and all the user logic blocks that are under the calibration debug clock domain (cal_debug_out_clk_clock_reset). If you are not using the Altera EMIF Debug Toolkit, connect this interface to the reset output of the custom debug logic.</p>

**Table 7-39: Interface: cal\_debug\_reset\_reset\_sink**

Interface type: Reset Input

Signals in Interface	Direction	Availability	Description
cal_debug_reset_n	Input	<ul style="list-style-type: none"> <li>• <b>EMIF Debug Toolkit / On-Chip Debug Port=Export</b></li> </ul>	<p>The calibration debug reset input signal. Require asynchronous assertion and synchronous deassertion.</p> <p>Connect this interface to the (av1_rst_out) reset output of the <b>Arria 10 EMIF Debug Component</b> IP or to (cal_debug_out_reset_reset_source) clock input of the other external memory interface IP, depending on which IP the cal_debug_avalon_slave interface is being connected to.</p>

**Table 7-40: Interface: clks\_sharing\_master\_out\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
clks_sharing_master_out	Input	<ul style="list-style-type: none"> <li>• <b>Core clocks sharing=Master</b></li> </ul>	<p>The core clock output signals.</p> <p>Connect this interface to the (clks_sharing_slave_in_conduit_end) conduit of the other external memory interface IP with the Core clock sharing set to slave or other PLL Slave.</p>

**Table 7-41: Interface: clks\_sharing\_slave\_in\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
clks_sharing_slave_in	Input	<ul style="list-style-type: none"> <li><b>Core clocks sharing=Slave</b></li> </ul>	The core clock input signals. Connect this interface to the (clks_sharing_master_out_conduit_end) conduit of the other external memory interface IP with the Core clock sharing set to Master or other PLL Master.

**Table 7-42: Interface: ctrl\_amm\_avalon\_slave**

Interface type: Avalon Memory-Mapped Slave

Signals in Interface	Direction	Availability	Description
amm_ready	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4 with Hard PHY &amp; Hard Controller</li> <li>QDR II/II+/II+ Xtreme, QDR IV</li> </ul>	The Avalon-MM signals between the external memory interface IP and the user logic. Connect this interface to the Avalon-MM Master of the user logic that needs to access the external memory device. For QDR II/II+/II+ Xtreme, connect the ctrl_amm_avalon_slave_0 to the user logic for read request and connect the ctrl_amm_avalon_slave_1 to the user logic for write request.
amm_read	Input		
amm_write	Input		
amm_address	Input		
amm_readdata	Output		
amm_writedata	Input		
amm_burstcount	Input		
amm_readdatavalid	Output		
amm_byteneable	Input	<ul style="list-style-type: none"> <li>DDR3, DDR4 with Hard PHY &amp; <b>Hard Controller and Enable DM pins=True</b></li> <li>QDR II/II+/II+ Xtreme with <b>Enable BWS# pins=True</b></li> </ul>	In Ping Pong PHY mode, each interface controls only one memory device. Connect ctrl_amm_avalon_slave_0 to the user logic that will access the first memory device, and connect ctrl_amm_avalon_slave_1 to the user logic that will access the secondary memory device.

**Table 7-43: Interface: ctrl\_auto\_preamble\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
ctrl_auto_preamble_req	Input	<ul style="list-style-type: none"> <li>DDR3, DDR4 with Hard PHY &amp; Hard Controller and <b>Enable Auto-Preamble Control=True</b></li> </ul>	<p>The auto-preamble control input signal. Asserting the <code>ctrl_auto_preamble_req</code> signal while issuing a read or write burst instructs the external memory interface IP to issue read or write with auto-preamble to the external memory device. This precharges the row immediately after the command currently accessing it finishes, potentially speeding up a future access to a different row of the same bank.</p> <p>Connect this interface to the conduit of the user logic block that controls when the external memory interface IP needs to issue read or write with auto-preamble to the external memory device.</p>

**Table 7-44: Interface: ctrl\_ecc\_user\_interrupt\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
ctrl_ecc_user_interrupt	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4 with Hard PHY &amp; Hard Controller and <b>Enable Error Detection and Correction Logic = True</b></li> </ul>	Controller ECC user interrupt interface for connection to a custom control block that must be notified when ECC errors occur.

**Table 7-45: Interface: ctrl\_mmr\_avalon\_slave**

Interface type: Avalon Memory-Mapped Slave

Signals in Interface	Direction	Availability	Description
mmr_waitrequest	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3 with Hard PHY &amp; Hard Controller and <b>Enable Memory-Mapped Configuration and Status Register (MMR)</b> =True</li> </ul>	<p>The Avalon-MM signals between the external memory interface IP and the user logic.</p> <p>Connect this interface to the Avalon-MM master of the user logic that needs to access the Memory-Mapped Configuration and Status Register (MMR) in the external memory interface IP.</p>
mmr_read	Input		
mmr_write	Input		
mmr_address	Input		
mmr_readdata	Output		
mmr_writedata	Input		
mmr_burstcount	Input		
mmr_bytelenable	Input		
mmr_beginburst-transfer	Input		
mmr_readdatavalid	Output		

**Table 7-46: Interface: ctrl\_power\_down\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
ctrl_power_down_ack	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3 with Hard PHY &amp; Hard Controller and <b>Enable Auto Power Down</b>=True</li> </ul>	<p>The auto power-down acknowledgement signals. When the <code>ctrl_power_down_ack</code> signal is asserted, it indicates that the external memory interface IP is placing the external memory device into power-down mode.</p> <p>Connect this interface to the conduit of the user logic block that requires the auto power-down status, or leave it unconnected.</p>

**Table 7-47: Interface: ctrl\_user\_priority\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
ctrl_user_priority_hi	Input	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3 with Hard PHY &amp; Hard Controller</li> <li>Avalon Memory-Mapped and <b>Enable Command Priority Control</b>=true</li> </ul>	<p>The command priority control input signal. Asserting the <code>ctrl_user_priority_hi</code> signal while issuing a read or write request instructs the external memory interface to treat it as a high-priority command. The external memory interface attempts to execute high-priority commands sooner, to reduce latency.</p> <p>Connect this interface to the conduit of the user logic block that determines when the external memory interface IP treats the read or write request as a high-priority command.</p>

**Table 7-48: Interface: emif\_usr\_clk\_clock\_source**

Interface type: Clock Output

Signals in Interface	Direction	Availability	Description
emif_usr_clk	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3, with Hard PHY &amp; Hard Controller</li> <li>QDR II/II+/II+ Xtreme</li> <li>QDR IV</li> </ul>	<p>The user clock output signal. The clock frequency in relation to the memory clock frequency depends on the <b>Clock rate of user logic</b> value set in the parameter editor.</p> <p>Connect this interface to the clock input of the respective user logic connected to the <code>ctrl_amm Avalon_slave_0</code> interface, or to any user logic block that must be clocked at the generated clock frequency.</p>

**Table 7-49: Interface: emif\_usr\_reset\_reset\_source**

Interface type: Reset Output

Signals in Interface	Direction	Availability	Description
emif_usr_reset_n	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3 with Hard PHY &amp; Hard Controller</li> <li>QDR II/II+/II+ Xtreme</li> <li>QDR IV</li> </ul>	<p>The user reset output signal. Asserted when the PLL becomes unlocked or the PHY is reset. Asynchronous assertion and synchronous deassertion.</p> <p>Connect this interface to the clock input of the respective user logic connected to the <code>ctrl_amm_avalon_slave_0</code> interface, or to any user logic block that must be clocked at the generated clock frequency.</p>

**Table 7-50: Interface: emif\_usr\_clk\_sec\_clock\_source**

Interface type: Clock Output

Signals in Interface	Direction	Availability	Description
emif_usr_clk_sec	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, with Ping Pong PHY</li> </ul>	<p>The user clock output signal. The clock frequency in relation to the memory clock frequency depends on the <b>Clock rate of user logic</b> value set in the parameter editor.</p> <p>Connect this interface to the clock input of the respective user logic connected to the <code>ctrl_amm_avalon_slave_1</code> interface, or to any user logic block that must be clocked at the generated clock frequency.</p>

**Table 7-51: Interface: emif\_usr\_reset\_sec\_reset\_source**

Interface type: Reset Output

Signals in Interface	Direction	Availability	Description
emif_usr_reset_n_sec	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, with Ping Pong PHY</li> </ul>	<p>The user reset output signal. Asserted when the PLL becomes unlocked or the PHY is reset. Asynchronous assertion and synchronous deassertion.</p> <p>Connect this interface to the clock input of the respective user logic connected to the <code>ctrl_amm_avalon_slave_1</code> interface, or to any user logic block that must be clocked at the generated clock frequency.</p>

**Table 7-52: Interface: global\_reset\_reset\_sink**

Interface type: Reset Input

Signals in Interface	Direction	Availability	Description
global_reset_n	Input	<ul style="list-style-type: none"> <li><b>Core Clock Sharing</b>=No Sharing / Master</li> </ul>	<p>The global reset input signal. Asserting the <code>global_reset_n</code> signal causes the external memory interface IP to be reset and recalibrated.</p> <p>Connect this interface to the reset output of the asynchronous or synchronous reset source that controls when the external memory interface IP needs to be reset and recalibrated.</p>

**Table 7-53: Interface: hps\_emif\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
hps_to_emif	Input	<ul style="list-style-type: none"> <li>Arria 10 EMIF for HPS IP</li> </ul>	The user interface signals between the external memory interface IP and the Hard Processor System (HPS).
emif_to_hps	Output		Connect this interface to the EMIF conduit of the Arria 10 Hard Processor System.

**Table 7-54: Interface: mem\_conduit\_end**

Interface type: Conduit

The memory interface signals between the external memory interface IP and the external memory device.

Export this interface to the top level for I/O assignments. Typically `mem_rm[0]` and `mem_rm[1]` connect to `CS2#` and `CS3#` of the memory buffer of all LRDIMM slots.

Signals in Interface	Direction	Availability
mem_ck	Output	Always available
mem_ck_n	Output	
mem_reset_n	Output	
mem_a	Output	
mem_k_n	Output	<ul style="list-style-type: none"> <li>QDR II</li> </ul>
mem_ras_n	Output	<ul style="list-style-type: none"> <li>DDR3</li> </ul>
mem_cas_n	Output	

Signals in Interface	Direction	Availability
mem_odt	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3</li> </ul>
mem_dqs	Bidirectional	
mem_dqs_n	Bidirectional	
mem_ba	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, RLDRAM 3</li> </ul>
mem_cs_n	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3, RLDRAM 3</li> </ul>
mem_dq	Bidirectional	
mem_we_n	Output	<ul style="list-style-type: none"> <li>DDR3, RLDRAM 3</li> </ul>
mem_dm	Output	<ul style="list-style-type: none"> <li>DDR3, LPDDR3, RLDRAM 3 with <b>Enable DM pins=True</b></li> </ul>
mem_rm	Output	<ul style="list-style-type: none"> <li>DDR3, RLDRAM 3 with <b>Memory format=LRDIMM</b> and <b>Number of rank multiplication pins &gt; 0</b></li> </ul>
mem_par	Output	<ul style="list-style-type: none"> <li>DDR3 with <b>Memory format=RDIMM / LRDIMM</b></li> <li>DDR4 with <b>Enable alert_n/par pins=True</b></li> </ul>
mem_alert_n	Input	
mem_cke	Output	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3</li> </ul>
mem_bg	Output	<ul style="list-style-type: none"> <li>DDR4</li> </ul>
mem_act_n	Output	
mem_dbi_n	Bidirectional	<ul style="list-style-type: none"> <li>DDR4 with <b>Enable DM pins=True</b> or <b>Write DBI=True</b> or <b>Read DBI=True</b></li> </ul>
mem_k	Output	<ul style="list-style-type: none"> <li>QDR II/II+/II+ Xtreme</li> </ul>
mem_wps_n	Output	
mem_rps_n	Output	
mem_doff_n	Output	
mem_d	Output	
mem_q	Input	
mem_cq	Input	
mem_cq_n	Input	
mem_bws_n	Output	
mem_dk	Output	
mem_dk_n	Output	
mem_ref_n	Output	

Signals in Interface	Direction	Availability
mem_qk	Input	• QDR II/II+/II+ Xtreme with <b>Enable BWS# pins=True</b>
mem_qk_n	Input	• RLDRAM 3
mem_ap	Output	• QDR IV with <b>Use Address Parity Bit=True</b>
mem_pe_n	Input	• QDR IV with <b>Use Address Parity Bit=True</b>
mem_ainv	Output	• QDR IV with <b>Address Bus Inversion=True</b>
mem_lda_n	Output	• QDR IV
mem_lda_b	Output	• QDR IV
mem_rwa_n	Output	• QDR IV
mem_rwb_n	Output	• QDR IV
mem_cfg_n	Output	• QDR IV
mem_lbk0_n	Output	• QDR IV
mem_lbk1_n	Output	• QDR IV
mem_dka	Output	• QDR IV
mem_dka_n	Output	• QDR IV
mem_dkb	Output	• QDR IV
mem_dkb_n	Output	• QDR IV
mem_qka	Input	• QDR IV
mem_qka_n	Input	• QDR IV
mem_qkb	Input	• QDR IV
mem_qkb_n	Input	• QDR IV
mem_dqa	Bidirectional	• QDR IV
mem_dqb	Bidirectional	• QDR IV
mem_dinva	Bidirectional	• QDR IV with <b>Data Bus Inversion=True</b>

Signals in Interface	Direction	Availability
mem_dinvb	Bidirectional	<ul style="list-style-type: none"> <li>• QDR IV with <b>Data Bus Inversion=True</b></li> </ul>

**Table 7-55: Interface: oct\_conduit\_end**

Interface type: Conduit

Signals in Interface	Direction	Availability	Description
oct_rzqin	Input	Always available	<p>The On-Chip Termination (OCT) RZQ reference resistor input signal.</p> <p>Export this interface to the top level for I/O assignments.</p>

**Table 7-56: Interface: pll\_ref\_clk\_clock\_sink**

Signals in Interface	Interface Type	Direction	Availability	Description
pll_ref_clk	Clock Input	Input	<ul style="list-style-type: none"> <li>• <b>Core clock sharing=No Sharing / Master</b></li> </ul>	<p>The PLL reference clock input signal.</p> <p>Connect this interface to the clock output of the clock source that matches the <b>PLL reference clock frequency</b> value set in the parameter editor.</p>

**Table 7-57: Interface: status\_conduit\_end**

Signals in Interface	Interface Type	Direction	Availability	Description
local_cal_success	Conduit	Output	Always available	<p>The PHY calibration status output signals. When the <code>local_cal_success</code> signal is asserted, it indicates that the PHY calibration was successful. Otherwise, if <code>local_cal_fail</code> signal is asserted, it indicates that PHY calibration has failed.</p> <p>Connect this interface to the conduit of the user logic block that requires the calibration status information, or leave it unconnected.</p>
local_cal_fail				

**Related Information**[http://www.alterawiki.com/wiki/Measuring\\_Channel\\_Signal\\_Integrity](http://www.alterawiki.com/wiki/Measuring_Channel_Signal_Integrity)

## Generated Files for Arria 10 External Memory Interface IP

When you complete the IP generation flow, there are generated files created in your project directory. The directory structure created varies somewhat, depending on the tool used to parameterize and generate the IP.

**Note:** The PLL parameters are statically defined in the `<variation_name>.parameters.tcl` at generation time. To ensure timing constraints and timing reports are correct, when you edit the PLL parameters, apply those changes to the PLL parameters in this file.

The following table lists the generated directory structure and key files created when generating the IP.

**Table 7-58: Generated Directory Structure and Key Files for Synthesis**

Directory	File Name	Description
<code>working_dir/</code>	<code>working_dir/&lt;Top-level Name&gt;/</code>	The Qsys files for your IP component or system based on your configuration.
<code>working_dir/&lt;Top-level Name&gt;/</code>	<code>*.ppf</code>	Pin Planner File for use with the Pin Planner.
<code>working_dir/&lt;Top-level Name&gt;/synth/</code>	<code>&lt;Top-level Name&gt;.v or &lt;Top-level Name&gt;.vhdl</code>	Qsys generated top-level wrapper for synthesis.
<code>working_dir/&lt;Top-level Name&gt;/altera_emif_&lt;acds version&gt;/synth/</code>	<code>*.v or (*.v and *.vhdl)</code>	Arria 10 EMIF (non-HPS) top-level dynamic wrapper files for synthesis. This wrapper instantiates the EMIF ECC and EMIF Debug Interface IP core.
<code>working_dir/&lt;Top-level Name&gt;/altera_emif_a10_hps_&lt;acds version&gt;/synth/</code>	<code>*.v or (*.v and *.vhdl)</code>	Arria 10 EMIF for HPS top-level dynamic wrapper files for synthesis.
<code>working_dir/&lt;Top-level Name&gt;/altera_emif_a10_hps_&lt;acds version&gt;/synth/</code>	<code>*.sv, *.sdc, *.tcl and *.hex and *_readme.txt</code>	Arria 10 EMIF Core RTL, constraints files, ROM content files and information files for synthesis.
<code>working_dir/&lt;Top-level Name&gt;/&lt;other components&gt;_&lt;acds version&gt;/synth/</code>	<code>*</code>	Whether the file type is set to Verilog or VHDL, all the Arria 10 EMIF Core RTL files will be generated as a SystemVerilog file. The readme.txt file contains information and guidelines specific to your configuration.
<code>working_dir/&lt;Top-level Name&gt;/&lt;other components&gt;_&lt;acds version&gt;/synth/</code>	<code>*</code>	Other EMIF ECC, EMIF Debug Interface IP or Merlin Interconnect component files for synthesis.

**Table 7-59: Generated Directory Structure and Key Files for Simulation**

Directory	File Name	Description
<code>working_dir/&lt;Top-level Name&gt;/sim/</code>	<code>&lt;Top-level Name&gt;.v or &lt;Top-level Name&gt;.vhdl</code>	Qsys generated top-level wrapper for simulation.

Directory	File Name	Description
<i>working_dir/&lt;Top-level Name&gt;/sim/&lt;simulator vendor&gt;/</i>	<b>*.tcl, *.cds.lib, *.lib, *.var, *.sh, *.setup</b>	Simulator-specific simulation scripts.
<i>working_dir/&lt;Top-level Name&gt;/altera_emif_&lt;acds version&gt;/sim/</i>	<b>*.v or *.vhdl</b>	Arria 10 EMIF (non-HPS) top-level dynamic wrapper files for simulation. This wrapper instantiates the EMIF ECC and EMIF Debug Interface IP cores.
<i>working_dir/&lt;Top-level Name&gt;/altera_emif_a10_hps_&lt;acds version&gt;/sim/</i>	<b>*.v or *.vhdl</b>	Arria 10 EMIF for HPS top-level dynamic wrapper files for simulation.
<i>working_dir/&lt;Top-level Name&gt;/altera_emif_arch_nf_&lt;acds version&gt;/sim/</i>	<b>*.sv or (*.sv and *.vhdl), *.hex and *_readme.txt</b>	Arria 10 EMIF RTL, ROM content files, and information files for simulation.  For SystemVerilog / Mix language simulator, you may directly use the files from this folder. For VHDL-only simulator, other than the ROM content files, you have to use files in <b>&lt;current folder&gt;/mentor</b> directory instead.  The <b>readme.txt</b> file contains information and guidelines specific to your configuration.
<i>working_dir/&lt;Top-level Name&gt;/&lt;other components&gt;_&lt;acds version&gt;/sim/</i>		Other EMIF ECC, EMIF Debug Interface IP, or Merlin Interconnect component files for simulation

Table 7-60: Generated Directory Structure and Key Files for Qsys-Generated Testbench System

Directory	File Name	Description
<i>working_dir/&lt;Top-level Name&gt;_tb/</i>	<b>*.qsys</b>	The Qsys files for the QSYS generated testbench system.
<i>working_dir/&lt;Top-level Name&gt;_tb/sim/</i>	<b>&lt;Top-level Name&gt;.v or &lt;Top-level Name&gt;.vhdl</b>	Qsys generated testbench file for simulation.  This wrapper instantiates BFM components. For Arria 10 EMIF IP, this module should instantiate the memory model for the memory conduit being exported from your created system.
<i>working_dir/&lt;Top-level Name&gt;_tb/&lt;Top-level Name&gt;_&lt;id&gt;/sim/</i>	<b>&lt;Top-level Name&gt;.v or &lt;Top-level Name&gt;.vhdl</b>	Qsys generated top-level wrapper for simulation.

Directory	File Name	Description
<b><i>working_dir/&lt;Top-level Name&gt;_tb/sim/&lt;simulator vendor&gt;/</i></b>	<b><i>*.tcl, *.cds.lib, *.lib, *.var, *.sh, *.setup</i></b>	Simulator-specific simulation scripts.
<b><i>working_dir/&lt;Top-level Name&gt;_tb/sim/&lt;simulator vendor&gt;/</i></b>	<b><i>*.v or *.vhdl</i></b>	Arria 10 EMIF (non-HPS) top-level dynamic wrapper files for simulation.  This wrapper instantiates the EMIF ECC and EMIF Debug Interface IP cores.
<b><i>working_dir/&lt;Top-level Name&gt;_tb/altera_emif_a10_hps_&lt;acds version&gt;/sim/</i></b>	<b><i>*.v or *.vhdl</i></b>	Arria 10 EMIF for HPS top-level dynamic wrapper files for simulation.
<b><i>working_dir/&lt;Top-level Name&gt;_tb/altera_emif_arch_nf_&lt;acds version&gt;/sim/</i></b>	<b><i>*.sv or (*.sv and *.vhdl), *.hex and *_readme.txt</i></b>	Arria 10 EMIF Core RTL, ROM content files and information files for simulation.  For SystemVerilog / Mix language simulator, you may use the files from this folder. For VHDL-only simulator, other than the ROM content files, you must use files in the <b>&lt;current folder&gt;/mentor</b> directory instead. The <b>readme.txt</b> file contains information and guidelines specific to your configuration.
<b><i>working_dir/&lt;Top-level Name&gt;_tb/sim/altera_emif_arch_nf_&lt;acds version&gt;/sim/mentor/</i></b>	<b><i>*.sv and *.vhdl</i></b>	Arria 10 EMIF Core RTL for simulation.  Only available when you create a VHDL simulation model. All <b>.sv</b> files are Mentor-tagged encrypted IP (IEEE Encrypted Verilog) for VHDL-only simulator support.
<b><i>working_dir/&lt;Top-level Name&gt;_tb/&lt;other components&gt;_&lt;acds version&gt;/sim/</i></b>	<b><i>*</i></b>	Other EMIF ECC, EMIF Debug Interface IP or Merlin Interconnect component files for simulation.
<b><i>working_dir/&lt;Top-level Name&gt;_tb/&lt;other components&gt;_&lt;acds version&gt;/sim/mentor/</i></b>	<b><i>*</i></b>	Other EMIF ECC, EMIF Debug Interface IP or Merlin Interconnect component files for simulation.  Only available depending on individual component simulation model support and when creating a VHDL simulation model. All files in this folder are Mentor-tagged encrypted IP (IEEE Encrypted Verilog) for VHDL-only simulator support.

**Table 7-61: Generated Directory Structure and Key Files for Example Simulation Designs**

Directory	File Name	Description
<b><i>working_dir/*_example_design*/</i></b>	<b>*.qsys, *.tcl and <b>readme.txt</b></b>	Qsys files, generation scripts, and information for generating the Arria 10 EMIF IP example design.  These files are available only when you generate an example design. You may open the .qsys file in Qsys to add more components to the example design.
<b><i>working_dir/*_example_design*/sim/ed_sim/sim/&lt;simulator vendor&gt;/</i></b>	<b>*.v or *.vhdl</b>	Qsys-generated top-level wrapper for simulation.
<b><i>working_dir/*_example_design*/sim/ed_sim/sim/altera_emif_&lt;acds_version&gt;/sim/</i></b>	<b>*.tcl, *cds.lib, *.lib, *.var, *.sh, *.setup</b>	Simulator-specific simulation scripts.
<b><i>working_dir/*_example_design*/sim/ed_sim/altera_emif_arch_nf_&lt;acds_version&gt;/sim/</i></b>	<b>*.v or *.vhdl</b>	Arria 10 EMIF (non-HPS) top-level dynamic wrapper files for simulation. This wrapper instantiates the EMIF ECC and EMIF Debug Interface IP cores.
<b><i>working_dir/*_example_design*/sim/ed_sim/altera_emif_arch_nf_&lt;acds_version&gt;/sim/</i></b>	<b>*sv or (*.sv and *.vhdl), *.hex and *_readme.txt</b>	Arria 10 EMIF RTL, ROM content files, and information files for simulation. For SystemVerilog / Mix language simulator, you may directly use the files from this folder. For VHDL-only simulator, other than the ROM content files, you have to use files in <b>&lt;current folder&gt;/mentor directory</b> instead. The <b>readme.txt</b> file contains information and guidelines specific to your configuration.
<b><i>working_dir/*_example_design*/sim/ed_sim/&lt;other components&gt;_&lt;acds_version&gt;/sim/</i></b>	<b>*</b>	Other EMIF ECC, EMIF Debug Interface IP, or Merlin Interconnect component files for simulation

**Table 7-62: Generated Directory Structure and Key Files for Example Synthesis Designs**

Directory	File Name	Description
<b><i>working_dir/*_example_design*/</i></b>	<b>*.qsys, *.tcl and <b>readme.txt</b></b>	Qsys files, generation scripts, and information for generating the Arria 10 EMIF IP example design.  These files are available only when you generate an example design. You may open the .qsys file in Qsys to add more components to the example design.
<b><i>working_dir/*_example_design*/qii/ed_synth/synth</i></b>	<b>*.v or (*.v and *.vhdl)</b>	Qsys-generated top-level wrapper for synthesis.

Directory	File Name	Description
<code>working_dir/*_example_design*/qii/ed_synth/altera_emif_&lt;acds_version&gt;/synth</code>	<code>*.v</code> or ( <code>*.v</code> and <code>*.vhdl</code> )	Arria 10 EMIF (non HPS) top-level dynamic wrapper files for synthesis. This wrapper instantiates the EMIF ECC and EMIF debug interface core IP.
<code>working_dir/*_example_design*/qii/ed_synth/&lt;other components&gt;_&lt;acds_version&gt;/synth</code>	*	Other EMIF ECC, EMIF debug interface IP, or Merlin interconnect component files for synthesis.

## Parameterizing Arria 10 External Memory Interface IP

The Arria 10 EMIF IP parameter editor allows you to parameterize settings for the DDR3, DDR4, LPDDR3, QDR II/II+/II+ Xtreme, QDR-IV, and RLDRAM 3 memory controllers for Arria 10 EMIF IP.

You can parameterize settings for the following tabs in the parameter editor:

- General
- I/O
- Memory Topology
- Memory Timing
- Board Timing
- Controller (Not applicable if you have selected Hard PHY only.)
- Diagnostics
- Example Designs

The text window at the bottom of the parameter editor displays information about the memory interface, as well as warning and error messages. You should correct any errors indicated in this window before clicking the **Finish** button.

**Note:** Default settings are the minimum required to achieve timing, and may vary depending on memory protocol.

The following tables describe the parameterization settings available in the parameter editor for the Arria 10 EMIF IP.

### General Parameters for Arria 10 EMIF IP

**Table 7-63: Group: General / Memory Protocol**

Display Name	Description	Protocol
Protocol	The protocol of the memory interface.	All

**Group: General / FPGA**

Display Name	Description	Protocol
<b>Speed grade</b>	<p>Specifies the speed grade of the FPGA, and whether it is an engineering sample (ES) or a production device. The value is automatically determined based on the device selected under <b>View &gt; Device Family</b>. If no device is specified, a production device of the fastest speed grade is assumed.</p> <p>You should always specify the correct target device during IP generation. Failure to do so may result in IP that does not work on hardware.</p>	All

**Table 7-64: Group: General / Interface**

Display Name	Description	Protocol
<b>Configuration</b>	<p>Specifies the configuration of the memory interface. Several options are available, depending on protocol:</p> <ul style="list-style-type: none"> <li>• (DDR3, DDR4, LPDDR3) You may select <b>Hard PHY and Hard Controller</b> or <b>Hard PHY-Only</b>.</li> <li>• (QDR II/II+/II+ Xtreme, QDR IV) You can select only <b>Hard PHY and Soft Controller</b>.</li> <li>• (RLDRAM 3) You can select <b>Hard PHY Only</b>. If you select <b>Hard PHY Only</b>, the AFI interface is exported so that you can connect your own memory controller or any third party IP.</li> </ul>	All
<b>Instantiate two controllers sharing a Ping Pong PHY</b>	Select this option to instantiate two identically configured memory controllers that share the same address/command bus through time multiplexing (i.e. Ping Pong PHY topology). Ping Pong PHY results in an interface with fewer pins, compared to having two independent interfaces. When this option is enabled, the IP core exposes two independent Avalon interfaces to user logic, and a single external memory interface with double the width for the data bus and the CS#, CKE, ODT, and CK/CK# signals. (This option is available only if you specify the Hard PHY and Hard Controller option.)	DDR3, DDR4

**Table 7-65: Group: General / Clocks**

Display Name	Description	Protocol
<b>Memory clock frequency</b>	Frequency of memory clock in MHz.	All
<b>Use recommended PLL reference clock frequency</b>	When checked, the PLL reference clock frequency is automatically calculated for best performance. Uncheck the check box if you want to specify your own PLL reference clock frequency.	All

Display Name	Description	Protocol
<b>PLL reference clock frequency</b>	PLL reference clock frequency. You must select a valid PLL reference clock frequency from the list. The values in the list can change if you change the memory interface frequency and/or the clock rate of user logic.	All
<b>PLL reference clock jitter</b>	The peak-to-peak jitter on the PLL reference clock source. Should not exceed 10 ps time interval error (TIE).  (Time interval error is the time deviation of the signal edge from its ideal position.)	All
<b>Clock rate of user logic</b>	Determines the clock frequency of user logic in relation to the memory clock frequency. For example, if the memory clock sent from the FPGA to the memory device is toggling at 800MHz, a "Quarter rate" interface means that the user logic in the FPGA runs at 200MHz.	All
<b>Core clocks sharing</b>	When a design contains multiple interfaces of the same protocol, rate, frequency, and PLL reference clock source, it is possible to share a common set of core clock domains, to reduce clock network usage and to avoid clock synchronization logic between interfaces. To achieve this, denote one of the interfaces as Master, and the remaining interfaces as Slave. Then, in the RTL, connect the <code>clks_sharing_master_out</code> signal from the master interface to the <code>clks_sharing_slave_in</code> signal of all the slave interfaces. Both master and slave interfaces still expose their own output clock ports in the RTL (for example, <code>emif_usr_clk</code> , <code>afi_clk</code> ), but the physical signals are equivalent and so it does not matter whether a clock port from a master or a slave is used.	All
<b>Specify additional core clocks based on existing PLL</b>	When checked, displays additional parameters allowing you to create additional output clocks based on the existing PLL. The additional output clocks that you create can be fed into the core, but have no guaranteed phase relationship with other clocks (such as the DPA clock) generated by the IP.	All

**Table 7-66: Group: General / Clocks / Output Clocks**

Display Name	Description	Protocol
<b>Number of additional core clocks</b>	Specifies the number of additional output clocks to create.	All
<b>Desired frequency</b>	For a given output clock, specifies the desired frequency.	All
<b>Actual frequency</b>	Indicates the actual frequency.	All

Display Name	Description	Protocol
<b>Phase shift units</b>	Specifies the units for phase shift.	All
<b>Phase shift</b>	Specifies the phase shift.	All
<b>Actual phase shift</b>	Indicates the actual phase shift.	All
<b>Desired duty cycle</b>	Specifies the desired duty cycle.	All
<b>Actual duty cycle</b>	Indicates the actual duty cycle.	All

### I/O Parameters for Arria 10 EMIF IP

The following tables list the I/O parameters for Arria 10 EMIF IP. Use HyperLynx or similar simulators to determine the best settings for your board.

**Table 7-67: Group: I/O / General I/O Settings**

Display Name	Description	Protocol
<b>Voltage</b>	I/O voltage of the signaling between memory device and memory interface.	All
<b>Use default I/O settings</b>	When checked, a legal set of I/O settings are automatically selected. Note that the default I/O settings are not necessarily optimized for your board. To achieve optimal signal integrity, perform I/O simulation and manually enter the I/O settings based on simulation results, instead of using the defaults.	All

**Table 7-68: Group: I/O / Address/Command**

Display Name	Description	Protocol
<b>I/O standard</b>	I/O standard for the address/command pins of the memory interface.	All
<b>Output mode</b>	Output termination or current strength of the address/command output pins.	All
<b>Slew rate</b>	Slew rate of the address/command output pins,	All

**Table 7-69: Group: I/O / Memory Clock**

Display Name	Description	Protocol
<b>I/O standard</b>	I/O standard for the memory clock pins of the memory interface.	All
<b>Output mode</b>	Output termination or current strength of the memory clock output pins.	All
<b>Slew rate</b>	Slew rate of the memory clock output pins.	All

**Table 7-70: Group: I/O / Data Bus**

Display Name	Description	Protocol
<b>I/O standard</b>	I/O standard for the data and data clock/strobe pins of the memory interface.	All
<b>Output mode</b>	Output termination or current strength of the data and data clock/strobe pins.	All
<b>Input mode</b>	Input termination of the data and data clock/strobe pins.	All

**Table 7-71: Group: I/O / PHY Inputs**

Display Name	Description	Protocol
<b>PLL reference clock I/O standard</b>	/O standard for the PLL reference clock of the memory interface.	All
<b>RZQ I/O standard</b>	I/O standard for the RZQ pin of the memory interface.	All
<b>RZQ Resistor</b>	Connect the RZQ pin to GND through an external resistor of the specified value.	All

## Memory Topology Parameters for Arria 10 EMIF IP

The following tables list the Memory Topology parameters for Arria 10 EMIF IP. Configure these parameters based on your selected memory manufacturer's data sheet.

**Table 7-72: Group: Memory Topology / Topology**

Display Name	Description	Protocol
<b>Memory format</b>	The format of the external memory device. The external memory device may be accessed directly as discrete components, or via DIMMs (which can be unbuffered, registered, or load-reduced, or SODIMM).	DDR3, DDR4
<b>DQ width</b>	Total number of DQ pins in the interface. Maximum supported width is 144, or 72 in Ping Pong PHY mode (DDR3 and DDR4).	DDR3, DDR4, LPDDR3
<b>DQ pins per DQS group</b>	Number of DQ pins per DQS group.	DDR3, DDR4
<b>Number of DQS Groups</b>	Total number of DQS groups in the interface. Automatically calculated as the total interface width divided by the number of DQ pins per DQS group.	DDR3, DDR4, LPDDR3
<b>Number of clocks</b>	Number of CK/CK# clock pairs exposed by the memory interface.	DDR3, DDR4
<b>Number of chip selects</b>	Total number of chip selects in the interface.	DDR3, DDR4, LPDDR3
<b>Number of DIMMs</b>	Total number of DIMMs.	DDR3, DDR4

Display Name	Description	Protocol
<b>Number of physical ranks per DIMM</b>	Number of ranks per DIMM. For LRDIMM, this represents the number of physical ranks on the DIMM behind the memory buffer.	DDR3, DDR4
<b>Number of rank multiplication pins</b>	Number of rank multiplication pins used to access all the physical ranks on an LRDIMM. These pins should be connected to CS#[2] and/or CS#[3] of all LRDIMMs in the system.	DDR3, DDR4
<b>Number of chip selects per DIMM</b>	Number of chip selects (i.e. CS#) per DIMM.	DDR3, DDR4
<b>Number of clock enables per DIMM</b>	Number of clock enables (for example, CKE) per DIMM. Only applicable to registered and load-reduced DIMMs.	DDR3, DDR4
<b>Row address width</b>	Row address width. This is used to derive the total number of address pins required.	DDR3, DDR4, LPDDR3
<b>Column address width</b>	Column address width. This is used to derive the total number of address pins required.	DDR3, DDR4, LPDDR3
<b>Bank address width</b>	Number of bank address pins.	DDR3, DDR4, LPDDR3, RLDIMM 3
<b>Bank group width</b>	Number of bank group pins.	DDR4
<b>DQ width per device</b>	<ul style="list-style-type: none"> <li>Number of DQ pins per RLDIMM 3 device.</li> <li>Number of DQ pins per port per QDR IV device</li> </ul>	RLDIMM 3, QDR IV
<b>DQ width</b>	Number of DQ pins of the memory interface. Automatically calculated based on the DQ width per device and whether width expansion is enabled.	RLDIMM 3
<b>QK width</b>	Number of QK clock pairs of the memory interface. This is equal to the number of read data groups, and is automatically calculated based on the DQ width per device and whether width expansion is enabled.	RLDIMM 3
<b>DK width</b>	Number of DK clock pairs of the memory interface. This is equal to the number of write data groups, and is automatically calculated based on the DQ width per device and whether width expansion is enabled.	RLDIMM 3
<b>CS# width</b>	Number of chip selects of the memory interface.	RLDIMM 3
<b>Enable DM pins</b>	Indicates whether the interface uses the DM pins. If enabled, one DM pin per DQS group is added.	DDR3, LPDDR3, RLDIMM 3
<b>Enable ALERT#/PAR pins</b>	Enables the alert_n/par pins used for address/command parity and CRC.	DDR4

Display Name	Description	Protocol
<b>ALERT# pin placement</b>	<p>The ALERT# (also known as ERR_OUT#) pin can either be placed in an address/command I/O lane or in a data I/O lane. If you place the pin in an address/command I/O lane, IP generation automatically selects a valid lane and pin location. If you place the pin in a data I/O lane, you must also specify the DQS group with which the pin is placed. It is recommended that you accept the default setting.</p> <p>For interfaces containing multiple memory devices, it is recommended that you connect their ALERT# pins together to the ALERT# pin on the FPGA.</p>	DDR4
<b>DQS group of ALERT#</b>	Select the DQS group with which the ALERT# pin is placed.	DDR4
<b>Data width per device</b>	Number of D and Q pins per QDR II device.	QDR II/II+ /II+ Xtreme
<b>Data width</b>	Number of D and Q pins of the memory interface. Automatically calculated based on the D and Q width per device and whether width expansion is enabled.	QDR II/II+ /II+ Xtreme
<b>BWS# width</b>	Number of BWS# pins of the memory interface. Automatically calculated based on the data width per device and whether width expansion is enabled.	QDR II/II+ /II+ Xtreme
<b>CQ width</b>	Width of the CQ (read strobe) clock on the memory device.	QDR II/II+ /II+ Xtreme
<b>K width</b>	Width of the K (address, command and write strobe) clock on the memory device.	QDR II/II+ /II+ Xtreme
<b>Enable BWS# pins</b>	Indicates whether the interface uses the BWS# pins. If enabled, 1 bws pin for every 9 D pins will be added.	QDR II/II+ /II+ Xtreme
<b>Enable width expansion</b>	Indicates whether to combine two memory devices to double the data bus width.	QDR II/II+ /II+ Xtreme, RLDRAM 3, QDR IV
<b>Enable depth expansion using twin die package</b>	Indicates whether to combine two RLDRAM 3 devices to double the address space.	RLDRAM 3
<b>Address width</b>	Number of address pins.	QDR II/II+ /II+ Xtreme, RLDRAM 3, QDR IV
<b>Burst length</b>	Burst length of the memory device.	QDR II/II+ /II+ Xtreme
<b>DQA/DQB width</b>	Number of DQ pins for port A or B of the memory interface. Automatically calculated based on the DQ width per device and whether width expansion is enabled.	QDR IV

Display Name	Description	Protocol
<b>QKA/QKB width</b>	Number of QK clock pairs for port A or B of the memory interface. Automatically calculated based on the DQ width per device and whether width expansion is enabled.	QDR IV
<b>DKA/DKB width</b>	Number of DK clock pairs for port A or B of the memory interface. Automatically calculated based on the DQ width per device and whether width expansion is enabled.	QDR IV
<b>DINVA/DINVB width</b>	Number of DINV pins for port A or B of the memory interface. Automatically calculated based on the DQ width per device and whether width expansion is enabled.	QDR IV
<b>Data mask</b>	Indicates whether the interface uses data mask. For DDR4, the data mask pin is named <code>dbi_n</code> even if you do not turn on DBI.	DDR4
<b>Write DBI</b>	Write data bus inversion.	DDR4
<b>Read DBI</b>	Read data bus inversion.  <b>Note:</b> To achieve maximum performance for a DDR4 or QDR-IV interface on an Arria 10 device, read data bus inversion must be enabled. For DDR4, periodic OCT calibration must also be enabled.	DDR4, QDR-IV

Table 7-73: Group: Memory Topology / Configuration Register Settings

Display Name	Description	Protocol
<b>Address bus inversion</b>	Enable address bus inversion.	QDR IV
<b>Data bus inversion</b>	Enable data bus inversion.	QDR IV
<b>ODT (Clock)</b>	Determines the configuration register setting that controls the clock ODT setting.	QDR IV
<b>ODT (Address/Command)</b>	Determines the configuration register setting that controls the address/command ODT setting.	QDR IV
<b>ODT (Data)</b>	Determines the configuration register setting that controls the data ODT setting.	QDR IV
<b>Output Drive (pull up)</b>	Determines the configuration register setting that controls the pull-up output drive setting.	QDR IV

Display Name	Description	Protocol
<b>Output Drive (pull down)</b>	Determines the configuration register setting that controls the pull-down output drive setting.	QDR IV

**Table 7-74: Group: Memory Topology / Mode Register Settings**

Display Name	Description	Protocol
<b>tRC</b>	Determines the mode register setting that controls the tRC.	RLDRAM 3
<b>Data Latency</b>	Determines the mode register setting that controls the data latency.	LPDDR3, RLDRAM 3
<b>Output drive</b>	Determines the mode register setting that controls the output drive setting.	RLDRAM 3
<b>ODT</b>	Determines the mode register setting that controls the ODT setting.	RLDRAM 3
<b>AREF protocol</b>	Determines the mode register setting that controls the AREF protocol setting.	RLDRAM 3
<b>Enable Mirror Addressing for odd ranks</b>	Enabling Address Mirroring for dual rank or quad rank DIMM.	DDR3, DDR4
<b>Burst Length</b>	Specifies the burst length.	DDR3, DDR4, LPDDR3, RLDRAM 3
<b>Write protocol</b>	Determines the mode register setting that controls the write protocol setting.	RLDRAM 3
<b>Read Burst Type</b>	Specifies whether accesses within a given burst are in sequential or interleaved order.	DDR3, DDR4
<b>DLL precharge power down</b>	Specifies whether the DLL in the memory device is off or on during precharge power-down.	DDR3
<b>Memory CAS latency setting</b>	Determines the number of clock cycles between the READ command and the availability of the first bit of output data at the memory device.	DDR3, DDR4

Display Name	Description	Protocol
<b>Output drive strength setting</b>	Specifies the output driver impedance setting at the memory device.	DDR3, DDR4, LPDDR3
<b>Memory additive CAS latency setting</b>	Determines the posted CAS additive latency of the memory device.	DDR3, DDR4
<b>ODT Rtt nominal value</b>	Determines the ODT resistance at the memory device.	DDR3, DDR4
<b>Auto self-refresh method</b>	Turn on or off auto selfrefresh.	DDR3, DDR4
<b>Self-refresh temperature</b>	Specifies the self-refresh temperature is "Normal" or "Extended" mode.	DDR3
<b>Memory write CAS latency setting</b>	Specifies the number of clock cycles from the releasing of the internal write to the latching of the first data in, at the memory device.	DDR3, DDR4
<b>Dynamic ODT (Rtt_WR) value</b>	Specifies the mode of the dynamic ODT feature of the memory device.	DDR3, DDR4
<b>DDR3 RDIMM/LRDIMM control words</b>	Each 4-bit word can be obtained from the manufacturer's data sheet, and should be entered in hexadecimal, starting with RC15 on the left and ending with RC0 on the right.	DDR3
<b>DDR4 RDIMM/LRDIMM control words</b>	Each 4/8-bit setting can be obtained from the manufacturer's data sheet, and should be entered in hexadecimal, starting with the 8-bit setting RCBx on the left and continuing to RC1x, followed by the 4-bit setting RC0F and ending with RC00 on the right.	DDR4
<b>DDR3 LRDIMM additional control words</b>	Each 8-bit word can be obtained from the LRDIMM SPD info, and should be entered in hexadecimal, starting with SPD(77-72) or SPD(83-78) on the left and ending with SPD(71-69) on the right.	DDR3

Display Name	Description	Protocol
<b>DDR4 LRDIMM additional control words</b>	Each 4-bit word can be obtained from the manufacturer's data sheet, and should be entered in hexadecimal, starting with BC0F on the left and ending with BC00 on the right.	DDR4
<b>DDR4 LRDIMM VREFDQ value</b>	Obtain this setting from DDR4 LRDIMM SPD byte #140 and enter it in hexadecimal.	DDR4
<b>Fine granularity refresh</b>	Fine granularity refresh setting (increased frequency of refresh in exchange for shorter refresh).	DDR4
<b>Temperature controlled refresh range</b>	Temperature controlled refresh range (Normal: 0C to 85C; Extended: 85C to 95C).	DDR4
<b>Temperature controlled refresh enable</b>	Temperature controlled refresh enable.	DDR4
<b>Internal VrefDQ monitor</b>	Enable/Disable Internal VrefDQ monitor.	DDR4
<b>Self-refresh abort</b>	Self refresh abort for latency reduction.	DDR4
<b>Read preamble</b>	Read preamble (cycles).	DDR4
<b>Write preamble</b>	Write preamble (cycles).	DDR4
<b>Addr/CMD parity latency</b>	Additional latency incurred by enabling Address/Command parity check.	DDR4
<b>ODT input buffer during powerdown mode</b>	Enable/Disable ODT input buffer during powerdown mode.	DDR4
<b>RTT PARK</b>	If set, termination is always enabled.	DDR4
<b>Use recommended initial VrefDQ value</b>	Select to use the recommended initial VrefDQ value. This value is used as a starting point, but may change after calibration completes.	DDR4
<b>VrefDQ training value</b>	VrefDQ training value.	DDR4
<b>VrefDQ training range</b>	VrefDQ training range.	DDR4

Display Name	Description	Protocol
<b>WL set</b>	The set of the currently selected write latency. Only certain memory devices support <b>WL Set B</b> .	LPDDR3
<b>DQ ODT</b>	The ODT setting for the DQ pins during writes.	LPDDR3
<b>Power down ODT</b>	Turn on turn off ODT during power down.	LPDDR3
<b>nWR cycles</b>	The number of clock cycles that determines when to start an internal precharge operation for a write burst with AP enabled.	LPDDR3

**Table 7-75: Group: Memory Topology / ODT Activation Settings**

Display Name	Description	Protocol
<b>Use Default ODT Assertion Tables</b>	Enables the default ODT assertion pattern as determined from vendor guidelines. These settings are provided as a default only; Altera recommends that you simulate your memory interface topology to determine the optimal ODT settings and assertion patterns for your interface.	DDR3, DDR4, LPDDR3
<b>ODT Assertion Table during Read Accesses</b>	Set the value to on to assert a given ODT signal when reading from a specific bank.	DDR3, DDR4
<b>ODT Assertion Table during Write Accesses</b>	Set the value to on to assert a given ODT signal when writing from a specific bank.	DDR3, DDR4
<b>Derived ODT Matrix for Read Accesses</b>	The matrix is derived based in settings for RTT_NOM and RTT_PARK (DDR4) and the read ODT assertion table.	DDR3, DDR4
<b>Derived ODT Matrix for Write Accesses</b>	The matrix is derived based in settings for RTT_WR, RTT_NOM and the write ODT assertion table.	DDR3, DDR4

### Memory Timing Parameters for Arria 10 EMIF IP

The following tables list the Memory Timing parameters for Arria 10 EMIF IP. Configure these parameters based on your selected memory manufacturer's data sheet.

**Table 7-76: Group: Memory Timing / Parameters Dependent on Speed Bin**

Display Name	Description	Protocol
<b>Speed bin</b>	Speed bin of the memory device used.	All
<b>tIS</b>	Data to DK setup.	QDR IV

Display Name	Description	Protocol
<b>tIS (base)</b>	Address and control setup to CK clock rise.	DDR3, DDR4, LPDDR3, RLDRAM 3
<b>tIS (base) AC level</b>	AC level of tIS (base) for derating purpose.	DDR3, DDR4, LPDDR3
<b>tIH</b>	DK to Data hold.	QDR IV
<b>tIH (base)</b>	Address and control hold after CK clock rise.	DDR3, DDR4, LPDDR3, RLDRAM 3
<b>tIH (base) DC level</b>	DC level of tIH (base) for derating purpose.	DDR3, DDR4, LPDDR3
<b>tdIVW_dj</b>	DQ Rx deterministic jitter.	DDR4
<b>VdiVW_total</b>	Rx Mask voltage - p-p total.	DDR4
<b>tDS (base)</b>	Base specification for data setup to DK.	DDR3, LPDDR3, RLDRAM 3
<b>tDS (base) AC level</b>	AC level of tDS (base) for derating purpose.	DDR3, LPDDR3
<b>tDH (base)</b>	Base specification for data hold from DK.	DDR3, LPDDR3, RLDRAM 3
<b>tDH (base) DC level</b>	DC level of tDH (base) for derating purpose.	DDR3, LPDDR3
<b>tDQSQ</b>	DQS, DQS# to DQ skew, per access.	DDR3, DDR4, LPDDR3
<b>tQKQ_max</b>	QK clock edge to DQ/DINV data edge (in same group).	RLDRAM 3, QDR IV
<b>tQH</b>	<ul style="list-style-type: none"> <li>DQ output hold time from DQS, DQS# (percentage of tCK).</li> <li>DQ/DINV output hold time from QK.</li> </ul>	<ul style="list-style-type: none"> <li>DDR3, DDR4, LPDDR3, RLDRAM 3</li> <li>QDR IV</li> </ul>
<b>tCKDK_max</b>	Clock to Input Data Clock (Max).	RLDRAM 3, QDR IV
<b>tCKDK_min</b>	Clock to Input Data Clock (Min)	RLDRAM 3, QDR IV
<b>tCKQK_max</b>	QK edge to clock edge skew (Max).	RLDRAM 3, QDR IV
<b>tDQSCK</b>	DQS output access time from CK/CK#.	DDR3, DDR4, LPDDR3
<b>tDQSS</b>	First latching edge of DQS to associated clock edge (percentage of tCK).	DDR3, DDR4, LPDDR3
<b>tQSH</b>	DQS Differential High Pulse Width (percentage of tCK).	DDR3, DDR4, LPDDR3

Display Name	Description	Protocol
<b>tDSH</b>	DQS falling edge hold time from CK (percentage of tCK).	DDR3, DDR4, LPDDR3
<b>tWLS</b>	Write leveling setup time from rising CK, CK# crossing to rising DQS, DQS# crossing.	DDR3, DDR4, LPDDR3
<b>tWLH</b>	Write leveling hold time from rising DQS, DQS# crossing to rising CK, CK# crossing.	DDR3, DDR4, LPDDR3
<b>tDSS</b>	DQS falling edge to CK setup time (percentage of tCK).	DDR3, DDR4, LPDDR3
<b>tINIT</b>	Memory initialization time at power-up.	DDR3, DDR4, LPDDR3
<b>tMRD</b>	Load mode register command period.	DDR3, DDR4
<b>tRAS</b>	Active to precharge time.	DDR3, DDR4, LPDDR3
<b>tRCD</b>	Active to read/write time.	DDR3, DDR4, LPDDR3
<b>tRP</b>	Precharge command period.	DDR3, DDR4, LPDDR3
<b>tWR</b>	Write recovery time.	DDR3, DDR4, LPDDR3
<b>tWL</b>	Write Latency.	QDR II/II+ /II+ Xtreme
<b>tRL</b>	Read Latency.	QDR II/II+ /II+ Xtreme
<b>tSA</b>	Address and control setup to K clock rise.	QDR II/II+ /II+ Xtreme
<b>tHA</b>	Address and control hold after K clock rise.	QDR II/II+ /II+ Xtreme
<b>tSD</b>	Data setup to clock (K/K#) rise.	QDR II/II+ /II+ Xtreme
<b>tHD</b>	Data hold after clock (K/K#) rise.	QDR II/II+ /II+ Xtreme
<b>tCQD</b>	Echo clock high to data valid.	QDR II/II+ /II+ Xtreme
<b>tCQDOH</b>	Echo clock high to data invalid.	QDR II/II+ /II+ Xtreme
<b>Internal Jitter</b>	QDRII internal jitter.	QDR II/II+ /II+ Xtreme
<b>tCQH</b>	Output clock (CQ/CQ#) HIGH.	QDR II/II+ /II+ Xtreme
<b>tCCQO</b>	K/K# clock rise to echo clock valid.	QDR II/II+ /II+ Xtreme
<b>tAS</b>	Address to clock setup.	QDR IV
<b>tAH</b>	Clock to address hold.	QDR IV
<b>tCS</b>	Control to clock setup.	QDR IV
<b>tCH</b>	Clock to control hold.	QDR IV

Display Name	Description	Protocol
tMRR	Mode register read command period.	LPDDR3
tMRW	Mode register write command period.	LPDDR3

**Table 7-77: Group: Memory Timing / Parameters Dependent on Speed Bin, Operating Frequency, and Page Size**

Display Name	Description	Protocol
tWTR	Write to read period.	DDR3, LPDDR3
tWTR_L	Write to read period (same bank group).	DDR4
tWTR_S	Write to read period (different bank group).	DDR4
tFAW	Four active window time.	DDR3, DDR4, LPDDR3
tRRD	RAS to RAS delay time.	DDR3, LPDDR3
tRRD_L	RAS to RAS delay time (same bank group).	DDR4
tRRD_S	RAS to RAS delay time (different bank group).	DDR4
tCCD_L	CAS to CAS delay time (same bank group).	DDR4
tCCD_S	CAS to CAS delay time (different bank group).	DDR4
tRTP	Read to precharge time.	DDR3, DDR4, LPDDR3

**Table 7-78: Group: Memory Timing / Parameters Dependent on Density and Temperature**

Display Name	Description	Protocol
tRFC	Auto-refresh command interval.	DDR3, DDR4, LPDDR3
tREFI	Refresh command interval.	DDR3, DDR4, LPDDR3

## Board Timing Parameters for Arria 10 EMIF IP

The following tables list the Board Timing parameters for Arria 10 EMIF IP. Configure these parameters to model the board-level effects in the timing analysis.

For accurate timing results, you must enter board settings parameters that are correct for your PCB.

The IP core supports single and multiple chip-select configurations. Altera has determined the effects on the output signaling of single-rank configurations for certain Altera boards, and included the channel uncertainties in the Quartus Prime software timing models. Because the timing models hold channel uncertainties that are representative of specific Altera boards, you must determine the board-level effects of your board, including any additional channel uncertainty relative to Altera's reference design, and enter

those values into the Board Timing tab in the parameter editor. You can use HyperLynx or a similar simulator to obtain values that are representative of your board.

**Table 7-79: Group: Board Timing / Intersymbol Interference / Crosstalk**

Display Name	Description	Protocol
<b>Use default ISI/crosstalk values</b>	Select to use the default ISI/crosstalk values for your chosen topology, please overwrite these values with values from your own board.	All
<b>Address and command ISI/crosstalk</b>	Address/command window reduction due to ISI/crosstalk.	All
<b>QK/QK# ISI/crosstalk</b>	Read data window reduction due to ISI/crosstalk of QK/QK# signal when driven by memory device during a read.	RLLDRAM 3, QDR IV
<b>Read DQS/DQS# ISI/crosstalk</b>	Read data window reduction due to ISI/crosstalk of DQS/DQS# signal when driven by memory device during a read.	DDR3, DDR4, LPDDR3, RLLDRAM 3
<b>Read DQ ISI/crosstalk</b>	Read data window reduction due to ISI/crosstalk of DQ signal when driven by memory device during a read.	DDR3, DDR4, LPDDR3, QDR IV
<b>Write DQS/DQS# ISI/crosstalk</b>	Write data window reduction due to ISI/crosstalk of DQS/DQS# signal when driven by memory interface during a write.	DDR3, DDR4, LPDDR3
<b>DK/DK# ISI/crosstalk</b>	Write data window reduction due to ISI/crosstalk of DK/DK# signal when driven by memory interface during a write.	RLLDRAM 3, QDR IV
<b>Write DQ ISI/crosstalk</b>	Write data window reduction due to ISI/crosstalk of DQ signal when driven by memory interface during a write.	DDR3, DDR4, LPDDR3, RLLDRAM 3, QDR IV
<b>CQ/CQ# ISI/crosstalk</b>	Read data window reduction due to ISI/crosstalk of CQ/CQ# signal when driven by memory device during a read.	QDR II/II+ /II+ Xtreme
<b>Read Q ISI/Crosstalk</b>	Read data window reduction due to ISI/crosstalk of Q signal when driven by memory device during a read.	QDR II/II+ /II+ Xtreme
<b>K/K# ISI/crosstalk</b>	Write data window reduction due to ISI/crosstalk of K/K# signal when driven by memory interface during a write.	QDR II/II+ /II+ Xtreme
<b>Write D ISI/crosstalk</b>	Write data window reduction due to ISI/crosstalk of D signal when driven by memory interface during a write.	QDR II/II+ /II+ Xtreme

Channel signal integrity is a measure of the distortion of the eye due to intersymbol interference, crosstalk, or other effects. Typically, when going from a single-rank configuration to a multi-rank configuration there is an increase in the channel loss, because there are multiple stubs causing reflections. Although the Quartus Prime software timing models include some channel uncertainty, you must perform your own channel signal integrity simulations and enter the additional channel uncertainty, relative to the reference eye, into the parameter editor.

PCB traces and device packages can introduce skews between signals that can reduce timing margins. Skews between different chip selects can further reduce the timing margin in multiple chip-select topologies. You can configure the following parameters to compensate for these variations. Use the *board skew parameter* tool to calculate board skew values.

For parameter containing delay values, delays should be measured as follows:

- Non-fly-by topology (Balanced Tree)
  - For discrete devices, all the delay (CK, address and command, DQ and DQS) from the FPGA are right to every memory device
  - For UDIMMs, all the delay (CK, address and command, DQ and DQS) from the FPGA to UDIMM connector for every memory device on the UDIMM. If UDIMM delay information is available, calculate delays to every memory device on the UDIMM.
  - For RDIMMs, the address and command and CK delay are from the FPGA to the register on the RDIMM. The DQ and DQS delay are from FPGA to RDIMM connector for every memory device on the RDIMM.
  - For LRDIMMS, the delay from the FPGA to the register on the LRDIMM.
- Fly-by topology
  - For discrete devices, the address and command and CK delay are from the FPGA to the first memory device. The DQ and DQS delay are from FPGA to every memory device.
  - For UDIMMs, the address and command and CK delay are from the FPGA to the UDIMM connector. The DQ and DQS delay are from the FPGA to UDIMM connector for every memory device on the UDIMM.
  - For RDIMMs, the address and command and CK delay are from the FPGA to the register on the RDIMM. The DQ and DQS delay are from FPGA to RDIMM connector for every memory device on the RDIMM.
  - For LRDIMMS, the delay from the FPGA to the buffer on the LRDIMM.

You must ensure the timing margin reported in TimeQuest Report DDR is positive when the board skew parameters are correct for the PCB.

**Table 7-80: Group: Board Timing / Board and Package Skews**

Display Name	Description	Protocol
<b>Package deskewed with board layout (DQS group)</b>	Turn on if your board layout compensates for skew on DQ and DM pins in the FPGA package.	DDR3, DDR4, LPDDR3
<b>Maximum board skew within DQS group</b>	The largest skew between all DQ and DM pins in a DQS group. Enter your board skew only. Package skew will be calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins.	DDR3, DDR4, LPDDR3
<b>Maximum system skew within DQS group</b>	The largest skew between all DQ and DM pins in a DQS group. Enter combined board and package skew. This value affects the read capture and write margins.	LPDDR3
<b>Package deskewed with board layout (QK group)</b>	Check this if your board layout compensates for skew on DQ and DM pins in the FPGA package.	RLLDRAM 3, QDR IV

Display Name	Description	Protocol
<b>Maximum board skew within QK group</b>	The largest skew between all DQ and DM pins in a QK group. Enter your board skew only. Package skew will be calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins.	RDRAM 3, QDR IV
<b>Package deskewed with board layout (address/command bus)</b>	Check this if your board layout compensates for skew on address and command pins in the FPGA package.	All
<b>Maximun system skew within address/command bus</b>	The largest skew between the address and command signals. Enter combined board and package skew.	All
<b>Maximun board skew within address/command bus</b>	The largest skew between the address and command signals. Enter your board skew only. Package skew is calculated automatically, based on the memory interface configuration, and added to this value.	LPDDR3
<b>Average delay difference between DQS and CK</b>	The average delay difference between the DQS signals and the CK signal, calculated by averaging the longest and smallest DQS delay minus the CK delay. Positive values represent DQS signals that are longer than CK signals and negative values represent DQS signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the DQS signals to have appropriate setup and hold margins.	DDR3, DDR4, LPDDR3
<b>Average delay difference between DK and CK</b>	The average delay difference between the DK signals and the CK signal, calculated by averaging the longest and smallest DK delay minus the CK delay. Positive values represent DK signals that are longer than CK signals and negative values represent DK signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the DK signals to have appropriate setup and hold margins.	RDRAM 3, QDR IV
<b>Maximum delay difference between devices</b>	The largest propagation delay on DQ signals between ranks. For example, in a two-rank configuration where you place devices in series there is an extra propagation delay for DQ signals going to and coming back from the furthest device compared to the nearest device.	LPDDR3, RDRAM 3, QDR IV
<b>Maximum delay difference between DIMMs/devices</b>	The largest propagation delay on DQ signals between ranks. For example, in a two-rank configuration where you place DIMMs in different slots there is an extra propagation delay for DQ signals going to and coming back from the furthest DIMM compared to the nearest DIMM.	DDR3, DDR4
<b>Maximum skew between DQS groups</b>	The largest skew between DQS signals in different DQS groups.	DDR3, DDR4, LPDDR3
<b>Maximum skew between DK groups</b>	The largest skew between DK signals in different DK groups.	RDRAM 3, QDR IV

Display Name	Description	Protocol
<b>Average delay difference between address/command and CK</b>	The average delay difference between the address and command signals and the CK signal, calculated by averaging the longest and smallest Address/Command signal delay minus the CK delay. Positive values represent address and command signals that are longer than CK signals and negative values represent address and command signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins.	DDR3, DDR4, LPDDR3, RLDRAM 3, QDR IV
<b>Maximum CK delay to device</b>	The delay of the longest CK trace from the FPGA to any device.	LPDDR3 RLDRAM 3, QDR IV
<b>Maximum CK delay to DIMM/device</b>	The delay of the longest CK trace from the FPGA to any DIMM/device.	DDR3, DDR4
<b>Maximum DK delay to device</b>	The delay of the longest DK trace from the FPGA to any device.	RLDRAM 3, QDR IV
<b>Maximun K delay to device</b>	The delay of the longest K trace from the FPGA to any device.	QDR II/II+ / II+ Xtreme
<b>Maximum DQS delay to DIMM/device</b>	The delay of the longest DQS trace from the FPGA to any DIMM/device.	DDR3, DDR4
<b>Package deskewed with board layout (Q group)</b>	Check this if your board layout compensates for skew on Q pins in the FPGA package.	QDR II/II+ /II+ Xtreme
<b>Maximum board skew within Q group</b>	The largest skew between all Q pins in a Q group. Enter your board skew only. Package skew will be calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins.	QDR II/II+ /II+ Xtreme
<b>Package deskewed with board layout (D group)</b>	Check this if your board layout compensates for skew on D and BWS# pins in the FPGA package.	QDR II/II+ /II+ Xtreme
<b>Maximum board skew within D group</b>	The largest skew between all D and BWS# pins in a D group. Enter your board skew only. Package skew will be calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins.	QDR II/II+ /II+ Xtreme
<b>Average delay difference between address/command and K</b>	The average delay difference between the address and command signals and the K signal, calculated by averaging the longest and smallest Address/Command signal delay minus the K delay. Positive values represent address and command signals that are longer than K signals and negative values represent address and command signals that are shorter than K signals. The Quartus Prime software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins.	QDR II/II+ /II+ Xtreme

Display Name	Description	Protocol
<b>Maximum DQS delay to device</b>	The delay of the longest DQS trace from the FPGA to any device.	LPDDR3
<b>Maximum K delay to device</b>	The delay of the longest K trace from the FPGA to any device.	QDR II/II+ /II+ Xtreme

**Related Information**

[http://www.alterawiki.com/wiki/Measuring\\_Channel\\_Signal\\_Integrity](http://www.alterawiki.com/wiki/Measuring_Channel_Signal_Integrity)

**Controller Parameters for Arria 10 EMIF IP****Table 7-81: Group: Controller**

Display Name	Description	Protocol
<b>Maximum Avalon-MM burst length</b>	Specifies the maximum burst length on the Avalon-MM bus.	QDR II/II+ /II+ Xtreme, QDR IV
<b>Generate power-of-2 data bus widths for Qsys</b>	If enabled, the Avalon data bus width is rounded down to the nearest power-of-2. The width of the symbols within the data bus is also rounded down to the nearest power-of-2. You should only enable this option if you know you will be connecting the memory interface to Qsys interconnect components that require the data bus and symbol width to be a power-of-2. If this option is enabled, you cannot utilize the full density of the memory device.	QDR II/II+ /II+ Xtreme, QDR IV

**Table 7-82: Group: Controller / Low Power Mode**

Display Name	Description	Protocol
<b>Enable Auto Power-Down</b>	Select this option to allow the controller to automatically place the memory device into power-down mode after a specified number of idle controller clock cycles.	DDR3, DDR4, LPDDR3
<b>Auto Power-Down Cycles</b>	The number of idle controller cycles after which the memory device is placed into power-down mode.	DDR3, DDR4, LPDDR3

**Table 7-83: Group: Controller / Efficiency**

Display Name	Description	Protocol
<b>Enable Auto-Precharge Control</b>	Select this option to enable the auto-precharge control on the controller top level. Asserting the auto-precharge control signal while requesting a read or write burst allows you to specify whether or not the controller should close (auto-precharge) the currently open page at the end of the read or write burst, potentially speeding up a future access to a different page of the same bank.	DDR3, DDR4, LPDDR3

Display Name	Description	Protocol
<b>Address Ordering</b>	Controls the mapping between the Avalon addresses and the memory device addresses.	DDR3, DDR4, LPDDR3
<b>Enable Reordering</b>	Enable this option to allow the controller to perform command and data reordering to improve efficiency.	DDR3, DDR4, LPDDR3
<b>Starvation limit for each command</b>	Specifies the number of commands that can be served before a starved command is starved.	DDR3, DDR4, LPDDR3
<b>Enable Command Priority Control</b>	Select this option to enable the user-requested command priority control on the controller top level. The control signal allows you to indicate whether a read or write request should be treated as high priority by the controller. The controller attempts to issue high priority commands sooner, to reduce latency.	DDR3, DDR4, LPDDR3

**Table 7-84: Group: Controller / Configuration, Status and Error Handling**

Display Name	Description	Protocol
<b>Enable Memory-Mapped Configuration and Status Register (MMR) Interface</b>	Enable this option to change or read out the memory timing parameters, memory address size, mode register settings, controller status, and request sideband operations.	DDR3, DDR4, LPDDR3
<b>Enable Error Detection and Correction Logic</b>	Select this option to enable error correction code (ECC) for single-bit error correction and double-bit error detection. Your memory interface must be a multiple of 16, 24, 40, or 72 bits wide in order to be able to use ECC. (ECC is implemented as soft logic.)	DDR3, DDR4
<b>Enable Auto Error Correction</b>	Select this option to allow the controller to perform auto correction when a single-bit error has been detected by the ECC logic.	DDR3, DDR4

**Table 7-85: Group: Controller / Data Bus Turnaround Time**

Display Name	Description	Protocol
<b>Additional read-to-write turnaround time (same rank)</b>	Specifies additional number of idle controller (not DRAM) cycles when switching the data bus from a read to a write within the same logical rank. This can be used to resolve bus contention issues specific to your board topology. The value is added to the default which is auto-calculated. Rely on default setting unless you suspect a problem.	DDR3, DDR4, LPDDR3

Display Name	Description	Protocol
<b>Additional write-to-read turnaround time (same rank)</b>	Specifies additional number of idle controller (not DRAM) cycles when switching the data bus from a write to a read within the same logical rank. This can be used to resolve bus contention issues specific to your board topology. The value is added to the default which is auto-calculated. Rely on default setting unless you suspect a problem.	DDR3, DDR4, LPDDR3
<b>Additional read-to-read turnaround time (different ranks)</b>	Specifies additional number of idle controller (not DRAM) cycles when switching the data bus from a read of one logical rank to a read of another logical rank. This can be used to resolve bus contention issues specific to your board topology. The value is added to the default which is auto-calculated. Rely on default setting unless you suspect a problem.	DDR3, DDR4, LPDDR3
<b>Additional read-to-write turnaround time (different ranks)</b>	Specifies additional number of idle controller (not DRAM) cycles when switching the data bus from a read of one logical rank to a write of another logical rank. This can be used to resolve bus contention issues specific to your board topology. The value is added to the default which is auto-calculated. Rely on default setting unless you suspect a problem.	DDR3, DDR4, LPDDR3

Display Name	Description	Protocol
<b>Additional write-to-write turnaround time (different ranks)</b>	Specifies additional number of idle controller (not DRAM) cycles when switching the data bus from a write of one logical rank to a write of another logical rank. This can be used to resolve bus contention issues specific to your board topology. The value is added to the default which is auto-calculated. Rely on default setting unless you suspect a problem.	DDR3, DDR4, LPDDR3
<b>Additional write-to-read turnaround time (different ranks)</b>	Specifies additional number of idle controller (not DRAM) cycles when switching the data bus from a write of one logical rank to a read of another logical rank. This can be used to resolve bus contention issues specific to your board topology. The value is added to the default which is auto-calculated. Rely on default setting unless you suspect a problem.	DDR3, DDR4, LPDDR3

## Diagnostic Parameters for Arria 10 EMIF IP

**Table 7-86: Group: Diagnostic / Simulation Options**

Display Name	Description	Protocol
<b>Calibration mode</b>	Specifies whether to skip memory interface calibration during simulation, or to simulate the full calibration process. Simulating the full calibration process can take a long time (hours or even days, depending on the width and depth of the memory interface) and is not recommended. Skipping the calibration process results in much faster simulation time, but is only expected to work when the memory model is ideal and the interconnect delays are zero. Abstract PHY is supported only with skip calibration; this option replaces the PHY with a model for fast simulation.	All

Display Name	Description	Protocol
Abstract PHY for fast simulation	Specifies to use Abstract PHY for simulation. Abstract PHY replaces the PHY with a model for fast simulation and can reduce simulation time by 2-3 times. Abstract PHY is available only when you select Skip Calibration. The Abstract PHY is available for DDR3 and DDR4, QDR II, QDR II+, and QDR II+Xtreme, and for QDR-IV and RLDRAM 3, on Arria 10 devices.	DDR3, DDR4, QDR II/II+/ Xtreme, QDR-IV, RLDRAM 3

Table 7-87: Group: Diagnostic / Calibration Debug Options

Display Name	Description	Protocol
<b>EMIF Debug Toolkit/On-Chip Debug Port</b>	Specifies the connectivity of an Avalon slave interface for use by the EMIF Debug Toolkit or user core logic access to calibration data. If you select <b>Disabled</b> , no debug features are enabled. If you select <b>Export</b> , an Avalon slave interface named <code>cal_debug</code> is exported from the IP. To use this interface with the EMIF Debug Toolkit, it is necessary to instantiate and connect an EMIF Debug Interface IP core to it, or connect it to the <code>cal_debug_out</code> interface of another EMIF core. If <b>Add EMIF Debug Interface</b> is selected, an EMIF Debug Interface component containing a JTAG Avalon Master is connected to the debug port. This option allows the core to be accessed using the EMIF Debug Toolkit. You should instantiate only one EMIF Debug Interface per I/O column. Additional EMIF or PHYLite cores can be chained to the first by enabling the <i>Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port</i> option for all cores in the chain, and selecting <i>Export</i> for the <i>Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port</i> option on all cores after the first.	All
<b>Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port</b>	If enabled, exports an Avalon-MM Master interface ( <code>cal_debug_out</code> ) which can be used to connect to the <code>cal_debug</code> interface of other EMIF cores residing in the same I/O column. This option only applies if the EMIF Debug Toolkit/On-Chip Debug Port option is not set to <b>Disabled</b> .	All
<b>Interface ID</b>	ID used to identify this interface within the I/O column. This is used by the EMIF Debug Toolkit and the On-Chip Debug Port. Interface IDs should be unique among EMIF cores within the same I/O column. If the EMIF Debug Toolkit/On-Chip Debug Port is set to <b>Disabled</b> , the interface ID is unused.	All

Display Name	Description	Protocol
<b>Skip address/command leveling calibration</b>	Skip the address and command leveling calibration stage. Address and command leveling attempts to center the memory clock edge against CS# by adjusting delay elements inside the PHY, and then applying the same delay offset to the rest of address and command pins	DDR4, LPDDR3
<b>Skip address/command deskew calibration</b>	Skip the address and command deskew calibration stage. Address and command deskew performs per-bit deskew for the address and command pins.	DDR4, LPDDR3
<b>Skip VREF calibration</b>	Skip the VREF calibration stage.	DDR4, QDR-IV
<b>Use Soft NIOS Processor for On-Chip Debug</b>	Enables the use of a soft Nios processor as a peripheral component to access the On-Chip Debug Port. Only one interface in a column can activate this option.	All

**Table 7-88: Group: Diagnostic / Example Design**

Display Name	Description	Protocol
<b>Number of core clocks sharing slaves to instantiate in the example design</b>	Number of core clock sharing slaves to instantiate in the example design. Applies only if the <b>Core clocks sharing</b> parameter in the <b>General</b> tab is set to <b>Master</b> or <b>Slave</b> .	All
<b>Use a separate RZQ resistor for every sharing interface</b>	Specifies to use a separate RZQ resistor for every sharing interface. Applies only if the <i>Core clocks sharing</i> parameter in the <i>General</i> tab is set to Master or Slave. If this option is unchecked, the RZQ resistor for the master interface is used to calibrate OCT settings for all slave interfaces	All
<b>Enable In-System Sources and Probes</b>	Enable in-system sources and probes of common debug signals, such as calibration status, and example traffic generator per-bit status in the example design.	All

**Table 7-89: Group: Diagnostic / Traffic Generator**

Display Name	Description	Protocol
<b>Use configurable Avalon traffic generator 2.0</b>	Specifies to use the new configurable Avalon traffic generator in the example design.	All
<b>Export Traffic Generator 2.0 configuration interface</b>	Exports the configuration interface for the Traffic Generator 2.0.	All
<b>Data Pattern Length</b>	Specifies the length of the deterministic data pattern, or the $n+1$ PRBS coefficient, that is, a value of 8 for PRBS-7, for use with the Traffic Generator 2.0.	All
<b>Byte Enable Pattern Length</b>	Specifies the length of the deterministic byte enable pattern, or the $n+1$ PRBS coefficient, that is, a value of 8 for PRBS-7, for use with the Traffic Generator 2.0.	All

Display Name	Description	Protocol
<b>Bypass the default traffic pattern</b>	When enabled, this option bypasses the traffic generator 2.0 default pattern after reset.	All
<b>Bypass the user-configured traffic stage</b>	Bypasses the user-configured traffic generator 2.0 pattern after reset. (The traffic generator can still be reconfigured later.) If this option is unchecked, the traffic generator does not assert a pass or fail status until the traffic generator is configured and signaled to start, via the Avalon configuration interface. Configuration can be accomplished by connecting to the traffic generator using the EMIF Debug Toolkit, or by using custom logic connected to the Avalon-MM configuration slave port on the traffic generator. Configuration can also be simulated using the example testbench provided in <code>altera_emif_avl_tg_2_tb.sv</code> .	All
<b>Bypass the traffic generator repeated-writes/repeated-reads test pattern</b>	Bypasses the traffic generator's repeat test stage. Every write and read is repeated several times.	All
<b>Bypass the traffic generator stress pattern</b>	Bypasses the traffic generator's stress pattern.	All

**Table 7-90: Group: Diagnostic / Performance**

Display Name	Description	Protocol
<b>Enable Efficiency Monitor</b>	Adds an efficiency monitor component to the Avalon-MM interface of the memory controller. This component can be accessed by the EMIF Debug Toolkit to view statistics on the interface efficiency.	All

**Table 7-91: Group: Diagnostic / Miscellaneous**

Display Name	Description	Protocol
<b>Use short Qsys interface names</b>	This parameter enables short interface names, which generally provide better usability and consistency with Qsys components. If this parameter is disabled, the Qsys interface names exposed by the IP include the type and direction of the interfaces. Support for long interface names provides backward compatibility, and will be removed in a future release.	All

**Table 7-92: Example design/Available Example Designs**

Display Name	Description	Protocol
Select design	<p>Create a Quartus Prime project that instantiates an external memory interface (with the same configuration that you specify in this GUI) and an example traffic generator. When the Quartus Prime software creates the design, you can:</p> <ul style="list-style-type: none"> <li>• Optionally specify the target device and pin location assignments</li> <li>• Run a full compilation with the Quartus Prime software</li> <li>• Verify timing closure</li> <li>• Test the interface on your board using the programming file that the Quartus Prime assembler generates.</li> </ul>	All

**Table 7-93: Example design/Example Design Files**

Display Name	Description	Protocol
Simulation	Generate an example design for simulation. The generated example designs for various simulators are stored under the <b>sim</b> subdirectory.	All
Synthesis	Generate an example design for synthesis. The generated example design is stored under the <b>qii</b> subdirectory.	All

**Table 7-94: Example design/Generated HDL Format**

Display Name	Description	Protocol
Simulation HDL format	Format of HDL files generated for simulating the example design.	All

**Table 7-95: Example design/Target Development Kit**

Display Name	Description	Protocol
Select board	If you select a development kit with a memory module, the generated example design contains all settings (including fixed pin assignments) to run on the selected board. Always apply IP settings directly from a development kit preset to guarantee results when testing the development kit. You must select a development kit preset to generate a working example design for the respective development kit. If you are generating your example design for a custom board rather than a development kit, select <i>none</i> from the <i>Select board</i> dropdown.	All

## About Memory Presets

Presets help simplify the process of copying memory parameter values from memory device data sheets to the EMIF parameter editor.

For DDRx protocols, the memory presets are named using the following convention:

PROTOCOL-SPEEDBIN LATENCY FORMAT-AND-TOPLOGY CAPACITY (INTERNAL-ORGANIZATION)

For example, the preset named `DDR4-2666U CL18 Component 1CS 2Gb (512Mb x 4)` refers to a DDR4 x4 component rated at the DDR4-2666U JEDEC speed bin, with nominal CAS latency of 18 cycles, one chip-select, and a total memory space of 2Gb. The JEDEC memory specification defines multiple speed bins for a given frequency (that is, DDR4-2666U and DDR4-2666V). You may be able to determine the exact speed bin implemented by your memory device using its nominal latency. When in doubt, contact your memory vendor.

For RLDRAMx and QDRx protocols, the memory presets are named based on the vendor's device part number.

When the preset list does not contain the exact configuration required, you can still minimize data entry by selecting the preset closest to your configuration and then modify parameters as required.

Prior to production you should always review the parameter values to ensure that they match your memory device data sheet, regardless of whether a preset is used or not. Incorrect memory parameters can cause functional failures.

## x4 Mode for Arria 10 External Memory Interface

Non-HPS Arria 10 external memory interfaces support DQ pins-per-DQS group-of-4 (x4 mode) for DDR3 and DDR4 memory protocols.

The following restrictions apply to the use of x4 mode:

- The total interface width is limited to 72 bits.
- You must disable the **Enable DM pins** option.
- For DDR4, you must disable the **DBI** option.

**Note:** x4 mode is not available for Arria 10 EMIF IP for HPS.

## Additional Notes About Parameterizing Arria 10 EMIF IP for HPS

Although Arria 10 EMIF IP and Arria 10 EMIF IP for HPS are similar components, there are some additional requirements necessary in the HPS case.

The following rules and restrictions apply to Arria 10 EMIF IP for HPS:

- Supported memory protocols are limited to DDR3 and DDR4.
- The only supported configuration is the hard PHY with the hard memory controller.
- The maximum memory clock frequency for Arria 10 EMIF IP for HPS may be different than for regular Arria 10 EMIF IP. Refer to the External Memory Interface Spec Estimator for details.
- Only half-rate interfaces are supported.
- Sharing of clocks is not supported.
- The total interface width is limited to a multiple of 16, 24, 40 or 72 bits (with ECC enabled), or a positive value divisible by the number of DQ pins per DQS group (with ECC not enabled). For devices other than 10ASXXXKX40, the total interface width is further limited to a maximum of 40 bits with ECC enabled and 32 bits with ECC not enabled.
- Only x8 data groups are supported; that is, DQ pins-per-DQS group must be 8.
- DM pins must be enabled.

- The EMIF debug toolkit is not supported.
- Ping Pong PHY is not supported.
- The interface to and from the HPS is a fixed-width conduit.
- A maximum of 3 address/command I/O lanes are supported. For example:
  - DDR3
    - For component format, maximum number of chip selects is 2.
    - For UDIMM or SODIMM format:
      - Maximum number of DIMMs is 2, when the number of physical ranks per DIMM is 1.
      - Maximum number of DIMMs is 1, when the number of physical ranks per DIMM is 2.
      - Maximum number of physical ranks per DIMMs is 2, when the number of DIMMs is 1.
    - For RDIMM format:
      - Maximum number of clocks is 1.
      - Maximum number of DIMMs is 1.
      - Maximum number of physical ranks per DIMM is 2.
      - LRDIMM memory format is not supported.
    - DDR4
      - For component format:
        - Maximum number of clocks is 1.
        - Maximum number of chip selects is 2
      - For UDIMM or RDIMM format:
        - Maximum number of clocks is 1.
        - Maximum number of DIMMs is 2, when the number of physical ranks per DIMM is 1.
        - Maximum number of DIMMs is 1, when the number of physical ranks per DIMM is 2.
        - Maximum number of physical ranks per DIMM is 2, when the number of DIMMs is 1.
      - For SODIMM format:
        - Maximum number of clocks is 1.
        - Maximum number of DIMMs is 1.
        - Maximum number of physical ranks per DIMM is 1.
    - Arria 10 EMIF IP for HPS also has specific pin-out requirements. For information, refer to *Planning Pin and FPGA Resources*.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"><li>Modified window duration in descriptions of <i>tDQSCK Delta Medium</i> and <i>tDQSCK Delta Long</i> in the <i>Parameter Descriptions</i> table in <i>Memory Timing Parameters for DDR2, DDR3, and LPDDR2 SDRAM for UniPHY IP</i>.</li><li>Modified description of <i>Enable the Efficiency Monitor and Protocol Checker on the Controller Avalon Interface</i> in the <i>Simulation Options</i> table in <i>Diagnostics for UniPHY IP</i>.</li><li>Removed <i>cal_debug_burstcount</i> from <i>Interface: cal_debug_avalon_slave</i> table in <i>Qsys Interfaces</i>.</li><li>Modified <i>Direction</i> information for entries in <i>Interface: cal_debug_out_avalon_master</i> table in <i>Qsys Interfaces</i>.</li><li>Removed two rows from the <i>Generated Directory Structure and Key Files for Simulation</i> table in <i>Generated Files for Arria 10 External Memory Interface IP</i>.</li><li>Replaced <i>Generated Directory Structure and Key Files for Example Designs</i> table with <i>Generated Directory Structure and Key Files for Example Simulation Designs</i> and <i>Generated Directory Structure and Key Files for Example Synthesis Designs</i> tables in <i>Generated Files for Arria 10 External Memory Interface IP</i>.</li><li>Added entry for <i>Slew rate to Group: I/O / Address/Command</i> and <i>Group: I/O / Memory Clock</i> tables in <i>I/O Parameters for Arria 10 EMIF IP</i>.</li><li>Modified description of <i>DDR3 LRDIMM additional control words</i> entry in <i>Group: Memory Topology / Mode Register Settings</i> table in <i>Memory Topology Parameters for Arria 10 EMIF IP</i>.</li><li>Modified supported protocol information for <i>Enable Error Detection and Correction Logic</i> and <i>Enable Auto Error Correction</i> entries in <i>Group: Controller / Configuration, Status and Error Handling</i> table in <i>Controller Parameters for Arria 10 EMIF IP</i>.</li><li>Minor change to description of <i>Calibration mode</i> in <i>Group: Diagnostic / Simulation Options</i> table in <i>Diagnostic Parameters for Arria 10 EMIF IP</i>.</li><li>Added QDR-IV to supported protocols for <i>Skip VREF calibration</i> in <i>Group: Diagnostic / Calibration Debug Options</i> table in <i>Diagnostic Parameters for Arria 10 EMIF IP</i>.</li><li>Added <i>Calibration address 0</i>, <i>Calibration address 1</i>, and <i>Enable automatic calibration after reset</i> to <i>Group: Diagnostic / Calibration Debug Options</i> table in <i>Diagnostic Parameters for Arria 10 EMIF IP</i>.</li><li>Added <i>Group: Diagnostic / Traffic Generator</i> table to <i>Diagnostic Parameters for Arria 10 EMIF IP</i>. Moved some entries from <i>Group: Diagnostic / Example Design</i> table to new <i>Group: Diagnostic / Traffic Generator</i> table.</li></ul>

Date	Version	Changes
November 2015	2015.11.02	<ul style="list-style-type: none"> <li>Added note to descriptions of <i>Minimum delay difference between CK and DQS</i> and <i>Maximum delay difference between CK and DQS</i> in <i>Board Skew parameters for LPDDR2/DDR2/DDR3 SDRAM</i>.</li> <li>Added text to description of Maximum skew between DQS groups in <i>Board Skew parameters for LPDDR2/DDR2/DDR3 SDRAM</i>.</li> <li>Changed description of <i>emif_usr_clk</i> in the <i>Interface: emif_usr_clk_clock_source</i> table in <i>Qsys Interfaces</i>.</li> <li>Changed description of <i>emif_usr_reset_n</i> in the <i>Interface: emif_usr_reset_reset_source</i> table in <i>Qsys Interfaces</i>.</li> <li>Added <i>Interface: emif_usr_clk_sec_clock_source</i> and <i>Interface: emif_usr_reset_sec_reset_source</i> tables in <i>Qsys Interfaces</i>.</li> <li>Added several options to the <i>Simulation Options</i>, <i>Calibration Debug Options</i>, and <i>Example Design</i> tables in <i>Diagnostic Parameters for Arria 10 EMIF IP</i>.</li> <li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> <li>Added LPDDR3</li> <li>Removed <b>Use address parity bit</b> parameter for QDR IV</li> <li>Removed following DDR4 parameters: <ul style="list-style-type: none"> <li><b>Write CRC enable</b></li> <li><b>DDR4 geardown mode</b></li> <li><b>Per-DRAM addressability</b></li> <li><b>Temperature sensor readout</b></li> <li><b>Write CMD latency for CRC/DM enable</b></li> <li><b>MPR read format</b></li> <li><b>CS to Addr/CMD Latency</b></li> <li><b>Enable DM pins</b></li> <li><b>Addr/CMD persistent error</b></li> <li><b>Write DBI</b></li> <li><b>Read DBI</b></li> </ul> </li> <li>Removed group board timing/slew rates table</li> <li>Removed <b>Maximum system skew within QK group</b> parameters for RLDRAM 3</li> <li>Removed <b>Maximum system skew within Q group</b> and <b>Maximum skew within D group</b> parameters</li> </ul>
May 2015	2015.05.04	Added information to the <i>Description</i> column for the <i>cal_debug_avalon_slave</i> , <i>cal_debug_clk_clock_sink</i> , and <i>cal_debug_out_reset_reset_source</i> tables in the <i>Qsys Interfaces</i> topic.

Date	Version	Changes
December 2014	2014.12.15	<ul style="list-style-type: none"><li>Added MAX 10 device support to the <i>PHY Parameters</i> table in <i>PHY Settings for UniPHY IP</i></li><li>Changed <i>Memory Parameters for LPDDR2, DDR2, and DDR3 SDRAM</i> table to accommodate MAX 10 devices.</li><li>Added <i>Enable Deep Power-Down Controls</i> parameter to <i>Controller Settings</i> table in <i>Controller Settings for UniPHY IP</i> section.</li><li><i>Arria 10 External Memory Interface IP</i> section:<ul style="list-style-type: none"><li>Removed references to Arria 10 devices from <i>Board Settings</i> topic in <i>UniPHY-Based External Memory Interface IP</i> section. Added new <i>Board Timing</i> topic.</li><li>Added QDR IV support to tables in <i>Qsys Interfaces</i> section.</li><li>Removed <i>afi_c</i> from <i>afi_conduit_end</i> table and <i>mem_c</i> from <i>mem_conduit_end</i> table.</li><li>Changed description of <i>global_reset_n</i> signal in the <i>Interface: global_reset_reset_sink</i> table.</li><li>Added <i>Board Timing</i> section and subtopics.</li></ul></li><li>Changed <i>Memory Initialization Options</i> for <i>DDR3</i> table to accommodate MAX 10 devices.</li><li>Replaced <i>Parameter Descriptions</i> table in <i>Memory Timing Parameters for DDR2, DDR3, and LPDDR2 SDRAM for UniPHY IP</i> section with new table including LPDDR2.</li><li>Removed <i>General Settings for Arria 10 EMIF IP</i> section.</li><li><i>Parameterizing Arria 10 External Memory Interface IP</i> section:<ul style="list-style-type: none"><li>Added DDR4 support to tables.</li><li>Changed descriptions of <i>RDIMM/LRDIMM control words</i> and <i>LRDIMM additional control words</i> in <i>Group: Memory Topology / Mode Register Settings</i> table.</li><li>Removed <b>Chip ID Width</b> from <i>Group: Memory Topology / Topology</i> table.</li><li>Added entry for <b>Instantiate two controllers sharing a Ping Pong PHY</b>, and expanded description of <b>Configuration</b>, in the <i>Group: General / Interface</i> table.</li><li>Changed <b>Total interface width</b> entry to <b>DQ width</b> and changed <b>Place ALERT# pin</b> to <b>ALERT# pin placement</b> in <i>Group: Memory Topology / Topology</i> table.</li></ul></li></ul>
August 2014	2014.08.15	<ul style="list-style-type: none"><li>Added notes about Arria 10 EMIF IP to beginning of chapter.</li><li>Added IP Catalog to <i>Design Flows</i> figure.</li><li>Replaced <i>MegaWizard Plug-In Manager Flow</i> with <i>IP Catalog Design Flow</i>.</li><li>Revised <i>Specify Parameters for the Qsys Flow</i> and <i>Completing the Qsys System</i> sections.</li></ul>

Date	Version	Changes
		<ul style="list-style-type: none"> <li>• Added information to description of <code>mem_doff_n</code> in <i>QDR II and QDR II+ SRAM Controller with UniPHY Interfaces</i> table.</li> <li>• Reorganized into separate sections for <i>UniPHY-Based External Memory Interfaces</i> and <i>Arria 10 External Memory Interface IP</i>.</li> <li>• Replaced the following sections with the new <i>Arria 10 EMIF IP Interfaces</i> section: <ul style="list-style-type: none"> <li>• <i>DDR3 Controller with Arria 10 EMIF Interfaces</i></li> <li>• <i>LPDDR2 SDRAM Controller with Arria 10 EMIF Interfaces</i></li> <li>• <i>QDR II/II+ Controller with Arria 10 EMIF Interfaces</i></li> <li>• <i>RDRAM II Controller with Arria 10 EMIF Interfaces</i></li> <li>• <i>RDRAM 3 Controller with Arria 10 EMIF Interfaces</i></li> </ul> </li> <li>• Changed name of <i>Generated Files for Memory Controllers with Arria 10 EMIF IP</i> section to <i>Generated Files for Arria 10 External Memory Interface IP</i>, and revised content.</li> <li>• Revised content of <i>General Settings for Arria 10 EMIF IP</i> section.</li> <li>• Added <i>Parameterizing Arria 10 External Memory Interface IP</i> section.</li> <li>• Revised content of <i>Memory Topology for LPDDR2, DDR3 and DDR4 SDRAM for Arria 10 EMIF IP</i> section.</li> <li>• Added <i>Memory Parameters for QDR IV for Arria 10 EMIF IP</i> section.</li> <li>• Revised <i>Memory Parameters for RDRAM 3 for Arria 10 EMIF IP</i> section.</li> <li>• Added slew rate information for QDR II, QDR II+, and QDR II+ Xtreme to table in <i>Slew Rates for Arria 10 EMIF IP</i> section.</li> <li>• Revised <i>ISI Parameters</i> table in <i>Intersymbol Interference Channel Signal Integrity for UniPHY IP</i> section.</li> <li>• Revised description of fly-by topology for UDIMMs in <i>Board Skews for UniPHY IP</i> section.</li> <li>• Added MAX 10 to <i>Simulation Options</i> table in <i>Diagnostics for UniPHY IP</i> section.</li> <li>• Added note to descriptions of <code>Minimum delay difference between CK and DQS</code> and <code>Maximum delay difference between CK and DQS</code> in <i>Board Skew Parameters for LPDDR2/DDR2/DDR3 SDRAM</i> section.</li> <li>• Revised content of <i>Parameter Descriptions</i> table in <i>Board and Package Skews for LPDDR2/DDR3/DDR4 SDRAM for Arria 10 EMIF IP</i>.</li> <li>• Revised content of <i>Parameter Descriptions</i> table in <i>Board and Package Skews for QDR II, QDR II+, and QDR II+ Xtreme for Arria 10 EMIF IP</i>.</li> <li>• Revised content of <i>Parameter Descriptions</i> table in <i>Board and Package Skews for RDRAM II and RDRAM 3 for Arria 10 EMIF IP</i>.</li> <li>• Revised content of <i>Controller Settings</i> table in <i>Controller Settings for Arria 10 EMIF IP</i>.</li> <li>• Added <i>Diagnostics for Arria 10 EMIF IP</i>.</li> </ul>

Date	Version	Changes
December 2013	2013.12.16	<ul style="list-style-type: none"><li>Removed references to ALTMEMPHY.</li><li>Removed references to HardCopy.</li><li>Removed references to Stratix II devices.</li><li>Removed references to SOPC Builder.</li><li>Added Arria 10 information to <i>Qsys Interface</i>, <i>Generated Files</i>, <i>Parameter Settings</i>, <i>Board Settings</i>, and <i>Controller Settings</i> sections.</li><li>Added descriptions of several registered DIMM parameters to <i>Memory Parameters for LPDDR2, DDR2, and DDR3 SDRAM</i> table.</li><li>Added steps for compiling example project.</li><li>Added clock information to <i>Adding Pins and DQ Group Assignments</i>.</li><li>Updated <i>Intersymbol Interference for UniPHY IP</i> to <i>Intersymbol Interference Channel Signal Integrity for UniPHY IP</i>.</li><li>Added <i>Intersymbol Interference Channel Signal Integrity for Arria 10 EMIF IP</i>.</li></ul>
November 2012	6.0	<ul style="list-style-type: none"><li>Added RLDRAM 3 information.</li><li>Added LPDDR2 information.</li><li>Changed chapter number from 8 to 9.</li></ul>
June 2012	5.0	<ul style="list-style-type: none"><li>Added number of sharing interfaces parameters to Clock Parameters table.</li><li>Added <b>DQ/DQS Package Deskeew</b> and <b>Address/Command Package Deskeew</b> descriptions to Board Skew Parameters table.</li><li>Added equations for multiple boards to several parameter descriptions in Board Skew Parameters table.</li><li>Added Feedback icon.</li></ul>
November 2011	4.0	<ul style="list-style-type: none"><li>Updated <i>Installation and Licensing</i> section.</li><li>Combined <i>Qsys</i> and <i>SOPC Builder Interfaces</i> sections.</li><li>Combined parameter settings for DDR, DDR2, DDR3 SDRAM, QDRII SRAM, and RLDRAM II for both ALTMEMPHY and UniPHY IP.</li><li>Added parameter usage details to <i>Parameterizing Memory Controllers with UniPHY IP</i> section.</li><li>Moved <i>Functional Description</i> section for DDR, DDR2, DDR3 SDRAM, QDRII SRAM, and RLDRAM II to volume 3 of the <i>External Memory Interface Handbook</i>.</li></ul>

Date	Version	Changes
June 2011	3.0	<ul style="list-style-type: none"> <li>Removed references to High-Performance Controller.</li> <li>Updated High-Performance Controller II information.</li> <li>Removed HardCopy III, HardCopy IV E, HardCopy IV GX, Stratix III, and Stratix IV support.</li> <li>Updated <i>Generated Files</i> lists.</li> <li>Added <i>Qsys and SOPC Builder Interfaces</i> section.</li> </ul>
December 2010	2.1	<ul style="list-style-type: none"> <li>Updated Design Flows and Generated Files information.</li> <li>Updated <i>Parameterizing Memory Controllers with UniPHY IP</i> chapter</li> </ul>
July 2010	2.0	<ul style="list-style-type: none"> <li>Added information for new GUI parameters: <b>Controller latency</b>, <b>Enable reduced bank tracking for area optimization</b>, and <b>Number of banks to track</b>.</li> <li>Removed information about IP Advisor. This feature is removed from the DDR/DDR2 SDRAM IP support for version 10.0.</li> </ul>
February 2010	1.3	Corrected typos.
February 2010	1.2	<ul style="list-style-type: none"> <li>Full support for Stratix IV devices.</li> <li>Added timing diagrams for initialization and calibration stages for HPC.</li> </ul>
November 2009	1.1	Minor corrections.
November 2009	1.0	Initial release.

2016.05.02

EMI\_DG



Subscribe



Send Feedback

To simulate your design you require the following components:

- A simulator—The simulator must be any Altera-supported VHDL or Verilog HDL simulator
- A design using one of Altera's external memory IP
- An example driver (to initiate read and write transactions)
- A testbench and a suitable memory simulation model

The Altera External Memory Interface IP is not compatible with the Qsys Testbench System generation feature. Instead, use the simulation example design of your generated IP as a reference for how to create a simulatable design, completed with a memory interface, a memory model, and a traffic generator.

## Memory Simulation Models

There are two types of memory simulation models that you can use:

- Altera-provided generic memory model.

The Quartus® Prime software generates this model together with the example design and this model adheres to all the memory protocol specifications. You can parameterize the generic memory model.

- Vendor-specific memory model.

Memory vendors such as Micron and Samsung provide simulation models for specific memory components that you can download from their websites.

**Note:** Altera does not provide support for vendor-specific memory models.

## Simulation Options

The following simulation options are available with the example testbench to improve simulation speed:

- Full calibration—Calibrates the same way as in hardware, and includes all phase, delay sweeps, and centering on every data bit.

**Note:** Arria 10 EMIF full calibration simulation will be available in a future release of the Quartus Prime software.

- Quick calibration—Calibrates the read and write latency only, skipping per bit deskew. (Not available for Arria 10 EMIF IP.)
- Skip calibration—Provides the fastest simulation. It loads the settings calculated from the memory configuration and enters user mode.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

By default, the UniPHY IP generates abstract PHY, which uses skip calibration regardless of the simulation options that you chose in the parameter editor.

**Note:** For proper simulation of DQS Tracking, you must enable either full calibration or quick calibration.

The following table lists typical simulation times implemented using UniPHY IP. The simulation times in the table are estimates based on average run times of a few example designs. The simulation times for your design may vary depending on the memory interface specifications, simulator, or the system you are using.

**Table 8-1: Typical Simulation Times Using UniPHY IP**

Calibration Mode/Run Time <sup>(7)</sup>	Estimated Simulation Time	
	Small Interface (×8 Single Rank)	Large Interface (×72 Quad Rank)
Full <ul style="list-style-type: none"> <li>• Full calibration</li> <li>• Includes all phase/delay sweeps and centering</li> </ul>	10 minutes	~ 1 day
Quick <ul style="list-style-type: none"> <li>• Scaled down calibration</li> <li>• Calibrate one pin</li> </ul>	3 minutes	4 hours
Skip <ul style="list-style-type: none"> <li>• Skip all calibration, jump to user mode</li> <li>• Preload calculated settings</li> </ul>	3 minutes	20 minutes

**Note to Table:**

1. Uses one loop of driver test. One loop of driver is approximately 600 read or write requests, with burst length up to 64.
2. Simulation times shown in this table are approximate measurements made using Synopsys VCS. Simulation times can vary considerably, depending on the IP configuration, the simulator used, and the computer or server used.

**Table 8-2: Typical Simulation Times Using Arria 10 EMIF IP**

Calibration Mode/Run Time <sup>(7)</sup>	Estimated Simulation Time	
	Small Interface ( $\times 8$ Single Rank)	Large Interface ( $\times 72$ Quad Rank)
Full <ul style="list-style-type: none"> <li>• Full calibration</li> <li>• Includes all phase/delay sweeps and centering</li> </ul>	20 minutes	~ 1 day
Skip <ul style="list-style-type: none"> <li>• Skip all calibration, jump to user mode</li> <li>• Preload calculated settings</li> </ul>	10 minutes	25 minutes
Abstract PHY <ul style="list-style-type: none"> <li>• Replace PHY and external memory model with a single abstract PHY model.</li> <li>• <b>IMPORTANT:</b> <i>External memory model is NOT used in this mode. No I/O switching occurs to the external memory model.</i></li> </ul>	1 minute	5 minutes

Note to Table:

1. Uses one loop of driver test. One loop of driver is approximately 600 read or write requests, with burst length up to 64.
2. Simulation times shown in this table are approximate measurements made using Synopsys VCS. Simulation times can vary considerably, depending on the IP configuration, the simulator used, and the computer or server used.

For more information about steps to follow before simulating, modifying the vendor memory model, and simulation flow for UniPHY IPs, refer to the “Simulation Walkthrough with UniPHY IP”.

#### Related Information

[Simulation Walkthrough with UniPHY IP](#) on page 8-3

## Simulation Walkthrough with UniPHY IP

Simulation of the whole memory interface is a good way to determine the latency of your system. However, the latency found in simulation may be different than the latency found on the board because functional simulation does not take into account board trace delays and different process, voltage, and temperature scenarios.

For a given design on a given board, the latency found may differ by one clock cycle (for full-rate designs) or two clock cycles (for half-rate designs) upon resetting the board. Different boards can also show different latencies even with the same design.

The UniPHY IP supports only functional simulation. Functional simulation is supported at the RTL level and after generating a post-fit functional simulation netlist. The post-fit netlist for designs that contain UniPHY IP is a hybrid of the gate level (for FPGA core) and RTL level (for the external memory interface IP). Altera recommends that you validate the functional operation of your design using RTL simulation, and the timing of your design using TimeQuest Timing Analysis.

For UniPHY-based external memory interfaces, you can perform functional simulation using an example design that is generated with your IP core. The example design files are created in the `\<variation_name>_example_design` directory.

You can use the IP functional simulation model with any Altera-supported VHDL or Verilog HDL simulator.

After you have generated the memory IP, view the README.txt file located in the `\<variation_name>_example_design\simulation` directory for instructions on how to generate the simulation example design for Verilog HDL or VHDL. The **README.txt** file also explains how to run simulation using the ModelSim-Altera software. Altera provides simulation scripts for the Mentor, Cadence, Aldec, and Synopsys simulators; however, detailed instructions on how to perform simulation using these third party simulators are not provided.

## Simulation Scripts

The Quartus Prime software generates three simulation scripts during project generation for four different third party simulation tools—Cadence, Synopsys, Aldec, and Mentor.

The simulation scripts reduce the number of files that you need to compile separately before simulating a design. These scripts are located in three separate folders under the `<project_directory>\<variation_name>_sim` directory, each named after the names of the simulation tools. The example designs also provide equivalent scripts after you run the **.tcl** script from the project located in the `\<variation_name>_example_design\simulation` directory.

The order of the files in the simulation scripts is important. Ensure that you maintain the files in order, to avoid error messages and warning messages during simulation. If you choose not to use the Altera-generated simulation scripts in your simulation environment, you must maintain the specified file order when compiling the memory controller with the user-generated simulation script.

## Preparing the Vendor Memory Model

You can replace the Altera-supplied memory model with a vendor-specific memory model. In general, you may find vendor-specific models to be standardized, thorough, and well supported, but sometimes more complex to setup and use.

**Note:** Altera does not provide support for vendor-specific memory models. If you do want to replace the Altera-supplied memory model with a vendor-supplied memory model, you should observe the following guidelines:

- Ensure that you have the correct vendor-supplied memory model for your memory device.
- Disconnect all signals from the default memory model and reconnect them to the vendor-supplied memory model.
- If you intend to run simulation from the Quartus Prime software, ensure that the **.qip** file points to the vendor-supplied memory model.

When you are using a vendor-supplied memory model instead of the generated functional simulation model, you must modify the vendor memory model and the testbench files by following these steps:

1. Obtain and copy the vendor memory model to the `\<variation_name>_example_design\simulation\<variation_name>_sim\ submodules` directory. For example, obtain the `ddr2.v` and `ddr2_parameters.vh` simulation model files from the Micron website and save them in the directory.
  - The auto-generated generic SDRAM model may be used as a placeholder for a specific vendor memory model.
  - Some vendor DIMM memory models do not use data mask (DM) pin operation, which can cause calibration failures. In these cases, use the vendor's component simulation models directly.
2. Open the vendor memory model file in a text editor and specify the speed grade and device width at the top of the file. For example, you can add the following statements for a DDR2 SDRAM model file:

```
'define sg25
```

```
'define x8
```

The first statement specifies the memory device speed grade as -25 (for 400 MHz operation). The second statement specifies the memory device width per DQS.

3. Check that the following statement is included in the vendor memory model file. If not, include it at the top of the file. This example is for a DDR2 SDRAM model file:

```
`include "ddr2_parameters.vh"
```

4. Save the vendor memory model file.
5. Open the simulation example project file `<variation_name>_example_sim.qpf`, located in the `<variation_name>_example_design\simulation` directory.
6. On the Tools menu, select **TCL scripts** to run the `generate_sim_verilog_example_design.tcl` file, in which generates the simulation example design.
7. To enable vendor memory model simulation, you have to include and compile the vendor memory model by adding it into the simulation script. Open the `.tcl` script, `msim_setup.tcl`, located in the `<variation_name>_example_design\simulation\verilog\mentor` directory in the text editor. Add in the following line in the '# Compile the design files in correct order' section:

```
vlog      +incdir+$QSYS_SIMDIR/submodules/  
          "$QSYS_SIMDIR/submodules/<vendor_memory>.v"  
          -work <variation_name>_example_sim_work
```

8. Open the simulation example design, `<variation_name>_example_sim.v`, located in the `<variation_name>_example_design\simulation\verilog` directory in a text editor and delete the following module:

```
alt_mem_if_<memory_type>_mem_model_top_<memory_type>_mem_if_dm_pins_en_mem_if_dqs  
n_en
```

**Note:** The actual name of the pin may differ slightly depending on the memory controller you are using.

9. Instantiate the downloaded memory model and connect its signals to the rest of the design.
10. Ensure that the ports names and capitalization in the memory model match the port names and capitalization in the testbench.

**Note:** The vendor memory model may use different pin names and capitalization than the generated functional simulation model.

11. Save the testbench file.

The original instantiation may be similar to the following code:

```

alt_mem_if_ddr2_mem_model_top_mem_if_dm_pins_en_mem_if_dqsn_en #(
    .MEM_IF_ADDR_WIDTH          (13),
    .MEM_IF_ROW_ADDR_WIDTH      (12),
    .MEM_IF_COL_ADDR_WIDTH      (8),
    .MEM_IF_CS_PER_RANK         (1),
    .MEM_IF_CONTROL_WIDTH       (1),
    .MEM_IF_DQS_WIDTH          (1),
    .MEM_IF_CS_WIDTH            (1),
    .MEM_IF_BANKADDR_WIDTH      (3),
    .MEM_IF_DQ_WIDTH            (8),
    .MEM_IF_CK_WIDTH            (1),
    .MEM_IF_CLK_EN_WIDTH        (1),
    .DEVICE_WIDTH                (1),
    .MEM_TRCD                   (6),
    .MEM_TRTP                   (3),
    .MEM_DQS_TO_CLK_CAPTURE_DELAY (100),
    .MEM_IF_ODT_WIDTH           (1),
    .MEM_MIRROR_ADDRESSING_DEC  (0),
    .MEM_REGDIMM_ENABLED         (0),
    .DEVICE_DEPTH                (1),
    .MEM_INIT_EN                 (0),
    .MEM_INIT_FILE               (""),
    .DAT_DATA_WIDTH              (32)
) m0 (
    .mem_a      (e0_memory_mem_a),      // memory.mem_a
    .mem_ba     (e0_memory_mem_ba),     // .mem_ba
    .mem_ck     (e0_memory_mem_ck),     // .mem_ck
    .mem_ck_n   (e0_memory_mem_ck_n),   // .mem_ck_n
    .mem_cke    (e0_memory_mem_cke),    // .mem_cke
    .mem_cs_n   (e0_memory_mem_cs_n),   // .mem_cs_n
    .mem_dm     (e0_memory_mem_dm),     // .mem_dm
    .mem_ras_n  (e0_memory_mem_ras_n),  // .mem_ras_n
    .mem_cas_n  (e0_memory_mem_cas_n),  // .mem_cas_n
    .mem_we_n   (e0_memory_mem_we_n),   // .mem_we_n
    .mem_dq     (e0_memory_mem_dq),     // .mem_dq
    .mem_dqs    (e0_memory_mem_dqs),    // .mem_dqs
    .mem_dqs_n  (e0_memory_mem_dqs_n),  // .mem_dqs_n
    .mem_odt    (e0_memory_mem_odt)     // .mem_odt
);

```

Replace the original code with the following code:

```

ddr2 memory_0 (
    .addr (e0_memory_mem_a), // memory.mem_a
    .ba (e0_memory_mem_ba), // .mem_ba
    .clk (e0_memory_mem_ck), // .mem_ck
    .clk_n (e0_memory_mem_ck_n), // .mem_ck_n
    .cke (e0_memory_mem_cke), // .mem_cke
    .cs_n (e0_memory_mem_cs_n), // .mem_cs_n
    .dm_rdqs (e0_memory_mem_dm), // .mem_dm
    .ras_n (e0_memory_mem_ras_n), // .mem_ras_n
    .cas_n (e0_memory_mem_cas_n), // .mem_cas_n
    .we_n (e0_memory_mem_we_n), // .mem_we_n
    .dq (e0_memory_mem_dq), // .mem_dq
    .dqs (e0_memory_mem_dqs), // .mem_dqs
    .rdqs_n (), // .mem_dqs_n
    .dqs_n (e0_memory_mem_dqs_n), // .mem_dqs_n
    .odt (e0_memory_mem_odt) // .mem_odt);

```

If you are interfacing with a DIMM or multiple memory components, you need to instantiate all the memory components in the simulation file.

## Functional Simulation with Verilog HDL

Altera provides simulation scripts for you to run the example design. The simulation scripts are for Synopsys, Cadence, Aldec, and Mentor simulators.

The simulation scripts are located in the following main folder locations:

Simulation scripts in the simulation folders are located as follows:

- `<variation_name>_example_design\simulation\verilog\mentor\msim_setup.tcl`
- `<variation_name>_example_design\simulation\verilog\synopsys\vcs\vcs_setup.sh`
- `<variation_name>_example_design\simulation\verilog\synopsys\vcsmx\vcsmx_setup.sh`
- `<variation_name>_example_design\simulation\verilog\aldec\rivierapro_setup.tcl`
- `<variation_name>_example_design\simulation\verilog\cadence\ncsim_setup.sh`

Simulation scripts in the `<>_sim_folder` are located as follows:

- `<variation_name>_sim\mentor\msim_setup.tcl`
- `<variation_name>_sim\cadence\ncsim_setup.sh`
- `<variation_name>_sim\synopsys\vcs\vcs_setup.sh`
- `<variation_name>_sim\vcsmx\vcsmx_setup.sh`
- `<variation_name>_sim\aldec\rivierapro_setup.tcl`

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Mentor Graphics ModelSim® and QuestaSim Support* chapter in volume 3 of the Quartus Prime Handbook.

### Related Information

[Mentor Graphics ModelSim and QuestaSim Support](#)

## Functional Simulation with VHDL

The UniPHY VHDL file set is provided for customers who want to generate the top-level RTL instance of their UniPHY cores in VHDL.

Prior to Quartus Prime version 15.1, the VHDL fileset was composed entirely of VHDL files. Beginning with Quartus Prime version 15.1, only the top-level IP instance file is guaranteed to be written in VHDL; submodules can still be written in Verilog/SystemVerilog (encrypted or plaintext), or in VHDL. Note that the ModelSIM Altera Edition is no longer restricted to a single HDL language, as of version 15.1; however, some files may still be encrypted in order to be excluded from the maximum unencrypted module limit of this tool.

Because the VHDL fileset consists of both VHDL and Verilog files, you must follow certain mixed-language simulation guidelines. The general guideline for mixed-language simulation is that you must always link the Verilog files (whether encrypted or not) against the Verilog version of the Altera libraries, and the VHDL files (whether simgen-generated or pure VHDL) against the VHDL libraries.

Altera provides simulation scripts for you to run the example design. The simulation scripts are for Synopsys, Cadence, Aldec, and Mentor simulators. These simulation scripts are located in the following main folder locations:

Simulation scripts in the simulation folders are located as follows:

- `<variation_name>_example_design\simulation\vhdl\mentor\msim_setup.tcl`
- `<variation_name>_example_design\simulation\vhdl\synopsys\vcsim\vcsmx\vcsmx_setup.sh`
- `<variation_name>_example_design\simulation\vhdl\cadence\ncsim_setup.sh`
- `<variation_name>_example_design\simulation\vhdl\aldec\rivierapro_setup.tcl`

Simulation scripts in the `<>_sim_folder` are located as follows:

- `<variation_name>_sim\mentor\msim_setup.tcl`
- `<variation_name>_sim\cadence\ncsim_setup.sh`
- `<variation_name>_sim\synopsys\vcsim\vcsmx\vcsmx_setup.sh`
- `<variation_name>_sim\aldec\rivierapro_setup.tcl`

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Mentor Graphics ModelSim® and QuestaSim Support* chapter in volume 3 of the Quartus Prime Handbook.

#### Related Information

[Mentor Graphics ModelSim and QuestaSim Support](#)

## Simulating the Example Design

This topic describes how to simulate the example design in Cadence, Synopsys, Mentor, and Aldec simulators.

To simulate the example design in the Quartus Prime software using the Cadence simulator, follow these steps:

1. At the Linux shell command prompt, change directory to `<name>_example_design\simulation\<verilog/vhdl>\cadence`
2. Run the simulation by typing the following command at the command prompt:

```
sh ncsim_setup.sh
```

To simulate the example design in the Quartus Prime software using the Synopsys simulator, follow these steps:

1. At the Linux shell command prompt, change directory to `<name>_example_design\simulation\<verilog/vhdl>\synopsys\vcsim`
2. Run the simulation by typing the following command at the command prompt:

```
sh vcsmx_setup.sh
```

To simulate the example design in the Quartus Prime software using the Mentor simulator, follow these steps:

1. At the Linux or Windows shell command prompt, change directory to **<name>\_example\_design\simulation\<verilog/vhdl>\mentor**
2. Execute the **msim\_setup.tcl** script that automatically compiles and runs the simulation by typing the following command at the Linux or Windows command prompt:

```
vsim -do run.do
```

or

Type the following command at the ModelSim command prompt:

```
do run.do
```

To simulate the example design in the Quartus Prime software using the Aldec simulator, follow these steps:

1. At the Linux or Windows shell command prompt, change directory to **<name>\_example\_design\simulation\<verilog/vhdl>\aldec**
2. Execute the **rivierapro\_setup.tcl** script that automatically compiles and runs the simulation by typing the following command at the Linux or Windows command prompt:  
`vsim -do rivierapro.tcl`
3. To compile and elaborate the design after the script loads, type `ld_debug`.
4. Type `run -all` to run the simulation.

For more information about simulation, refer to the *Simulating Altera Designs* chapter in volume 3 of the Quartus Prime Handbook.

If your Quartus Prime project appears to be configured correctly but the example testbench still fails, check the known issues on the Knowledge Database page of the Altera website before filing a service request.

#### Related Information

- [Simulating Altera Designs](#)
- [Knowledge Database](#)

## UniPHY Abstract PHY Simulation

UniPHY IP generates both synthesizable and abstract models for simulation, with the abstract model as default. The UniPHY abstract model replaces the PLL with simple fixed-delay model, and the detailed models of the hard blocks with simple cycle-accurate functional models.

Full calibration mode cannot be used with abstract models, which is the default model type for all devices except Arria V and Cyclone V. In addition to enabling full calibration during generation, you must also disable the use of abstract models by modifying the generated simulation scripts as described below. For VHDL, the UniPHY abstract model is the only option because you cannot switch to regular simulation model. The PLL frequencies in simulation may differ from the real time simulation due to pico-second timing rounding.

However, you can switch to regular simulation models for Verilog HDL language. The full and quick calibration modes are available for regular simulation models.

Add an additional command line to the compilation script for the two relevant files to enable regular PHY simulation:

```
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
```

The two relevant files are:

- In *<variation\_name>\_example\_design/simulation/verilog/submodules*:

**<variation\_name>\_example\_sim\_e0\_if0\_p0.sv**

and

**<variation\_name>\_example\_sim\_e0\_if0\_pll0.sv**

or

- In *<variation\_name>\_sim/submodules*:

**<variation\_name>\_p0.sv**

and

**<variation\_name>\_pll0.sv**

To switch to regular simulation models for the Verilog HDL language on the example simulation design, follow the appropriate steps for your simulator:

- For the Mentor simulator, edit the **msim\_setup.tcl** file as follows:

```
vlog
- sv "$QSYS_SIMDIR/submodules/<variation_name>_example_sim_e0_if0_p0.sv"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
-work <variation_name>_example_sim_work

vlog - sv
"$QSYS_SIMDIR/submodules/<variation_name>/_example_sim_e0_if0_pll0.sv"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
-work <variation_name>_example_sim_work
```

- For the Cadence simulator, edit the **ncsim\_setup.sh** file as follows:

```
ncvlog
- sv "$QSYS_SIMDIR/submodules/<variation_name>_example_sim_e0_if0_p0.sv"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
-work <variation_name>_example_sim_work
- cdslib ./cds_libs/skip_example_sim_work.cds.lib

ncvlog - sv
"$QSYS_SIMDIR/submodules/<variation_name>_example_sim_e0_if0_pll0.sv"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
-work <variation_name>_example_sim_work
- cdslib ./cds_libs/<variation_name>_example_sim_work.cds.lib
```

- For the Synopsys simulator, edit the **vscmx\_setup.sh** file as follows:

```
vlogan +v2k - sverilog
"$QSYS_SIMDIR/submodules/<variation_name>_example_sim_e0_if0_p0.sv"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
- work <variation_name>_example_sim_work

vlogan +v2k - sverilog
"$QSYS_SIMDIR/submodules/<variation_name>_example_sim_e0_if0_pll0.sv"
+define+ALTERA_ALT_MEM_IF_PHY_FAST_SIM_MODEL=0
- work <variation_name>_example_sim_work
```

If you use the UniPHY abstract model, the simulation is two times faster in magnitude if compared to the real simulation model. Instantiating a standalone UniPHY IP in your design further improves the simulation time if you use a half-rate controller with UniPHY or a larger memory DQ width.

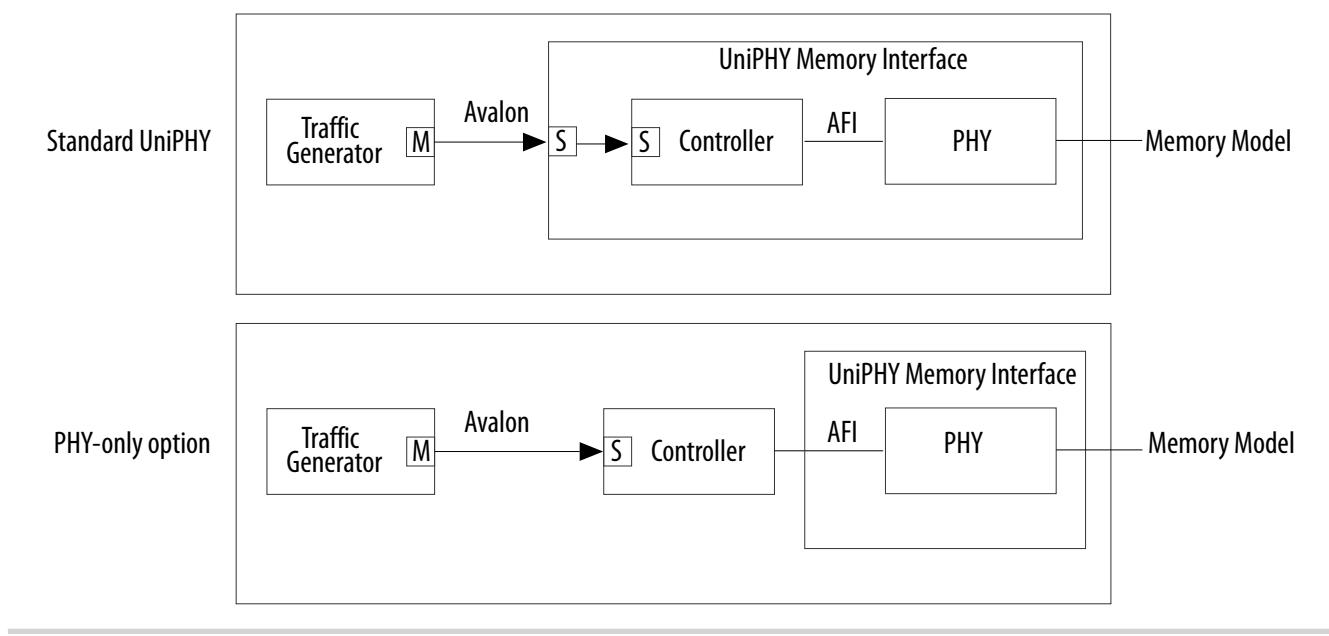
## PHY-Only Simulation

To enable PHY-only simulation in the parameter editor, under **PHY Settings** tab, in the FPGA section, turn on **Generate PHY only**. This setting also applies to designs using Qsys. This option allows you to replace the Altera high-performance memory controllers with your own custom controller.

When you are using a standard UniPHY memory interface, by default, the parameter editor generates an external memory interface with a controller and a PHY. The controller and PHY are connected internally with the Altera PHY interface (AFI). The memory interface has an Avalon slave port that connects to the controller to allow communication from the user logic. When you turn on the PHY-only option, the parameter editor generates the PHY without the controller. In this case, the PHY is accessed via the AFI port, which can be externally connected to a custom controller. In the example design, a controller is instantiated externally to the memory interface. This provides a fully functional example design and demonstrates how to connect the controller to manage the transactions from the traffic generator.

The following figure shows the difference in the UniPHY memory interface when the PHY-only option is enabled.

**Figure 8-1: PHY-only Option**



## Post-fit Functional Simulation

The post-fit functional simulation does not work for the UniPHY IP core because of the following inherent problems:

- The UniPHY sample 'X's during calibration, in which causes an issue during timing simulation
- Some internal transfers that are 0-cycle require delays to properly function in a post-fit netlist

To enable functional simulation for a design that uses UniPHY IP core, a quasi-post-fit scheme is implemented. This scheme allows gate-level simulation of the full design (excluding the UniPHY IP), while you use RTL simulation for the UniPHY IP. The quasi-post-fit scheme involves partitioning blocks in the EMIF and swapping them with simulation RTL. With this workaround the memory interface is partially post-fit RTL and partially premap RTL, therefore the simulation flow is not impeded.

Gate simulation for the hard memory controller is not supported.

## Running Post-fit Simulation

Assuming that the UniPHY IP has been generated and inserted in some larger design, follow these steps to run post-fit simulation:

1. In the Quartus Prime software, set up a project that contains a UniPHY IP core.
2. On the Assignments menu, click **Assignment Editor**.
3. In the assignment editor, add the global assignment `VERILOG_MACRO` and set the value to `SYNTH_FOR_SIM=1`.
4. On the Assignments menu, click **Settings**.
5. In the **Category** list, under **EDA Tools Settings**, select **Simulation**.
6. On the Simulation page, select a tool name (for example, ModelSim-Altera).
7. In the **Format for output netlist** list, select a HDL language.
8. In the **Output directory** box, type or browse to the location where you want output files saved.
9. Click **More EDA Netlist Writer Settings** to choose from a list of other options.
10. Set the value for **Maintain hierarchy** to **PARTITION\_ONLY**, and click **OK**.
11. Elaborate the project. On the Processing menu, select **Start** and click **Start Hierarchy Elaboration**.
12. In the Project Navigator window, click the **Hierarchy** tab. In the **Entity** box, locate the instances for the following devices:
  - a. For instances in Stratix III, Stratix IV, Arria II GX, Arria II GZ , click the + icon to expand the following top-level design entities, right-click on the lower-level entities, select **Design Partition**, and click **Set as Design Partition**:
    - <hierarchy path to UniPHY top-level>\<name>\_if0:if0\<name>\_if0\_p0:p0
    - <hierarchy path to UniPHY top-level>\<name>\_if0:if0\<name>\_if0\_s0:s0
  - b. For instances in Arria V or Stratix V, click the + icon to expand the following top-level design entity, right-click on the lower-level entities, select **Design Partition**, and click **Set as Design Partition**:
    - <hierarchy path to UniPHY top-level>\<name>\_if0:if0\<name>\_if0\_s0:s0
- For instances of hard memory interfaces in Arria V, no design partition is necessary.
13. In the **Design Partitions Window**, ensure that the netlist type value of the design partitions listed in Step12 a and 12b are set to **Post-synthesis**.
14. On the Processing menu, select **Start** and click **Start Analysis and Synthesis**.
15. Run the Pin assignments script. To run the pin assignment script, follow these steps:

- a. On the **Tools** menu, click **TCL Scripts**.
  - b. In the **Libraries** list, locate the <name>\_pin\_assignment.tcl.
  - c. Click **Run**.
16. On the Processing menu, select **Start** and click **Partition Merge**.
17. On the Processing menu, select **Start** and click **Start Fitter**.
18. On the Processing menu, select **Start** and click **Start EDA netlist writer**.
19. The output post-fit netlist is located in the directory you chose in Step 8.
20. Assume that the netlist filename is **dut.vo** (or **dut.vho** for VHDL). Replace the instance of the partitioned modules (specified in step 12) in **dut.vo** and instantiate the original instance of the RTL. As a result, the RTL of those modules will simulate correctly instead of the the post-fit netlist. For example, you can delete the definition of the <name>\_if0\_s0 (and <name>\_if0\_p0, if appropriate) modules in the post-fit netlist, and ensure that your simulator compiles the post-fit netlist and all the UniPHY RTL in order to properly link these modules for simulation.
- (This step does not apply to hard memory interfaces on Arria V devices.)
21. To match the post-fit netlist instantiation of s0 (and p0, if appropriate) with the original RTL module definition (specified in step 12), you must also account for three device input ports that are added to the post-fit netlist. The easiest way to do this is to delete the following three connections from the s0 (and p0, if appropriate) instances in the post-fit netlist:
- .devpor(devpor)
  - .devclrn(devclrnn)
  - .devoe(devpoe)
- (This step does not apply to hard memory interfaces on Arria V devices.)
22. For Stratix V the <name>\_if0\_s0 instance in the post-fit netlist will also have a connection .QIC\_GND\_PORT( <wire name> ) that you must delete because it does not match with the original RTL module.
- (This step does not apply to hard memory interfaces on Arria V devices.)
23. Set up and run your simulator.

## Simulation Issues

When you simulate an example design in the ModelSim, you might see the following warnings, which are expected and not harmful:

```
# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (1 or 1) does not match connection size (7) for port
'local_size'.

#           Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (9 or 9) does not match connection size (1) for port
'ctl_cal_byte_lane_sel_n'.

#           Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (18 or 18) does not match connection size (1) for port
'afi_doing_read'.

#           Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst
```

```

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (2 or 2) does not match connection size (1) for port
'afi_rdata_valid'.

#           Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (112 or 112) does not match connection size (1) for port
'bank_information'.

#           Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_controller_phy.sv(402
): [PCDPC] - Port size (8 or 8) does not match connection size (1) for port
'bank_open'.

#           Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst
/bank_timer_wrapper_inst/bank_timer_inst

# ** Warning: (vsim-3722)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_alt_ddrx_bank_timer_
wrapper.v(1191): [TFMPC] - Too few port connections. Expected 127, found 126.

#           Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_inst
/bank_timer_wrapper_inst/bank_timer_inst

# ** Warning: (vsim-3015)
D:/design_folder/iptest10/simulation/uniphy_s4/rtl/uniphy_s4_alt_ddrx_bank_timer_
wrapper.v(1191): [TFMPC] - Missing connection for port 'wr_to_rd_to_pch_all'.

#           Region:
/uniphy_s4_example_top_tb/dut/mem_if/controller_phy_inst/alt_ddrx_controller_
inst/bank_timer_wrapper_inst/rank_monitor_inst

# ** Warning: (vsim-8598) Non-positive replication multiplier inside concat.
Replication will be ignored

Warning-[OSPA-N] Overriding same parameter again

/p/eda/acd/altera/quartusII/10.1/quartus/eda/sim_lib/synopsys/stratixv_atoms_
ncrypt.v, 8499

Warning-[ZONMCM] Zero or negative multiconcat multiplier
../quartus_stratix5/ddr3_ctlr_sim/ddr3_ctlr_sequencer.sv, 916

Zero or negative multiconcat multiplier is found in design. It will be
replaced by 1'b0.

Source info: {INIT_COUNT_WIDTH {1'b0}}
```

Warning-[PCWM-W] Port connection width mismatch  
 ../quartus\_stratix5/ddr3\_ctlr\_sim/ddr3\_ctlr\_sequencer\_cpu.v, 2830

"the\_sequencer\_cpu\_nios2\_oci\_itrace"

The following 38-bit expression is connected to 16-bit port "jdo" of module "ddr3\_ctlr\_sequencer\_cpu\_nios2\_oc\_i\_trace", instance "the\_sequencer\_cpu\_nios2\_oc\_i\_trace".

Expression: jdo  
use +lint=PCWM for more details

## Simulation Walkthrough with Arria 10 EMIF IP

Simulation of the whole memory interface is a good way to determine the latency of your system. However, the latency found in simulation may be different than the latency found on the board because functional simulation does not take into account board trace delays and different process, voltage, and temperature scenarios.

For a given design on a given board, the latency found may differ by one clock cycle (for full-rate designs) or two clock cycles (for half-rate designs) upon resetting the board. Different boards can also show different latencies even with the same design.

The Arria 10 EMIF IP supports only functional simulation. Functional simulation is supported at the RTL level and after generating a post-fit functional simulation netlist. The post-fit netlist for designs that contain Arria 10 EMIF IP is a hybrid of the gate level (for FPGA core) and RTL level (for the external memory interface IP). Altera recommends that you validate the functional operation of your design using RTL simulation, and the timing of your design using TimeQuest Timing Analysis.

For Arria 10 EMIF IP, you can perform functional simulation of an example design that is generated with your IP core. The example design files are created in the `\<variation_name>_example_design` directory.

You can use the IP functional simulation model with any Altera-supported VHDL or Verilog HDL simulator.

After you have generated the memory IP, view the `README.txt` file located in the `\<variation_name>_example_design` directory for instructions on how to generate the simulation example design for Verilog HDL or VHDL. Simulation filesets for both Verilog HDL and VHDL are located in `\<variation_name>_example_design\sim`. The `README.txt` file also explains how to run simulation using the ModelSim-Altera software. Altera provides simulation scripts for the Mentor, Cadence, Aldec, and Synopsys simulators; however, detailed instructions on how to perform simulation using these third party simulators are not provided.

## Skip Calibration Versus Full Calibration

Calibration must occur shortly after the memory device is initialized, to compensate for uncertainties of the hardware system, including silicon PVT variation, circuit board trace delays, and skewed arrival times. Such variations are usually not present in an RTL simulation environment, therefore there are two options for how the calibration algorithm behaves during simulation: Skip Calibration mode (which is the default), and Full Calibration mode.

### Skip Calibration Mode

In Skip Calibration mode, the calibration processor assumes an ideal hardware environment, where PVT variations, board delays, and trace skews are all zero. Instead of running the actual calibration routine, the calibration processor calculates the expected arrival time of read data based on the memory latency values that you provide, thus reducing much simulation processing. Skip calibration mode is recommended for use during system development, because it allows you to focus on interacting with the controller and optimizing your memory access patterns, thus facilitating rapid RTL development.

## Full Calibration Mode

Full Calibration mode simulates every stage of the calibration algorithm immediately after memory device initialization. Because the calibration algorithm processes each data group sequentially and each pin in each group individually, simulation time increases with the number of groups and data pins in your interface. You can observe how the calibration algorithm compensates for various delays in the system by incorporating your own board delay model based on trace delays from your PCB design tools. Due to the large simulation overhead, Full Calibration simulation mode is not recommended for rapid development of IP cores.

## VHDL Support

VHDL support for mixed-language simulators is implemented by generating the top-level wrapper for the core in VHDL, while all submodules are provided as clear text SystemVerilog files.

A set of precompiled device libraries is provided for use with the ModelSIM Altera Edition single-language simulator which is supplied with the Quartus Prime software. Submodules normally provided as cleartext SystemVerilog files are encrypted using IEEE Verilog encryption for ModelSIM AE.

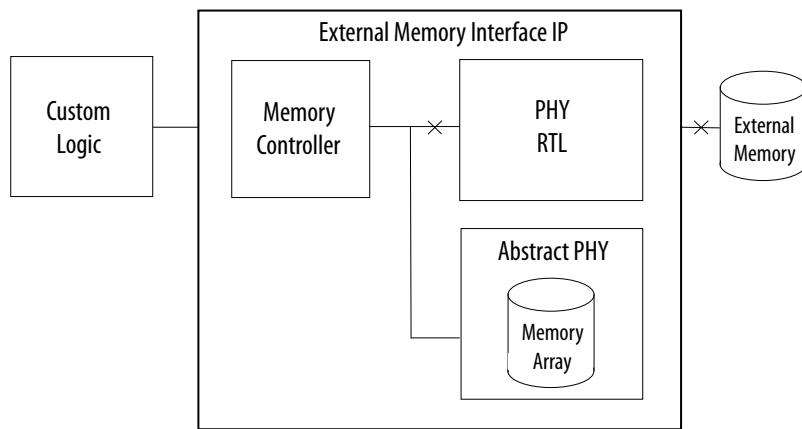
## Arria 10 Abstract PHY Simulation

The abstract PHY is a simulation model of the EMIF PHY that can decrease simulation time by 3-10 times. The abstract PHY replaces the lane and the external memory model with a single model containing an internal memory array. No switching of the I/Os to the external memory model occurs when simulating with the abstract PHY.

Abstract PHY reduces simulation time by two mechanisms:

- The Nios processor has been disabled and replaced by HDL forces that are applied at the beginning of simulation. The HDL forces are a minimum set of configuration registers that allow the EMIF to be configured properly for simulation. The write and read latency values applied by the HDL forces are not representative of the post-calibration values applied to the EMIF running on hardware. However, as long as the customer logic is avalon- and afi-compliant, these values allow for successful RTL simulation.
- The abstract PHY eliminates the need for full-speed clocks and therefore simulation of the abstract PHY does not require full-speed clock simulation events.

To use the abstract PHY, turn on **Simulation Options > Abstract PHY for fast simulation** on the **Diagnostic** tab. When you turn on the abstract PHY, the EMIF IP is configured as shown below. The PHY RTL and external memory model are disconnected from the data path and in their place is the abstract PHY containing an internal memory array. The abstract PHY is designed with no high-speed clocks, resulting in the removal of all high-speed clock simulator events.

**Figure 8-2: Abstract PHY**

**Note:** You cannot observe the external memory device signals when you are using the abstract PHY.

If the memory controller is normally part of the EMIF IP, it will continue to be instantiated and used when simulating with the abstract PHY. Because the memory controller regulates the throughput characteristics of data to the external memory interface, these throughput characteristics are maintained when simulating with the abstract PHY. It is important to understand that the abstract PHY is not a cycle-accurate mode of the EMIF IP, and therefore you should not expect to see cycle-accurate behavior.

The HDL forces are created by the Quartus Prime software at IP generation, and therefore you can run abstract PHY simulations immediately upon generation of the EMIF IP.

## Simulation Scripts

The Quartus Prime software generates three simulation scripts during project generation for four different third party simulation tools—Cadence, Synopsys, Aldec, and Mentor.

The simulation scripts reduce the number of files that you need to compile separately before simulating a design. These scripts are located in four separate folders under the `<project_directory>\<variation_name>_sim` directory, each named after the names of the simulation tools. The example designs also provide equivalent scripts after you run the `.tcl` script from the project located in the `\<variation_name>_example_design\sim` directory.

## Functional Simulation with Verilog HDL

Altera provides simulation scripts for you to run the example design. The simulation scripts are for Synopsys, Cadence, Aldec, and Mentor simulators.

The simulation scripts are located in the following main folder locations:

Simulation scripts in the simulation folders are located as follows:

- `<variation_name>_example_design\sim\mentor\msim_setup.tcl`
- `<variation_name>_example_design\sim\synopsys\vcs\vcs_setup.sh`
- `<variation_name>_example_design\sim\synopsys\vcsmx\vcsmx_setup.sh`
- `<variation_name>_example_design\sim\aldec\rivierapro_setup.tcl`
- `<variation_name>_example_design\sim\cadence\ncsim_setup.sh`

Simulation scripts in the <>**\_sim\_folder** are located as follows:

- <*variation\_name*>\_sim\mentor\msim\_setup.tcl
- <*variation\_name*>\_sim\cadence\ncsim\_setup.sh
- <*variation\_name*>\_sim\synopsys\vcs\vcs\_setup.sh
- <*variation\_name*>\_sim\synopsys\vcs\vcsmx\vcsmx\_setup.sh
- <*variation\_name*>\_sim\aldec\rivierapro\_setup.tcl

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Mentor Graphics ModelSim® and QuestaSim Support* chapter in volume 3 of the Quartus Prime Handbook.

#### Related Information

[Mentor Graphics ModelSim and QuestaSim Support](#)

## Functional Simulation with VHDL

The Arria 10 EMIF VHDL fileset is provided for customers that wish to generate the top-level RTL instance of their Arria 10 EMIF cores in VHDL.

Prior to Quartus Prime version 15.1, the VHDL fileset was comprised entirely of VHDL files. Beginning with Quartus Prime version 15.1, only the top-level IP instance file is guaranteed to be written in VHDL; submodules can still be deployed as Verilog/SystemVerilog (encrypted or plaintext) files, or VHDL files. Note that the ModelSIM Altera Edition is no longer restricted to a single HDL language as of Quartus 15.1; however, some files may still be encrypted in order to be excluded from the maximum unencrypted module limit of this tool.

Because the VHDL fileset consists of both VHDL and Verilog files, you must follow certain mixed-language simulation guidelines. The general guideline for mixed-language simulation is that you must always link the Verilog files (whether encrypted or not) against the Verilog version of the Altera libraries, and the VHDL files (whether SimGen-generated or pure VHDL) against the VHDL libraries.

Altera provides simulation scripts for you to run the example design. The simulation scripts are for Synopsys, Cadence, Aldec, and Mentor simulators. These simulation scripts are located in the following main folder locations:

Simulation scripts in the simulation folders are located as follows:

- <*variation\_name*>\_example\_design\sim\mentor\msim\_setup.tcl
- <*variation\_name*>\_example\_design\sim\synopsys\vcsmx\vcsmx\_setup.sh
- <*variation\_name*>\_example\_design\sim\synopsys\vcs\vcs\_setup.sh
- <*variation\_name*>\_example\_design\sim\cadence\ncsim\_setup.sh
- <*variation\_name*>\_example\_design\sim\aldec\rivierapro\_setup.tcl

Simulation scripts in the <>**\_sim\_folder** are located as follows:

- <*variation\_name*>\_sim\mentor\msim\_setup.tcl
- <*variation\_name*>\_sim\cadence\ncsim\_setup.sh
- <*variation\_name*>\_sim\synopsys\vcsmx\vcsmx\_setup.sh
- <*variation\_name*>\_sim\aldec\rivierapro\_setup.tcl

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Mentor Graphics ModelSim® and QuestaSim Support* chapter in volume 3 of the Quartus Prime Handbook.

#### Related Information

[Mentor Graphics ModelSim and QuestaSim Support](#)

## Simulating the Example Design

This topic describes how to simulate the example design in Cadence, Synopsys, Mentor, and Aldec simulators.

To simulate the example design in the Quartus Prime software using the Cadence simulator, follow these steps:

1. At the Linux shell command prompt, change directory to **<name>\_example\_design\sim\cadence**
2. Run the simulation by typing the following command at the command prompt:

```
sh ncsim_setup.sh
```

To simulate the example design in the Quartus Prime software using the Synopsys simulator, follow these steps:

1. At the Linux shell command prompt, change directory to **<name>\_example\_design\sim\synopsys\vcsmx**
2. Run the simulation by typing the following command at the command prompt:

```
sh vcsmx_setup.sh
```

To simulate the example design in the Quartus Prime software using the Mentor simulator, follow these steps:

1. At the Linux or Windows shell command prompt, change directory to **<name>\_example\_design\sim\mentor**
2. Execute the **msim\_setup.tcl** script that automatically compiles and runs the simulation by typing the following command at the Linux or Windows command prompt:

```
vsim -do msim_setup.tcl
```

or

Type the following command at the ModelSim command prompt:

```
do msim_setup.tcl
```

**Note:** Altera does not provide the **run.do** file for the example design with the Arria 10 EMIF interface.

To simulate the example design in the Quartus Prime software using the Aldec simulator, follow these steps:

1. At the Linux or Windows shell command prompt, change directory to **<name>\_example\_design\sim\aldec**
2. Execute the **rivierapro\_setup.tcl** script that automatically compiles and runs the simulation by typing the following command at the Linux or Windows command prompt:  
`vsim -do rivierapro.tcl`
3. To compile and elaborate the design after the script loads, type `ld_debug`.
4. Type `run -all` to run the simulation.

For more information about simulation, refer to the *Simulating Altera Designs* chapter in volume 3 of the Quartus Prime Handbook.

If your Quartus Prime project appears to be configured correctly but the example testbench still fails, check the known issues on the Knowledge Database page of the Altera website before filing a service request.

#### Related Information

- [Simulating Altera Designs](#)
- [Knowledge Database](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"> <li>• Added paragraph about incompatibility with the Qsys Testbench System generation feature to <i>Simulating Memory IP</i>.</li> <li>• Updated <i>Simulation Options</i> and <i>Abstract PHY Simulation</i> topics for changes to Abstract PHY.</li> <li>• Added note to tables 10-1 and 10-2.</li> <li>• Updated <i>Functional Simulation with VHDL</i> topic in both <i>Simulation Walkthrough with UniPHY IP</i> and <i>Simulation Walkthrough with Arria 10 EMIF IP</i> sections.</li> </ul>
November 2015	2015.11.02	<ul style="list-style-type: none"> <li>• Added <i>Abstract PHY Simulation</i> topic.</li> <li>• Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> </ul>
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Removed references to MegaWizard Plug-In Manager.
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>• Removed references to ALTMEMPHY.</li> <li>• Removed references to HardCopy.</li> <li>• Removed references to SOPC Builder.</li> <li>• Added <i>Simulation Walkthrough with Arria 10 EMIF IP</i>.</li> <li>• Clarified explanation of full calibration mode with abstract models in <i>Abstract PHY</i> section.</li> </ul>
November 2012	6.0	Changed chapter number from 9 to 10.
June 2012	5.0	<ul style="list-style-type: none"> <li>• Added path to simulation scripts for Riviera-PRO to <i>Functional Simulations</i> section.</li> <li>• Added simulation procedure for Riviera-PRO to <i>Simulating the Example Design</i> section.</li> <li>• Updated the <i>Abstract PHY</i> section.</li> <li>• Updated the <i>Post-fit Functional Simulation</i> procedure.</li> <li>• Added Feedback icon.</li> </ul>

Date	Version	Changes
November 2011	4.0	<ul style="list-style-type: none"><li>Added the <i>PHY-Only Simulation</i> section.</li><li>Added the <i>Post-fit Functional Simulation</i> section.</li><li>Updated the <i>Simulation Walkthrough with UniPHY IP</i> section.</li></ul>
June 2011	3.0	<ul style="list-style-type: none"><li>Added an overview about memory simulation.</li><li>Added the <i>Simulation Walkthrough with UniPHY IP</i> section.</li></ul>
December 2010	2.1	Updated fr 10.1 release.
July 2010	2.0	Updated for 10.0 release.
January 2010	1.1	Corrected typos.
November 2009	1.0	Initial release.

# Analyzing Timing of Memory IP

9

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The external memory physical layer (PHY) interface offers a combination of source-synchronous and self-calibrating circuits to maximize system timing margins. The physical layer interface is a plug-and-play solution that the Quartus® Prime TimeQuest Timing Analyzer timing constrains and analyzes.

The Altera IP and the numerous device features offered by Arria® II, Arria V, Arria 10, Cyclone® V, Stratix® III, Stratix IV, and Stratix V FPGAs, greatly simplify the implementation of an external memory interface.

This chapter details the various timing paths that determine overall external memory interface performance, and describes the timing constraints and assumptions that the PHY IP uses to analyze these paths.

This chapter focuses on timing constraints for external memory interfaces based on the UniPHY IP. For information about timing constraints and analysis of external memory interfaces and other source-synchronous interfaces based on the ALTDQ\_DQS and ALTDQ\_DQS2 IP cores, refer to AN 433: *Constraining and Analyzing Source-Synchronous Interfaces* and the *Quartus Prime TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus Prime Handbook*.

External memory interface timing analysis is supported only by the TimeQuest Timing Analyzer, for the following reasons:

- The wizard-generated timing constraint scripts support only the TimeQuest analyzer.
- The Classic Timing Analyzer does not offer analysis of source-synchronous outputs. For example, write data, address, and command outputs.
- The Classic Timing Analyzer does not support detailed rise and fall delay analysis.

The performance of an FPGA interface to an external memory device is dependent on the following items:

- Read datapath timing
- Write datapath timing
- Address and command path timing
- Clock to strobe timing ( $t_{DQSS}$  in DDR and DDR2 SDRAM, and  $t_{KHK#H}$  in QDR II and QDRII+ SRAM)
- Read resynchronization path timing (applicable for DDR, DDR2, and DDR3 SDRAM in Arria II, Arria 10, Stratix III, Stratix IV, and Stratix V devices)
- Write leveling path timing (applicable for DDR2 and DDR3 SDRAM with UniPHY, and DDR3 and DDR4 SDRAM with Arria 10 EMIF IP.)
- PHY timing paths between I/O element and core registers
- PHY and controller internal timing paths (core  $f_{MAX}$  and reset recovery/removal)

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

- I/O toggle rate
- Output clock specifications
- Bus turnaround timing (applicable for RLDRAM II and DDR2 and DDR3 SDRAM with UniPHY)

**Note:** External memory interface performance depends on various timing components, and overall system level performance is limited by performance of the slowest link (that is, the path with the smallest timing margins).

#### Related Information

- [AN 433: Constraining and Analyzing Source-Synchronous Interfaces](#)
- [Quartus PrimeTimeQuest Timing Analyzer](#)

## Memory Interface Timing Components

There are several categories of memory interface timing components, including source-synchronous timing paths, calibrated timing paths, internal FPGA timing paths, and other FPGA timing parameters.

Understanding the nature of timing paths enables you to use an appropriate timing analysis methodology and constraints. The following section examines these aspects of memory interface timing paths.

### Source-Synchronous Paths

Source-synchronous timing paths are those where clock and data signals pass from the transmitting device to the receiving device.

An example of a source-synchronous timing path is the FPGA-to-memory write datapath. The FPGA device transmits DQ output data signals to the memory along with a center-aligned DQS output strobe signal. The memory device uses the DQS signal to clock the data on the DQ pins into its internal registers.

**Note:** For brevity, the following topics refer to data signals and clock strobe signals as DQ signals and DQS signals, respectively. While the terminology is formally correct only for DDR-type interfaces and does not match QDR II, QDR II+ and RLDRAM II pin names, the behavior is similar enough that most timing properties and concepts apply to both. The clock that captures address and command signals is always referred to as CK/CK# too.

### Calibrated Paths

Calibrated timing paths are those where the clock used to capture data is dynamically positioned within the data valid window (DVW) to maximize timing margin.

For UniPHY-based controllers and Arria 10 EMIF controllers, the sequencer block analyzes all path delays between the read capture registers and the read FIFO buffer to set up the FIFO write clock phase for optimal timing margin. The read postamble calibration process is implemented in a similar manner to the read resynchronization calibration. In addition, the sequencer block calibrates a read data valid signal to the delay between a controller issuing a read command and read data returning to controller.

In DDR2, DDR3, and RLDRAM II with UniPHY, and in Arria 10 EMIF, the IP calibrates the write-leveling chains and programmable output delay chain to align the DQS edge with the CK edge at memory to meet the  $t_{DQSS}$ ,  $t_{DSS}$ , and  $t_{DSH}$  specifications.

Both UniPHY IP and Arria 10 EMIF IP enable the dynamic deskew calibration with the Nios II sequencer for read and write paths. Dynamic deskew process uses the programmable delay chains that exist within the read and write data paths to adjust the delay of each DQ and DQS pin to remove the skew between

different DQ signals and to centre-align the DQS strobe in the DVW of the DQ signals. This process occurs at power up for the read and the write paths.

## Internal FPGA Timing Paths

Other timing paths that have an impact on memory interface timing include FPGA internal  $f_{MAX}$  paths for PHY and controller logic.

This timing analysis is common to all FPGA designs. With appropriate timing constraints on the design (such as clock settings), the TimeQuest Timing Analyzer reports the corresponding timing margins.

For more information about the TimeQuest Timing Analyzer, refer to the *Quartus Prime TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus Prime Handbook*.

### Related Information

[Quartus Prime TimeQuest Timing Analyzer](#)

## Other FPGA Timing Parameters

Some FPGA data sheet parameters, such as I/O toggle rate and output clock specifications, can limit memory interface performance.

I/O toggle rates vary based on speed grade, loading, and I/O bank location— top/bottom versus left/right. This toggle rate is also a function of the termination used (OCT or external termination) and other settings such as drive strength and slew rate.

**Note:** Ensure you check the I/O performance in the overall system performance calculation. Altera recommends that you perform signal integrity analysis for the specified drive strength and output pin load combination.

For information about signal integrity, refer to the board design guidelines chapters and *AN 476: Impact of I/O Settings on Signal Integrity in Stratix III Devices*.

Output clock specifications include clock period jitter, half-period jitter, cycle-to-cycle jitter, and skew between FPGA clock outputs. You can obtain these specifications from the FPGA data sheet and must meet memory device requirements. You can use these specifications to determine the overall data valid window for signals transmitted between the memory and FPGA device.

### Related Information

[AN476: Impact of I/O Settings on Signal Integrity in Stratix III Devices](#)

## FPGA Timing Paths

The following topics describe the FPGA timing paths, the timing constraints examples, and the timing assumptions that the constraint scripts use.

In Arria II, Arria V, Arria V GZ, Arria 10, Cyclone V, Stratix III, Stratix IV, and Stratix V devices, the interface margin is reported based on a combination of the TimeQuest Timing Analyzer and further steps to account for calibration that occurs at runtime. First the TimeQuest analyzer returns the base setup and hold slacks, and then further processing adjusts the slacks to account for effects which cannot be modeled in TimeQuest.

## Arria II Device PHY Timing Paths

The following table lists all Arria II devices external memory interface timing paths.

**Table 9-1: Arria II Devices External Memory Interface Timing Paths<sup>(1)</sup>**

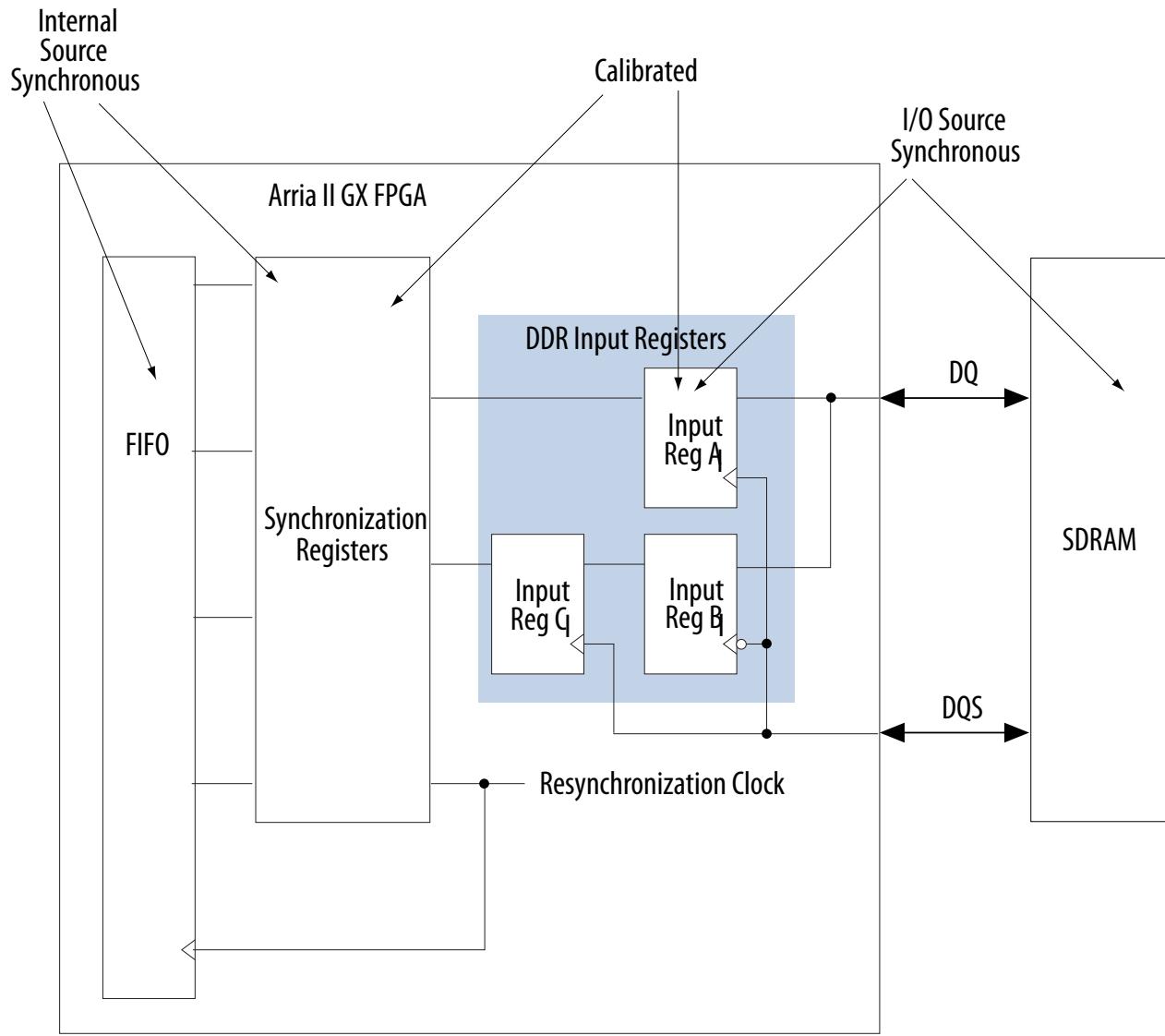
Timing Path	Circuit Category	Source	Destination
Read Data <sup>(2) (6)</sup>	Source-Synchronous	Memory DQ, DQS Pins	DQ Capture Registers in IOE
Write Data <sup>(2) (6)</sup>	Source-Synchronous	FPGA DQ, DQS Pins	Memory DQ, DM, and DQS Pins
Address and command <sup>(2)</sup>	Source-Synchronous	FPGA CK/CK# and Addr/Cmd Pins	Memory Input Pins
Clock-to-Strobe <sup>(2)</sup>	Source-Synchronous	FPGA CK/CK# and DQS Output Pins	Memory Input Pins
Read Resynchronization <sup>(2)</sup>	Calibrated	IOE Capture Registers	IOE Resynchronization Registers
Read Resynchronization <sup>(2) (5)</sup>	Calibrated	IOE Capture Registers	Read FIFO in FPGA Core
PHY IOE-Core Paths <sup>(2)</sup>	Source-Synchronous	IOE Resynchronization Registers	FIFO in FPGA Core
PHY and Controller Internal Paths <sup>(2)</sup>	Internal Clock $f_{MAX}$	Core Registers	Core Registers
I/O Toggle Rate <sup>(3)</sup>	I/O	FPGA Output Pin	Memory Input Pins
Output Clock Specifications (Jitter, DCD) <sup>(4)</sup>	I/O	FPGA Output Pin	Memory Input Pins

Notes to Table:

1. Timing paths applicable for an interface between Arria II devices and SDRAM component.
2. Timing margins for this path are reported by the TimeQuest Timing Analyzer Report DDR function.
3. Altera recommends that you perform signal integrity simulations to verify I/O toggle rate.
4. For output clock specifications, refer to the *Arria II Device Data Sheet* chapter of the *Arria II Handbook*.
5. Only for UniPHY IP.
6. Arria II GX devices use source-synchronous and calibrated path.

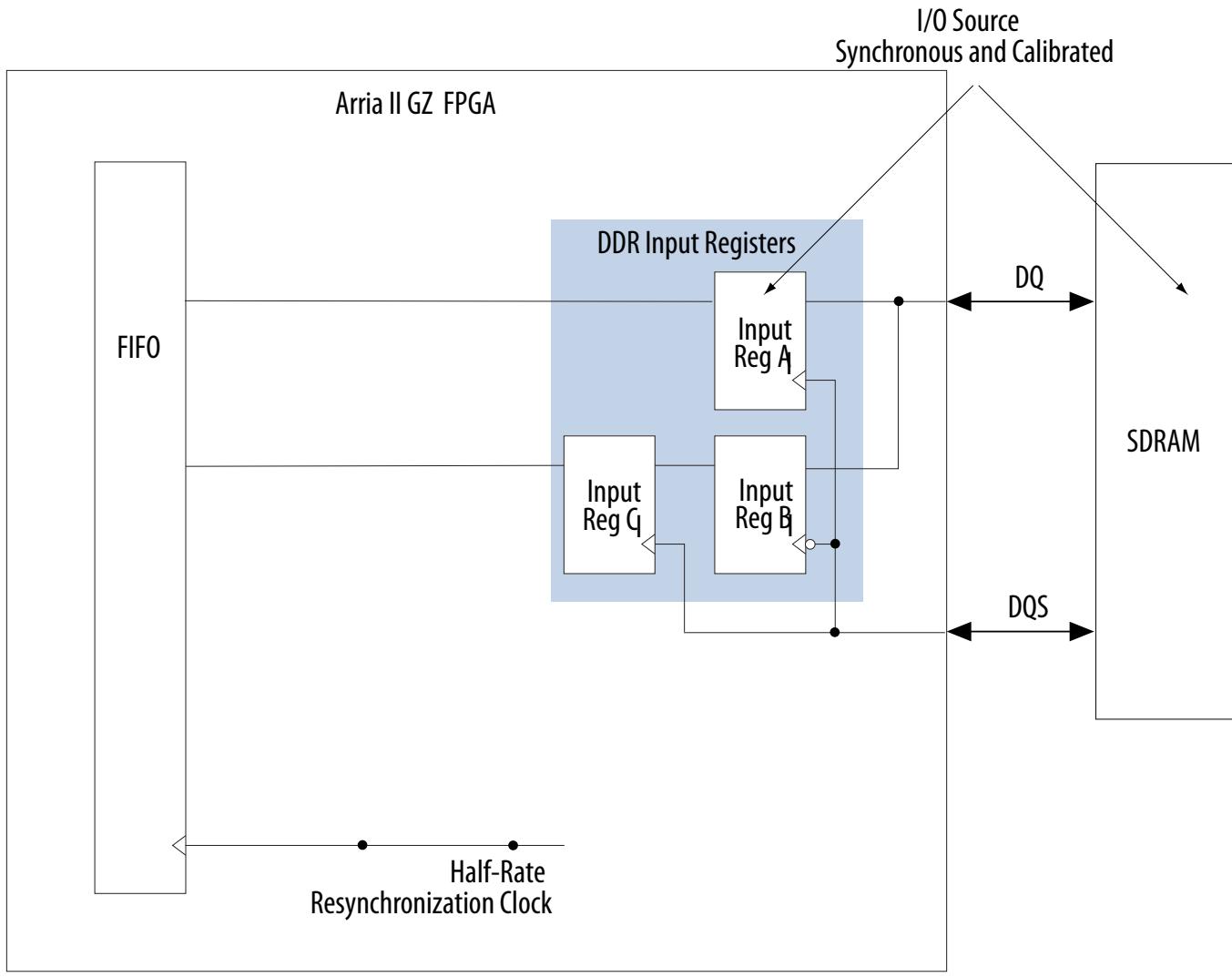
The following figure shows the Arria II GX devices input datapath registers and circuit types.

**Note:** UniPHY IP interfaces bypass the synchronization registers.

**Figure 9-1: Arria II GX Devices Input Data Path Registers and Circuit Types in SDRAM Interface**

The following figure shows the Arria II GZ devices input datapath registers and circuit types.

Figure 9-2: Arria II GZ Devices Input Data Path Registers and Circuit Types in SDRAM Interface

**Related Information**[Device Datasheet for Arria II Devices](#)

## Stratix III and Stratix IV PHY Timing Paths

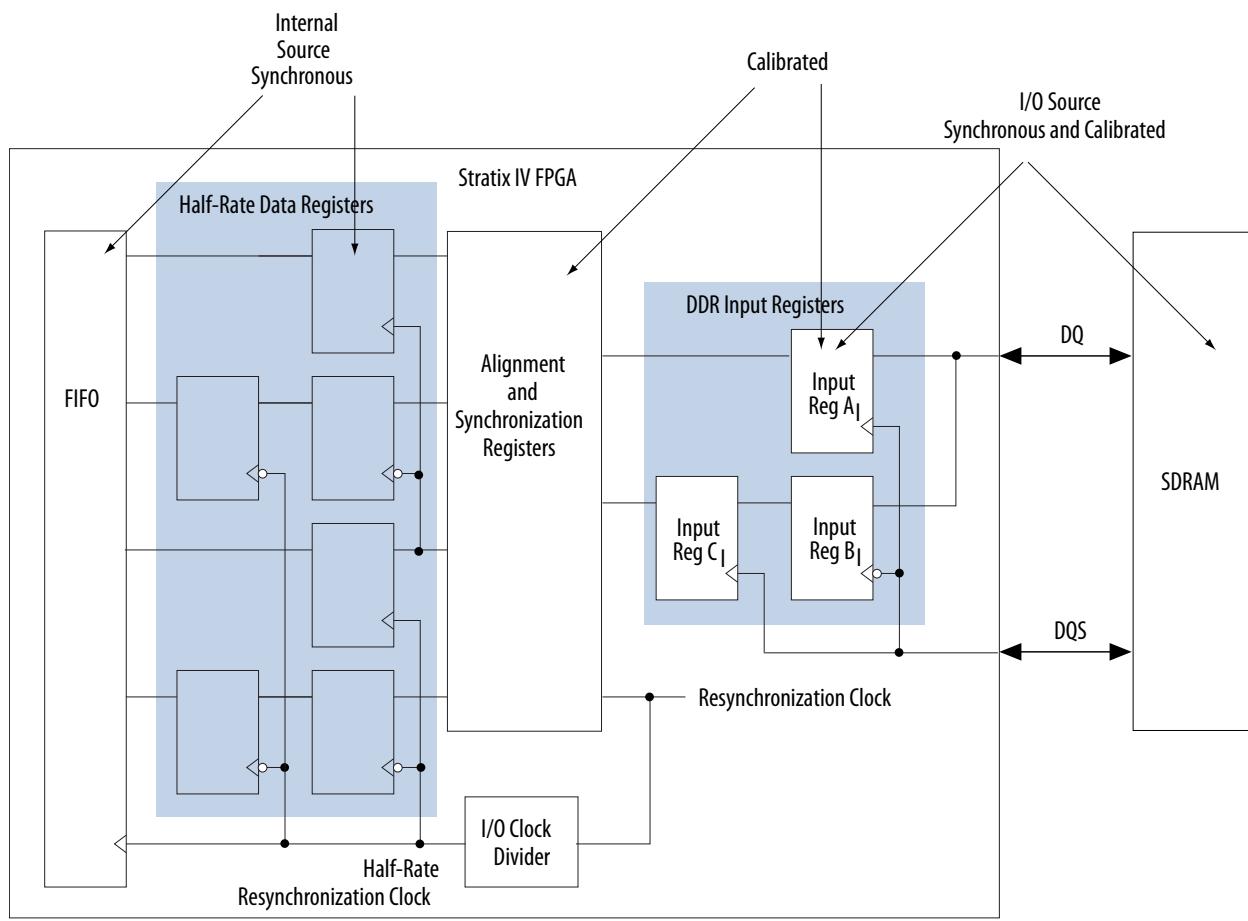
A close look at all the register transfers occurring in the Stratix III and Stratix IV input datapath reveals many source-synchronous and calibrated circuits.

**Note:** The information in the following figure and table is based on Stratix IV devices, but is also applicable to Stratix III devices.

The following figure shows a block diagram of this input path with some of these paths identified for Stratix IV devices. The output datapath contains a similar set of circuits.

**Note:** UniPHY IP interfaces bypass the alignment and synchronization registers.

Figure 9-3: Stratix IV Input Path Registers and Circuit Types in SDRAM Interface



The following table lists the timing paths applicable for an interface between Stratix IV devices and half-rate SDRAM components.

**Note:** The timing paths are also applicable to Stratix III devices, but Stratix III devices use only source-synchronous path for read and write data paths.

Table 9-2: Stratix IV External Memory Interface Timing Paths (Part 1 of 2)

Timing Path	Circuit Category	Source	Destination
Read Data <sup>(1)</sup>	Source-Synchronous and Calibrated	Memory DQ, DQS Pins	DQ Capture Registers in IOE
Write Data <sup>(1)</sup>	Source-Synchronous and Calibrated	FPGA DQ, DQS Pins	Memory DQ, DM, and DQS Pins
Address and command <sup>(1)</sup>	Source-Synchronous	FPGA CK/CK# and Addr/ Cmd Pins	Memory Input Pins

Timing Path	Circuit Category	Source	Destination
Clock-to-Strobe <sup>(1)</sup>	Source-Synchronous	FPGA CK/CK# and DQS Output Pins	Memory Input Pins
Read Resynchronization <sup>(1)</sup>	Calibrated	IOE Capture Registers	IOE Alignment and Resynchronization Registers
Read Resynchronization <sup>(1) (4)</sup>	Calibrated	IOE Capture Registers	Read FIFO in FPGA Core
PHY IOE-Core Paths <sup>(1)</sup>	Source-Synchronous	IOE Half Data Rate Registers and Half-Rate Resynchronization Clock	FIFO in FPGA Core
PHY & Controller Internal Paths <sup>(1)</sup>	Internal Clock $f_{MAX}$	Core registers	Core registers
I/O Toggle Rate <sup>(2)</sup>	I/O – Data sheet	FPGA Output Pin	Memory Input Pins
Output Clock Specifications (Jitter, DCD) <sup>(3)</sup>	I/O – Data sheet	FPGA Output Pin	Memory Input Pins

Notes to Table:

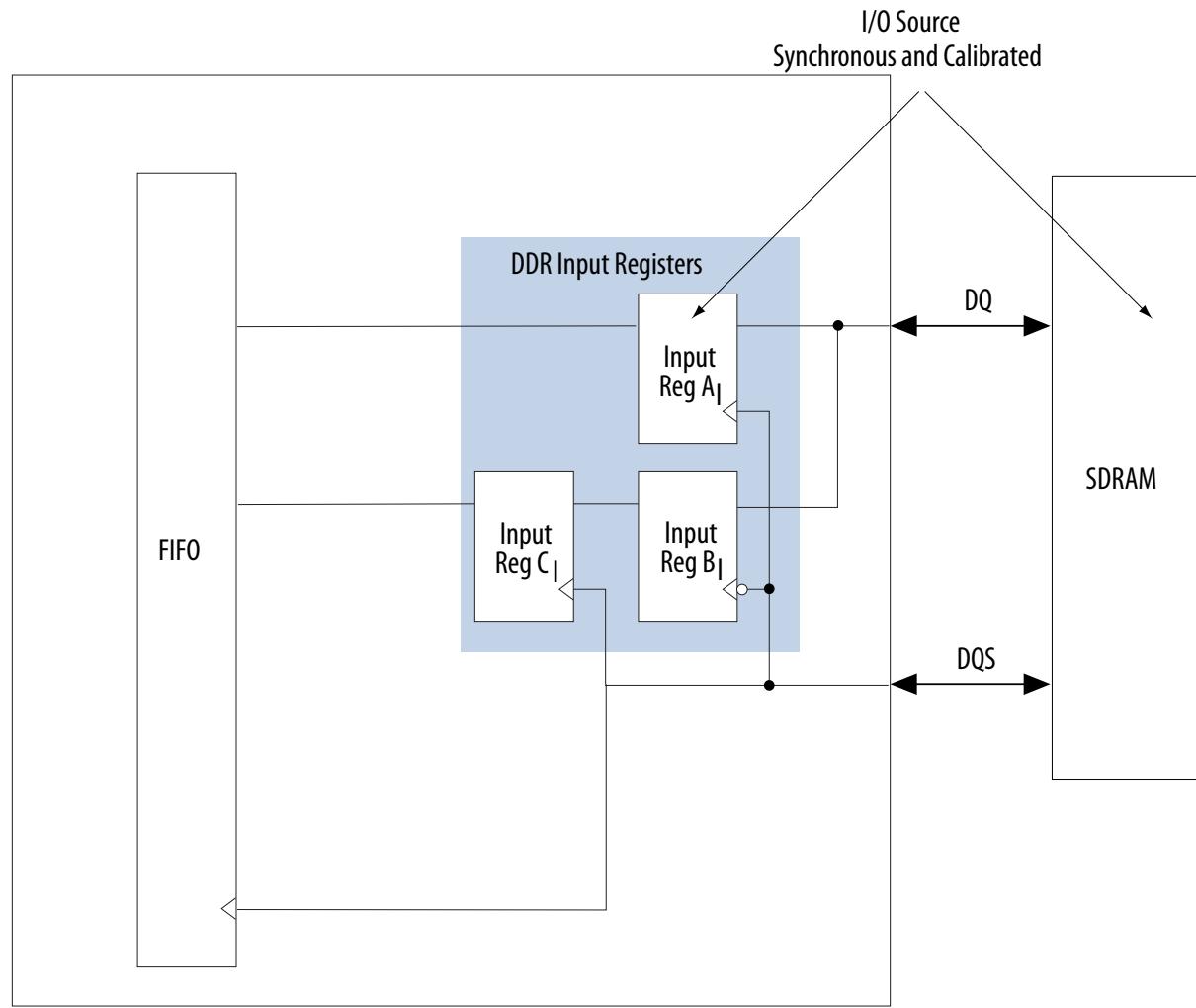
1. Timing margins for this path are reported by the TimeQuest Timing Analyzer Report DDR function.
2. Altera recommends that you perform signal integrity simulations to verify I/O toggle rate.
3. For output clock specifications, refer to the *DC and Switching Characteristics* chapter of the *Stratix IV Device Handbook*.
4. Only for UniPHY IP.

#### Related Information

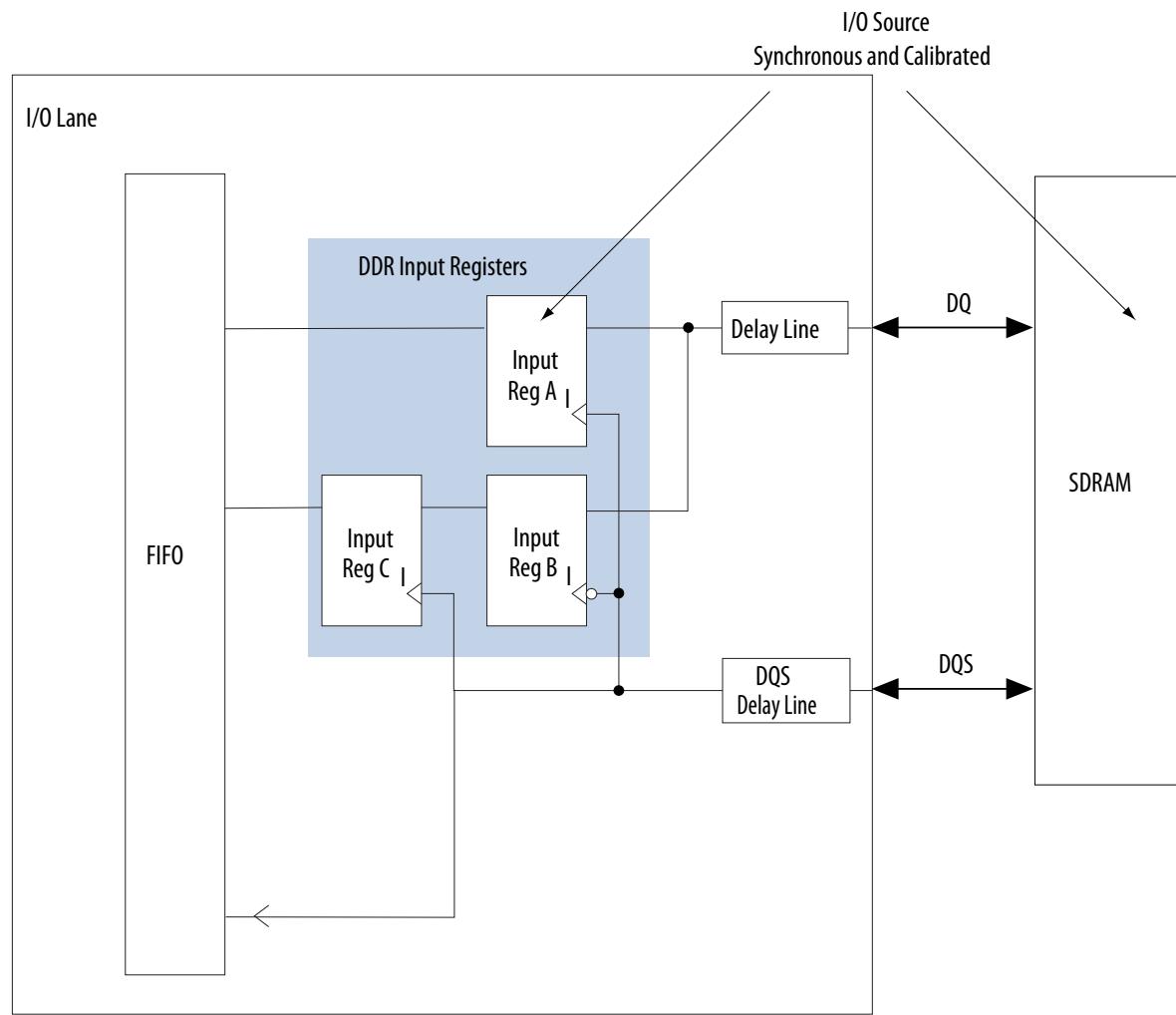
- [DC and Switching Characteristics for Stratix III Devices](#)
- [DC and Switching Characteristics for Stratix IV Devices](#)

## Arria V, Arria V GZ, Arria 10, Cyclone V, and Stratix V Timing paths

The following figures show block diagrams of the input data paths for Arria V, Arria V GZ, Cyclone V, and Stratix V devices, and for Arria 10 devices.

**Figure 9-4: Arria V, Arria V GZ, Cyclone V, and Stratix V Input Data Path**

The following figure shows a block diagram of the Arria 10 input data path.

**Figure 9-5: Arria 10 Input Data Path**

The following table lists all Arria V, Arria V GZ, Arria 10, Cyclone V, and Stratix V devices external memory interface timing paths.

**Table 9-3: Arria V, Arria V GZ, Arria 10, Cyclone V, and Stratix V External Memory Interface Timing Paths<sup>(1)</sup>**

Timing Path	Circuit Category	Source	Destination
Read Data <sup>(2)</sup>	Source-Synchronous and Calibrated	Memory DQ, DQS Pins	DQ Capture Registers in IOE
Write Data <sup>(2)</sup>	Source-Synchronous and Calibrated	FPGA DQ, DM, DQS Pins	Memory DQ, DM, and DQS Pins
Address and command <sup>(2)</sup>	Source-Synchronous	FPGA CK/CK# and Addr/Cmd Pins	Memory Input Pins

Timing Path	Circuit Category	Source	Destination
Clock-to-Strobe <sup>(2)</sup>	Source-Synchronous	FPGA CK/CK# and DQS Output Pins	Memory Input Pins
Read Resynchronization <sup>(2)</sup>	Source-Synchronous	IOE Capture Registers	Read FIFO in IOE
PHY & Controller Internal Paths <sup>(2)</sup>	Internal Clock fMAX	Core Registers	Core Registers
i/O Toggle Rate <sup>(3)</sup>	I/O – Data sheet	FPGA Output Pin	Memory Input Pins
Output Clock Specifications (Jitter, DCD) <sup>(4)</sup>	I/O – Data sheet	FPGA Output Pin	Memory Input Pins

Notes to Table:

1. This table lists the timing paths applicable for an interface between Arria V, Arria V GZ, Cyclone V, and Stratix V devices and half-rate SDRAM components.
2. Timing margins for this path are reported by the TimeQuest Timing Analyzer Report DDR function.
3. Altera recommends that you perform signal integrity simulations to verify I/O toggle rate.
4. For output clock specifications, refer to the *DC and Switching Characteristics* chapter of the respective *Device Handbook*.

The following table lists the Arria 10 external memory interface timing paths.

Timing Path	Circuit Category	Source	Destination
Read Data <sup>(1)</sup>	Source-Synchronous and Calibrated	Memory DQ, DQS Pins	DQ Capture Registers in IOE
Write Data <sup>(1)</sup>	Source-Synchronous and Calibrated	FPGA DQ, DM, DQS Pins	Memory DQ, DM, and DQS Pins
Address and Command <sup>(1)</sup>	Source-Synchronous	FPGA CK/CK# and Address/Command Pins	Memory Input Pins
Clock-to-Strobe <sup>(1)</sup>	Source-Synchronous	FPGA CK/CK# and DQS Output Pins	Memory Input Pins
PHY & Controller Internal Paths	Internal Clock fmax	Core Registers	Core Registers
I/O Toggle Rate <sup>(2)</sup>	I/O Data sheet	FPGA Output Pin	Memory Input Pins
Output Clock Specifications (Jitter, DCD) <sup>(3)</sup>	I/O Data sheet	FPGA Output Pin	Memory Input Pins

Timing Path	Circuit Category	Source	Destination
Notes to Table:			
<ol style="list-style-type: none"> <li>1. The Report DDR function in the TimeQuest Timing Analyzer reports the timing margins for this path.</li> <li>2. You should perform signal integrity simulations for verify I/O toggle rate.</li> <li>3. For output clock verifications, refer to the <i>DC and Switching Characteristics</i> chapter of the <i>Arria 10 Device Handbook</i>.</li> </ol>			

## Timing Constraint and Report Files for UniPHY IP

To ensure a successful external memory interface operation, the UniPHY IP generates two sets of files for timing constraints but in different folders and with slightly different filenames.

One set of files are used for the synthesis project, which is available under the `<variation_name>` folder located in the main project folder while the other set of files are the example designs, located in the `<variation_name>example design\example_project` folder.

The project folders contain the following files for timing constraints and reporting scripts:

- `<variation_name>.sdc`
- `<variation_name>_timing.tcl`
- `<variation_name>_report_timing.tcl`
- `<variation_name>_report_timing_core.tcl`
- `<variation_name>_pin_map.tcl`
- `<variation_name>_parameters.tcl`

### `<variation_name>.sdc`

The `<variation_name>.sdc` is listed in the wizard-generated Quartus Prime IP File (`.qip`). Including this file in the project allows the Quartus Prime Synthesis and Fitter to use the timing driven compilation to optimize the timing margins.

To analyze the timing margins for all UniPHY timing paths, execute the Report DDR function in the TimeQuest Timing Analyzer.

The UniPHY IP uses the `.sdc` to constrain internal FPGA timing paths, address and command paths, and clock-to-strobe timing paths, and more specifically:

- Creating clocks on PLL inputs
- Creating generated clocks
- Calling `derive_clock_uncertainty`
- Cutting timing paths for specific reset paths
- Setting input and output delays on DQ inputs and outputs
- Setting output delays on address and command outputs (versus CK/CK# outputs)

### `<variation_name>_timing.tcl`

This script includes the memory, FPGA, and board timing parameters for your variation. It is included within `<variation_name>_report_timing.tcl` and `<variation_name>.sdc`.

### **<variation\_name>\_report\_timing.tcl**

This script reports the timing slack for your variation. It runs automatically during compilation (during static timing analysis). You can also run this script with the Report DDR task in the TimeQuest Timing Analyzer. This script is run for every instance of the same variation.

### **<variation\_name>\_report\_timing\_core.tcl**

This script contains high-level procedures that the **<variation\_name>\_report\_timing.tcl** script uses to compute the timing slack for your variation. This script runs automatically during compilation.

### **<variation\_name>\_pin\_map.tcl**

This script is a library of functions and procedures that the **<variation\_name>\_report\_timing.tcl** and **<variation\_name>.sdc** scripts use. The **<variation\_name>\_pin\_assignments.tcl** script, which is not relevant to timing constraints, also uses this library.

### **<variation\_name>\_parameters.tcl**

This script defines some of the parameters that describe the geometry of the core and the PLL configuration. Do not change this file, except when you modify the PLL through the parameter editor. In this case, the changes to the PLL parameters do not automatically propagate to this file and you must manually apply those changes in this file.

## **Timing Constraint and Report Files for Arria 10 EMIF IP**

To ensure a successful external memory interface operation, the UniPHY IP generates two sets of files for timing constraints but in different folders and with slightly different filenames.

One set of files are used for synthesis project, which is available under the **<variation\_name>** folder located in the main project folder while the other set of files are the example designs, located in the **<variation\_name>\_example\_design\qii** folder.

The project folders contain the following files for timing constraints and reporting scripts:

- **<variation\_name>.sdc**
- **<variation\_name>\_ip\_parameters.tcl**
- **<variation\_name>\_parameters.tcl**
- **<variation\_name>\_pin\_map.tcl**
- **<variation\_name>\_report\_timing.tcl**

### **<variation\_name>.sdc**

The **<variation\_name>.sdc** file is listed in the Quartus Prime IP File (**.qip**), which you generate by running **make\_qii\_design.tcl**. The **<variation\_name>.sdc** file allows the Quartus Prime fitter to optimize timing margins with timing-driven compilation.

To analyze timing margins for all Arria 10 external memory interface IP timing paths, run the **Report DDR** function in the TimeQuest Timing Analyzer.

The Arria 10 EMIF IP uses the **.sdc** file for the following operations:

- Creating clocks on PLL inputs
- Creating generated clocks
- Calling **derive\_clock\_uncertainty**
- Creating a false path from a user clock to a hard memory controller clock, and vice versa
- Setting output delays on address and command outputs (versus CK/CK# outputs)

#### **<variation\_name>.ip\_parameters.tcl**

The **<variation\_name>.ip\_parameters.tcl** file is a script that lists the Arria 10 EMIF IP memory parameters and board parameters defined in the MegaWizard, which are used in the **.sdc** file and timing report scripts.

#### **<variation\_name>.parameters.tcl**

The **<variation\_name>.parameters.tcl** file is a script that lists the Arria 10 EMIF IP device and speed grade dependent values, which are used in the **.sdc** file and report timing scripts:

- Jitter
- Simultaneous switching noise
- Duty cycle distortion
- Calibration uncertainties

#### **<variation\_name>.pin\_map.tcl**

The **<variation\_name>.pin\_map.tcl** file is a library of functions and procedures that the **<variation\_name>.report\_timing.tcl** and **<variation\_name>.sdc** scripts use.

#### **<variation\_name>.report\_timing.tcl**

The **<variation\_name>.report\_timing.tcl** file is a script that contains timing analysis flow and reports the timing slack for your variation. This script runs automatically during calibration (during static timing analysis) by sourcing the following files:

- **<variation\_name>.ip\_parameters.tcl**
- **<variation\_name>.parameters.tcl**
- **<variation\_name>.pin\_map.tcl**
- **<variation\_name>.report\_timing\_core.tcl**

You can also run **<variation\_name>.report\_timing.tcl** with the **Report DDR** function in the TimeQuest Timing Analyzer. This script runs for every instance of the same variation.

#### **<variation\_name>.report\_timing\_core.tcl**

The **<variation\_name>.report\_timing\_core.tcl** file is a script that **<variation\_name>.report\_timing.tcl** uses to calculate the timing slack for your variation. **<variation\_name>.report\_timing\_core.tcl** runs automatically during compilation.

#### **<variation\_name>.report\_io\_timing.tcl**

The **<variation\_name>.report\_io\_timing.tcl** file is a script that contains an early I/O estimation for your external memory interface design, excluding FPGA core timing analysis. This script allows you to determine early I/O margins without having to compile your design.

# Timing Analysis Description

The following sections describe the timing analysis using the respective FPGA data sheet specifications and the user-specified memory data sheet parameters.

- Core to core (C2C) transfers have timing constraint created and are timing analyzed by TimeQuest Analyzer. Core timing does not include user logic timing within core or to and from EMIF block. Both UniPHY-based IP and Arria 10 EMIF IP provide the constrained clock to the customer logic.
- Core to periphery (C2P) transfers have timing constraint created and are timing analyzed by TimeQuest Analyzer. This is common for UniPHY and Arria 10 External Memory Interfaces IP. Because of the increased number of C2P/P2C signals in 20nm families compared to previous families, more work is expected to ensure that these special timing arcs are properly modeled, both during TimeQuest and compilation.
- Periphery to core (P2C) transfers have timing constraint created and are timing analyzed by TimeQuest Analyzer. This is common for UniPHY and Arria 10 External Memory Interfaces IP. Because of the increased number of C2P/P2C signals in 20nm families compared to previous families, more work is expected to ensure that these special timing arcs are properly modeled, both during TimeQuest and compilation.
- Periphery to periphery (P2P) transfers are modeled entirely by a minimum pulse width violation on the hard block, and have no internal timing arc. In UniPHY-based IP, P2P transfers are reported as part of Core Timing analysis. For Arria 10 EMIF IP, P2P transfers are modeled only by a minimum pulse width violation on hardened block.

To account for the effects of calibration, the UniPHY IP and Arria 10 EMIF IP include additional scripts that are part of the `<phy_variation_name>_report_timing.tcl` and `<phy_variation_name>_report_timing_core.tcl` files that determine the timing margin after calibration. These scripts use the setup and hold slacks of individual pins to emulate what is occurring during calibration to obtain timing margins that are representative of calibrated PHYs. The effects considered as part of the calibrated timing analysis include improvements in margin because of calibration, and quantization error and calibration uncertainty because of voltage and temperature changes after calibration. The calibration effects do not apply to Stratix III devices.

## Related Information

[Timing Constraint and Report Files for UniPHY IP](#) on page 9-12

## UniPHY IP Timing Analysis

The following topics describe timing analysis for UniPHY-based external memory interface IP.

### Address and Command

Address and command signals are single data rate signals latched by the memory device using the FPGA output clock.

Some of the address and command signals are half-rate data signals, while others, such as the chip select, are full-rate signals. The TimeQuest Timing Analyzer analyzes the address and command timing paths using the `set_output_delay (max and min)` constraints.

### PHY or Core

Timing analysis of the PHY or core path includes the path of soft registers in the device and the register in the I/O element.

However, the analysis does not include the paths through the pin or the calibrated path. The PHY or core analyzes this path by calling the `report_timing` command in `<variation_name>_report_timing.tcl` and `<variation_name>_report_timing_core.tcl`.

## PHY or Core Reset

The PHY or core reset is the internal timing of the asynchronous reset signals to the UniPHY IP.

The PHY or core analyzes this path by calling the `report_timing` command in `<variation_name>_report_timing.tcl` and `<variation_name>_report_timing_core.tcl`.

## Read Capture and Write

Stratix III memory interface designs perform read capture and write timing analysis using the TCCS and SW timing specification.

Read capture and write timing analysis for Arria II, Cyclone IV, Stratix IV, and Stratix V memory interface designs are based on the timing slacks obtained from the TimeQuest Timing Analyzer and all the effects included with the Quartus Prime timing model such as die-to-die and within-die variations, aging, systematic skew, and operating condition variations. Because the PHY IP adjusts the timing slacks to account for the calibration effects, there are two sets of read capture and write timing analysis numbers —**Before Calibration** and **After Calibration**.

### Stratix III

This topic details the timing margins, such as the read data and write data timing paths, which the TimeQuest Timing Analyzer callates for Stratix III designs. Timing paths internal to the FPGA are either guaranteed by design and tested on silicon, or analyzed by the TimeQuest Timing Analyzer using corresponding timing constraints.

For design guidelines about implementing and analyzing your external memory interface using the PHY in Stratix III and Stratix IV devices, refer to the design tutorials on the *List of designs using Altera External Memory IP* page of the Altera Wiki website.

Timing margins for chip-to-chip data transfers can be defined as:

Margin = bit period – transmitter uncertainties – receiver requirements

where:

- Sum of all transmitter uncertainties = transmitter channel-to-channel skew (TCCS).

The timing difference between the fastest and slowest output edges on data signals, including  $t_{CO}$  variation, clock skew, and jitter. The clock is included in the TCCS measurement and serves as the time reference.

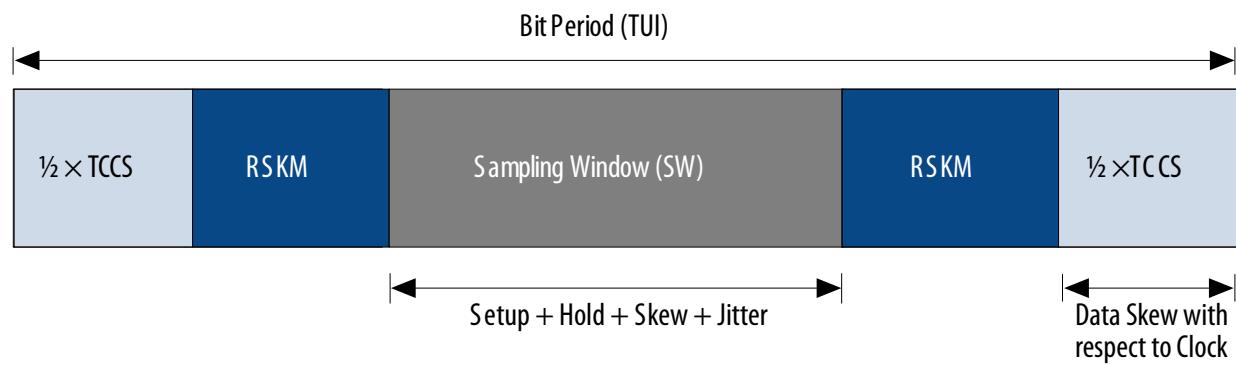
- Sum of all receiver requirements = receiver sampling window (SW) requirement.

The period of time during which the data must be valid to capture it correctly. The setup and hold times determine the ideal strobe position within the sampling window.

- Receiver skew margin (RSKM) = margin or slack at the receiver capture register.

For TCCS and SW specifications, refer to the *DC and Switching Characteristics* chapter of the *Stratix III Device Handbook*.

The following figure relates this terminology to a timing budget diagram.

**Figure 9-6: Sample Timing Budget Diagram**

The timing budget regions marked " $\frac{1}{2} \times \text{TCCS}$ " represent the latest data valid time and earliest data invalid times for the data transmitter. The region marked sampling window is the time required by the receiver during which data must stay stable. This sampling window comprises the following:

- Internal register setup and hold requirements
- Skew on the data and clock nets within the receiver device
- Jitter and uncertainty on the internal capture clock

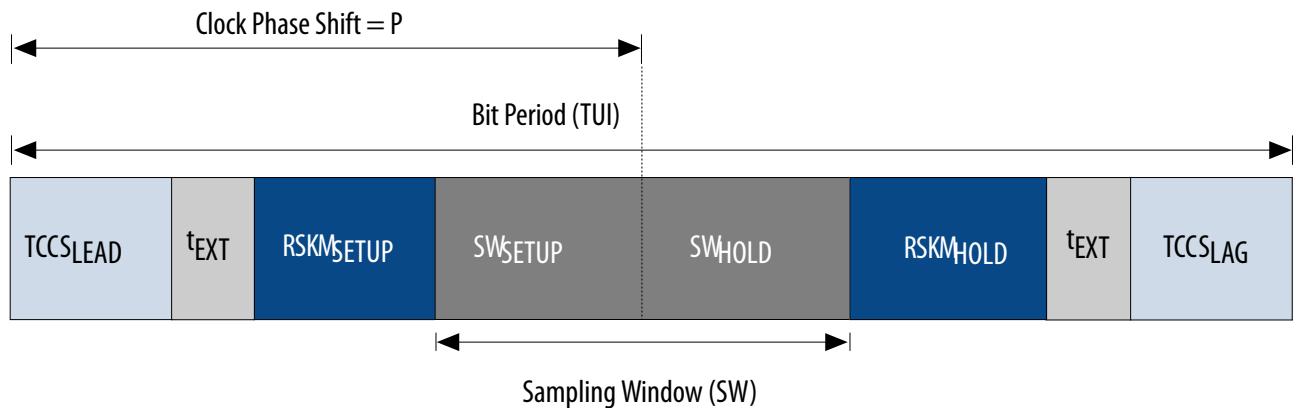
**Note:** The sampling window is not the capture margin or slack, but instead the requirement from the receiver. The margin available is denoted as RSKM.

The simple example illustrated in the preceding figure does not consider any board level uncertainties, assumes a center-aligned capture clock at the middle of the receiver sampling window region, and assumes an evenly distributed TCCS with respect to the transmitter clock pin. In this example, the left end of the bit period corresponds to time  $t = 0$ , and the right end of the bit period corresponds to time  $t = \text{TUI}$  (where TUI stands for time unit interval). Therefore, the center-aligned capture clock at the receiver is best placed at time  $t = \text{TUI}/2$ .

Therefore:

$$\text{the total margin} = 2 \times \text{RSKM} = \text{TUI} - \text{TCCS} - \text{SW}.$$

Consider the case where the clock is not center-aligned within the bit period (clock phase shift =  $P$ ), and the transmitter uncertainties are unbalanced ( $\text{TCCS}_{\text{LEAD}}$  and  $\text{TCCS}_{\text{LAG}}$ ).  $\text{TCCS}_{\text{LEAD}}$  is defined as the skew between the clock signal and latest data valid signal.  $\text{TCCS}_{\text{LAG}}$  is defined as the skew between the clock signal and earliest data invalid signal. Also, the board level skew across data and clock traces are specified as  $t_{\text{EXT}}$ . For this condition, you should compute independent setup and hold margins at the receiver ( $\text{RSKM}_{\text{SETUP}}$  and  $\text{RSKM}_{\text{HOLD}}$ ). In this example, the sampling window requirement is split into a setup side requirement ( $\text{SW}_{\text{SETUP}}$ ) and hold side ( $\text{SW}_{\text{HOLD}}$ ) requirement. The following figure illustrates the timing budget for this condition. A timing budget similar to that shown is used for Stratix III FPGA read and write data timing paths.

**Figure 9-7: Sample Timing Budget with Unbalanced (TCCS and SW) Timing Parameters**

Therefore:

$$\text{Setup margin} = \text{RSKM}_{\text{SETUP}} = P - \text{TCCS}_{\text{LEAD}} - \text{SW}_{\text{SETUP}} - t_{\text{EXT}}$$

$$\text{Hold margin} = \text{RSKM}_{\text{HOLD}} = (TUI - P) - \text{TCCS}_{\text{LAG}} - \text{SW}_{\text{HOLD}} - t_{\text{EXT}}$$

The timing budget illustrated in the first figure with balanced timing parameters applies for calibrated paths where the clock is dynamically center-aligned within the data valid window. The timing budget illustrated in the second figure with unbalanced timing parameters applies for circuits that employ a static phase shift using a DLL or PLL to place the clock within the data valid window.

#### Related Information

- [Read Capture](#) on page 9-18
- [List of designs using Altera External Memory IP](#)
- [Stratix III Device Handbook](#)

#### Read Capture

Memory devices provide edge-aligned DQ and DQS outputs to the FPGA during read operations. Stratix III FPGAs center-aligns the DQS strobe using static DLL-based delays. Stratix III devices use a source synchronous circuit for data capture.

When applying this methodology to read data timing, the memory device is the transmitter and the FPGA device is the receiver.

The transmitter channel-to-channel skew on outputs from the memory device is available from the corresponding device data sheet. Let us examine the TCCS parameters for a DDR2 SDRAM component.

For DQS-based capture:

- The time between DQS strobe and latest data valid is defined as  $t_{DQSQ}$
- The time between earliest data invalid and next strobe is defined as  $t_{QHS}$
- Based on earlier definitions,  $\text{TCCS}_{\text{LEAD}} = t_{DQSQ}$  and  $\text{TCCS}_{\text{LAG}} = t_{QHS}$

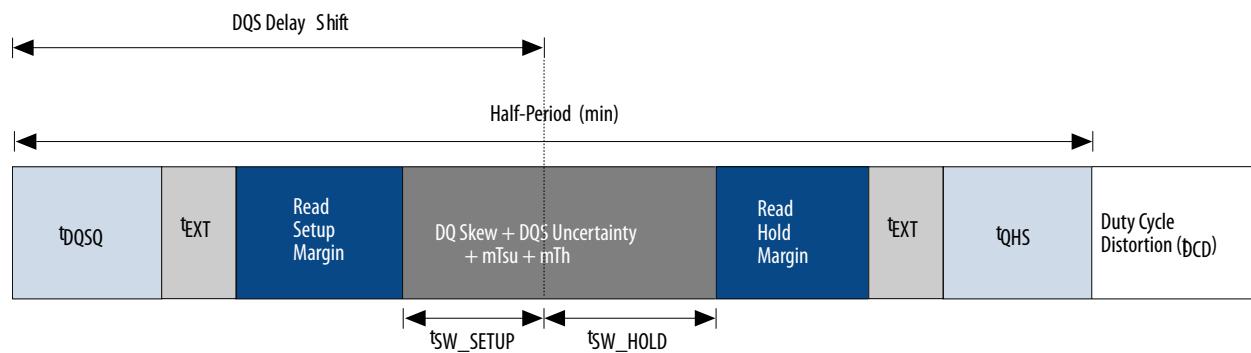
The sampling window at the receiver, the FPGA, includes several timing parameters:

- Capture register micro setup and micro hold time requirements
- DQS clock uncertainties because of DLL phase shift error and phase jitter
- Clock skew across the DQS bus feeding DQ capture registers
- Data skew on DQ paths from pin to input register including package skew

For TCCS and SW specifications, refer to the *DC and Switching Characteristics* chapter of the *Stratix III Device Handbook*.

The following figure shows the timing budget for a read data timing path.

**Figure 9-8: Timing Budget for Read Data Timing Path**



The following table lists a read data timing analysis for a Stratix III –2 speed-grade device interfacing with a 400-MHz DDR2 SDRAM component.

**Table 9-4: Read Data Timing Analysis for Stratix III Device with a 400-MHz DDR2 SDRAM<sup>(1)</sup>**

Parameter	Specifications	Value (ps)	Description
Memory Specifications (1)	t <sub>HP</sub>	1250	Average half period as specified by the memory data sheet, $t_{HP} = 1/2 * t_{CK}$
	t <sub>DCD</sub>	50	Duty cycle distortion = $2\% \times t_{CK} = 0.02 \times 2500$ ps
	t <sub>DQSQ</sub>	200	Skew between DQS and DQ from memory
	t <sub>QHS</sub>	300	Data hold skew factor as specified by memory

Parameter	Specifications	Value (ps)	Description
FPGA Specifications	$t_{SW\_SETUP}$	181	FPGA sampling window specifications for a given configuration (DLL mode, width, location, and so on.)
	$t_{SW\_HOLD}$	306	
Board Specifications	$t_{EXT}$	20	Maximum board trace variation allowed between any two signal traces (user specified parameter)
Timing Calculations	$t_{DVW}$	710	$t_{HP} - t_{DCD} - t_{DQSQ} - t_{QHS} - 2 \times t_{EXT}$
	$t_{DQS\_PHASE\_DELAY}$	500	Ideal phase shift delay on DQS capture strobe $= (\text{DLL phase resolution} \times \text{number of delay stages} \times t_{CK}) / 360^\circ = (36^\circ \times 2 \text{ stages} \times 2500 \text{ ps}) / 360^\circ = 500 \text{ ps}$
Results	Setup margin	99	$RSKM_{SETUP} = t_{DQSQ\_PHASE\_DELAY} - t_{DQSQ} - t_{SW\_SETUP} - t_{EXT}$
	Hold margin	74	$RSKM_{HOLD} = t_{HP} - t_{DCD} - t_{DQS\_PHASE\_DELAY} - t_{QHS} - t_{SW\_HOLD} - t_{EXT}$

#### Notes to Table:

1. This sample calculation uses memory timing parameters from a 72-bit wide 256-MB micron MT9HTF3272AY-80E 400-MHz DDR2 SDRAM DIMM.

#### Related Information

##### [Stratix III Device Handbook](#)

#### Write Capture

During write operations, the FPGA generates a DQS strobe and a center-aligned DQ data bus using multiple PLL-driven clock outputs. The memory device receives these signals and captures them internally. The Stratix III family contains dedicated DDIO (double data rate I/O) blocks inside the IOEs.

For write operations, the FPGA device is the transmitter and the memory device is the receiver. The memory device's data sheet specifies data setup and data hold time requirements based on the input slew

rate on the DQ/DQS pins. These requirements make up the memory sampling window, and include all timing uncertainties internal to the memory.

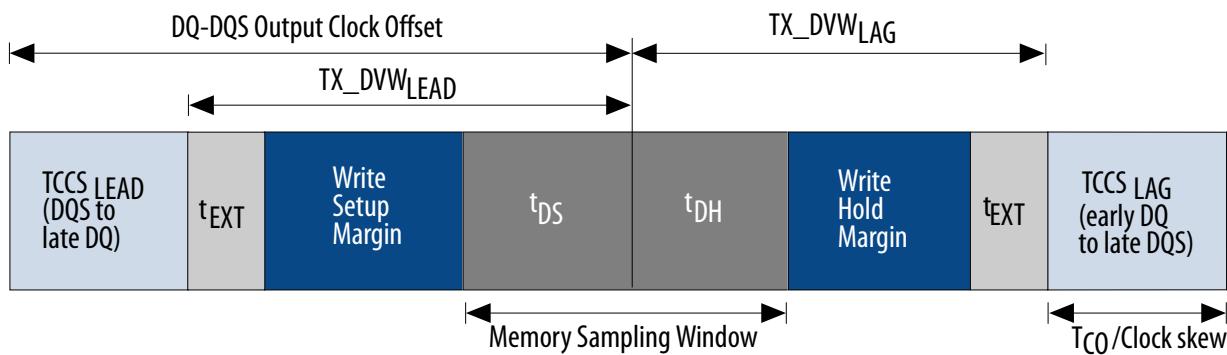
Output skew across the DQ and DQS output pins on the FPGA make up the TCCS specification. TCCS includes contributions from numerous internal FPGA circuits, including:

- Location of the DQ and DQS output pins
- Width of the DQ group
- PLL clock uncertainties, including phase jitter between different output taps used to center-align DQS with respect to DQ
- Clock skew across the DQ output pins, and between DQ and DQS output pins
- Package skew on DQ and DQS output pins

Refer to the *DC and Switching Characteristics* chapter of the *Stratix III Device Handbook* for TCCS and SW specifications.

The following figure illustrates the timing budget for a write data timing path.

**Figure 9-9: Timing Budget for Write Data Timing Path**



The following table lists a write data timing analysis for a Stratix III –2 speed-grade device interfacing with a DDR2 SDRAM component at 400 MHz. This timing analysis assumes the use of a differential DQS strobe with 2.0-V/ns edge rates on DQS, and 1.0-V/ns edge rate on DQ output pins. Consult your memory device's data sheet for derated setup and hold requirements based on the DQ/DQS output edge rates from your FPGA.

**Table 9-5: Write Data Timing Analysis for 400-MHz DDR2 SDRAM Stratix III Device <sup>(1)</sup>**

Parameter	Specifications	Value (ps)	Description
Memory Specifications <sup>(1)</sup>	$t_{HP}$	1250	Average half period as specified by the memory data sheet
	$t_{DSA}$	250	Memory setup requirement (derated for DQ/DQS edge rates and $V_{REF}$ reference voltage)
	$t_{DHA}$	250	Memory hold requirement (derated for DQ/DQS edge rates and $V_{REF}$ reference voltage)
FPGA Specifications	$TCCS_{LEAD}$	229	FPGA transmitter channel-to-channel skew for a given configuration (PLL setting, location, and width).
	$TCCS_{LAG}$	246	
Board Specifications	$t_{EXT}$	20	Maximum board trace variation allowed between any two signal traces (user specified parameter)
Timing Calculations	$t_{OUTPUT\_CLOCK\_OFFSET}$	625	Output clock phase offset between DQ & DQS output clocks = 90°. $t_{OUTPUT\_CLOCK\_OFFSET} = (\text{output clock phase DQ and DQS offset} \times t_{CK})/360^\circ = (90^\circ \times 2500)/360^\circ = 625$
	$TX\_DVW_{LEAD}$	396	Transmitter data valid window = $t_{OUTPUT\_CLOCK\_OFFSET} - TCCS_{LEAD}$
	$TX\_DVW_{LAG}$	379	Transmitter data valid window = $t_{HP} - t_{OUTPUT\_CLOCK\_OFFSET} - TCCS_{LAG}$

Parameter	Specifications	Value (ps)	Description
Results	Setup margin	126	$TX\_DVW_{LEAD} - t_{EXT} - t_{DSA}$
	Hold margin	109	$TX\_DVW_{LAG} - t_{EXT} - t_{DHA}$

Notes to Table:

1. This sample calculation uses memory timing parameters from a 72-bit wide 256-MB micron MT9HTF3272AY-80E 400-MHz DDR2 SDRAM DIMM

#### Related Information

- [Read Capture](#) on page 9-18
- [Stratix III Device Handbook](#)

#### Arria II, Arria V, Cyclone V, Stratix IV and Stratix V

Read capture timing analysis indicates the amount of slack on the DDR DQ signals that are latched by the FPGA using the DQS strobe output of the memory device.

#### Read Capture

The read capture timing paths are analyzed by a combination of the TimeQuest Timing Analyzer using the `set_input_delay (max and min)`, `set_max_delay`, and `set_min_delay` constraints, and further steps to account for calibration that occurs at runtime. The UniPHY IP include timing constraints in the `<phy_variation_name>.sdc` file, and further slack analysis in `<phy_variation_name>_report_timing.tcl` and `<phy_variation_name>_report_timing_core.tcl` files.

In Arria II and Stratix IV devices, the margin is reported based on a combination of the TimeQuest Timing Analyzer calculation results and further processing steps that account for the calibration that occurs at runtime. First, the TimeQuest analyzer returns the base setup and hold slacks, and further processing steps adjust the slacks to account for effects which the TimeQuest analyzer cannot model.

#### Write

Write timing analysis indicates the amount of slack on the DDR DQ signals that are latched by the memory device using the DQS strobe output from the FPGA device. The write timing paths are analyzed by a combination of the TimeQuest Timing Analyzer using the `set_output_delay (max and min)` and further steps to account for calibration that occurs at runtime. The UniPHY IP includes timing constraints in the `<phy_variation_name>.sdc` (UniPHY) file, and further slack analysis in the `<phy_variation_name>_report_timing.tcl` and `<phy_variation_name>_report_timing_core.tcl` files.

#### Read Resynchronization

In a UniPHY interface, a FIFO buffer synchronizes the data transfer from the data capture to the core.

The calibration process sets the depth of the FIFO buffer and no dedicated synchronization clock is required. Refer to `<phy_variation_name>_report_timing_core.tcl` for more information about the resynchronization timing margin equation.

## DQS versus CK—Arria II GX and Cyclone IV Devices

The DQS versus CK timing path indicates the skew requirement for the arrival time of the DQS strobe at the memory with respect to the arrival time of CK/CK# at the memory. Arria II GX and Cyclone IV devices require the DQS strobes and CK clocks to arrive edge aligned.

There are two timing constraints for DQS versus CK timing path to account for duty cycle distortion. The DQS/DQS# rising edge to CK/CK# rising edge ( $t_{DQSS}$ ) requires the rising edge of DQS to align with the rising edge of CK to within 25% of a clock cycle, while the DQS/DQS# falling edge setup/hold time from CK/CK# rising edge ( $t_{DSS}/t_{DSH}$ ) requires the falling edge of DQS to be more than 20% of a clock cycle away from the rising edge of CK.

The TimeQuest Timing Analyzer analyzes the DQS vs CK timing paths using the `set_output_delay (max and min)` constraints. For more information, refer to `<phy_variation_name>_phy_ddr_timing.sdc`.

### Write Leveling tDQSS

In DDR2 SDRAM and DDR3 SDRAM interfaces, write leveling  $t_{DQSS}$  timing is a calibrated path that details skew margin for the arrival time of the DQS strobe with respect to the arrival time of CK/CK# at the memory side.

For proper write leveling configuration, DLL delay chain must be equal to 8. The PHY IP reports the margin through an equation. For more information, refer to `<phy_variation_name>_report_timing_core.tcl`.

### Write Leveling tDSH/tDSS

In DDR2 SDRAM and DDR3 SDRAM interfaces, write leveling  $t_{DSH}/t_{DSS}$  timing details the setup and hold margin for the DQS falling edge with respect to the CK clock at the memory.

The PHY IP reports the margin through an equation. For more information, refer to `<phy_variation_name>_report_timing_core.tcl`.

### DK versus CK (RLDRAM II with UniPHY)

In RLDRAM II with UniPHY designs using the Nios-based sequencer, DK versus CK timing is a calibrated path that details skew margin for the arrival time of the DK clock versus the arrival time of CK/CK# on the memory side.

The PHY IP reports the margin through an equation. For more information, refer to `<phy_variation_name>_report_timing_core.tcl`.

### Bus Turnaround Time

In DDR2 and DDR3 SDRAM, and RLDRAM II (CIO) with UniPHY designs that use bidirectional data bus, you may have potential encounter with data bus contention failure when a write command follows a read command. The bus-turnaround time analysis determines how much margin there is on the switchover time and prevents bus contention.

If the timing is violated, you can either increase the controller's bus turnaround time, which may reduce efficiency or board traces delay. Refer to `<variation>_report_timing_core.tcl` for the equation. You can find this analysis in the timing report. This analysis is only available for DDR2/3 SDRAM and RLDRAM II UniPHY IPs in Arria II GZ, Arria V, Cyclone V, Stratix IV, and Stratix V devices.

To determine whether the bus turnaround time issue is the cause of your design failure and to overcome this timing violation, follow these steps:

1. When the design fails, change the default values of `MEM_IF_WR_TO_RD_TURNAROUND_OCT` and `MEM_IF_RD_TO_WR_TURNAROUND_OCT` parameters in the controller wrapper file to a maximum value of 5. If the design passes after the change, it is a bus turnaround issue.
2. To solve the bus turnaround time issue, reduce the values of the `MEM_IF_WR_TO_RD_TURNAROUND_OCT` and `MEM_IF_RD_TO_WR_TURNAROUND_OCT` parameters gradually until you reach the minimum value needed for the design to pass on board.

## Timing Analysis Description for Arria 10 EMIF IP

Timing analysis of Arria 10 external memory interface IP is somewhat simpler than that of UniPHY-based IP, because Arria 10 devices have more hardened blocks and there are fewer soft logic registers to be analyzed, because most are user logic registers.

Your Arria 10 EMIF IP includes a Synopsys Design Constraints File (**.sdc**) which contains timing constraints specific to your IP. The **.sdc** file also contains Tool Command Language (**.tcl**) scripts which perform various timing analyses specific to memory interfaces.

Two timing analysis flows are available for Arria 10 EMIF IP:

- Early I/O Timing Analysis, which is a precompilation flow.
- Full Timing Analysis, which is a post-compilation flow.

### PHY or Core

Timing analysis of the PHY or core path includes path from last set of registers in core to first set of registers to periphery (C2P), path from last set of registers in periphery to first of registers in core (P2C) and ECC related path if it is enabled.

As with 28 nm devices, core timing analysis excludes user logic timing to or from Altera EMIF blocks. The Arria 10 EMIF IP provides a constrained clock (for example: `ddr3_usr_clk`) with which to clock customer logic. In 28 nm devices, `pll_afi_clk` is used for this purpose.

The PHY or core analyzes this path by calling the `report_timing` command in `<variation_name>_report_timing.tcl` and `<variation_name>_report_timing_core.tcl`.

**Note:** In version 14.1, you may notice that the *Spatial Pessimism Removal* slack values in the **Core to Periphery** and **Periphery to Core** tables are always equal to zero. This occurs because pessimism removal is now integrated into the base timing analysis.

### I/O Timing

I/O timing analysis includes analysis of read capture, write, address and command, DQS gating, and write leveling.

The TimeQuest Timing Analyzer provides a breakdown of the timing budgets which details margin loss due to transmitter, receiver, and channel. TimeQuest displays the total margin in the last row of the timing report.

The I/O timing analysis described in the following topics is based on an Arria 10 -2 speed-grade device, interfacing with a DDR3 SDRAM UDIMM at 1066 MHz. A 1066 MHz DDR3 SDRAM UDIMM is used for the analysis.

#### Read Capture

Read capture timing analysis indicates the amount of slack on the DDR DQ signals that are latched by the FPGA using the DQS strobe output of the memory device.

The TimeQuest Timing Analyzer analyzes read capture timing paths through conventional static timing analysis and further processing steps that account for memory calibration (which may include pessimism removal) and calibration uncertainties as shown in the following figure.

**Figure 9-10: Read Capture Timing Analysis**

ed_synth_ddr3w Read Capture		
	Operation	Margin
1	Ideal Timing Window	0.469
2	ISI	0.063
3	SSI	0.023
4	Slew Rate Derating	0.050
5	tDQSQ effect	0.075
6	tQH effect	0.112
7	Memory Calibration	-0.075
8	Jitter Effects	0.072
9	Duty Cycle Distortion	0.031
10	Setup/Hold Time	0.016
11	EOL	0.025
12	Calibration Uncertainty	0.036
13	Skew Effect	0.000
14	Final Read Margin	0.040

The diagram illustrates the breakdown of the total Read Capture margin into three main categories: Channel Effects, Transmitter Effects (Memory), and Receiver Effects (FPGA). Each category is represented by a box with a red bracket grouping the corresponding rows from the table above. The 'Channel Effects' box groups rows 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13. The 'Transmitter Effects (Memory)' box groups row 2. The 'Receiver Effects (FPGA)' box groups row 14.

## Write

Write timing analysis indicates the amount of slack on the DDR DQ signals that are latched by the memory device using the DQS strobe output from the FPGA device.

As with read capture, the TimeQuest Timing Analyzer analyzes write timing paths through conventional static timing analysis and further processing steps that account for memory calibration (which may include pessimism removal) and calibration uncertainties as shown in the following figure.

**Figure 9-11: Write Timing Analysis**

ed_synth_ddr3w Write		
	Operation	Margin
1	Ideal Timing Window	0.469
2	ISI	0.063
3	SSO	0.047
4	Slew Rate Derating	0.118
5	tDS effect	0.053
6	tDH effect	0.055
7	Memory Calibration	-0.043
8	Jitter Effects	0.039
9	Duty Cycle Distortion	0.016
10	EOL	0.013
11	Calibration Uncertainty	0.038
12	Skew Effect	0.000
13	Final Write Margin	0.071

The diagram illustrates the breakdown of the total Write margin into three main categories: Channel Effects, Receiver Effects (Memory), and Transmitter Effects (FPGA). Each category is represented by a box with a red bracket grouping the corresponding rows from the table above. The 'Channel Effects' box groups rows 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, and 13. The 'Receiver Effects (Memory)' box groups row 2. The 'Transmitter Effects (FPGA)' box groups row 13.

## Address and Command

Address and command signals are single data rate signals latched by the memory device using the FPGA output clock; some are half-rate data signals, while others, such as the chip select, are full-rate signals.

The TimeQuest Timing Analyzer analyzes the address and command timing paths through conventional static timing analysis and further processing steps that account for memory pessimism removal (as

shown in the following figure). Depending on the memory protocol in use, if address command calibration is performed, calibration uncertainty is subtracted from the timing window while PVT variation and skew effects are not subtracted, and vice versa

**Figure 9-12: Address and Command Timing Analysis**

ed_synth_ddr3w Address/Command		
	Operation	Margin
1	Ideal Timing Window	0.937
2	ISI	0.094
3	SSO	0.023
4	Slew Rate Derating	0.118
5	tIS effect	0.060
6	tIH effect	0.095
7	Memory Calibration	-0.046
8	Jitter Effects	0.083
9	Duty Cycle Distortion	0.016
10	EOL	0.013
11	Calibration Uncertainty	0.041
12	PVT variation	0.027
13	Skew Effect	0.000
14	Final CA Margin	0.442

The diagram illustrates the breakdown of the total Address/Command timing margin into three primary categories of effects:

- Channel Effects**: This category includes ISI, SSO, and tIS/tIH effects.
- Receiver Effects (Memory)**: This category includes Memory Calibration, Jitter Effects, Duty Cycle Distortion, EOL, and Calibration Uncertainty.
- Transmitter Effects (FPGA)**: This category includes Slew Rate Derating, tDH effect, PVT variation, Skew Effect, and Final CA Margin.

### DQS Gating / Postamble

Postamble timing is a setup period during which the DQS signal goes low after all the DQ data has been received from the memory device during a read operation. After postamble time, the DQS signal returns from a low-impedance to a high-impedance state to disable DQS and disallow any glitches from writing false data over valid data.

The TimeQuest Timing Analyzer analyzes the postamble timing path in DDRx memory protocols only through an equation which considers memory calibration, calibration uncertainty, and tracking uncertainties as shown in the following figure.

**Figure 9-13: DQS Gating Timing Analysis**

ed_synth_ddr3w DQS Gating		
	Operation	Margin
1	Ideal Timing Window	0.750
2	ISI	0.094
3	SSI	0.012
4	Slew Rate Derating	0.050
5	tDQSCK	0.360
6	Memory Calibration	-0.144
7	Jitter Effects	0.167
8	Duty Cycle Distortion	0.002
9	EOL	0.002
10	Calibration Uncertainty	0.020
11	Tracking Uncertainty	0.078
12	Setup/Hold Time	0.016
13	Skew Effect	0.000
14	Final DQS Gating Margin	0.093

The diagram illustrates the breakdown of the total DQS Gating timing margin into three primary categories of effects:

- Channel Effects**: This category includes ISI and SSI.
- Transmitter Effects (Memory)**: This category includes tDQSCK, Memory Calibration, and Calibration Uncertainty.
- Receiver Effects (FPGA)**: This category includes Slew Rate Derating, Jitter Effects, Duty Cycle Distortion, EOL, Tracking Uncertainty, Setup/Hold Time, and Skew Effect.

### Write Leveling

In DDR3 SDRAM and DDR4 SDRAM interfaces, write leveling details the margin for the DQS strobe with respect to CK/CK# at the memory side.

The TimeQuest Timing Analyzer analyzes the write leveling timing path through an equation which considers memory calibration, calibration uncertainty and PVT variation as shown in the following figure.

**Figure 9-14: Write Leveling Timing Analysis**

ed_synth_ddr3w Write Levelling		
	Operation	Margin
1	Ideal Timing Window	0.937
2	ISI	0.031
3	SSO	0.035
4	tDQSS/tDSS/tDSH Effect	0.431
5	Memory Calibration	-0.172
6	tWLS/tWLH effect	0.000
7	Jitter Effects	0.165
8	Duty Cycle Distortion	0.016
9	EOL	0.000
10	Calibration Uncertainty	0.030
11	PVT variation	0.862
12	Skew Effect	0.000
13	Final Write Levelling Margin	0.401

The diagram illustrates the breakdown of the final write leveling margin into three categories: Channel Effects, Receiver Effects (Memory), and Transmitter Effects (FPGA). Red curly braces group the rows in the table into these categories. The 'Channel Effects' group includes rows 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, and 13. The 'Receiver Effects (Memory)' group includes rows 11 and 12. The 'Transmitter Effects (FPGA)' group includes row 13.

## Timing Report DDR

The **Report DDR** task in the TimeQuest Timing Analyzer generates custom timing margin reports for all UniPHY and Arria 10 EMIF IP instances in your design. The TimeQuest Timing Analyzer generates this custom report by sourcing the wizard-generated `<variation_name>_report_timing.tcl` script.

This `<variation_name>_report_timing.tcl` script reports the following timing slacks on specific paths of the DDR SDRAM:

- Read capture
- Read resynchronization
- Mimic, address and command
- Core
- Core reset and removal
- Half-rate address and command
- DQS versus CK
- Write
- Write leveling (t<sub>DQSS</sub>)
- Write leveling (t<sub>DSS</sub>/t<sub>DSH</sub>)
- DQS Gating (Postamble)

In Stratix III designs, the `<variation_name>_report_timing.tcl` script checks the design rules and assumptions as listed in “Timing Model Assumptions and Design Rules”. If you do not adhere to these assumptions and rules, you receive critical warnings when the TimeQuest Timing Analyzer runs during compilation or when you run the **Report DDR** task.

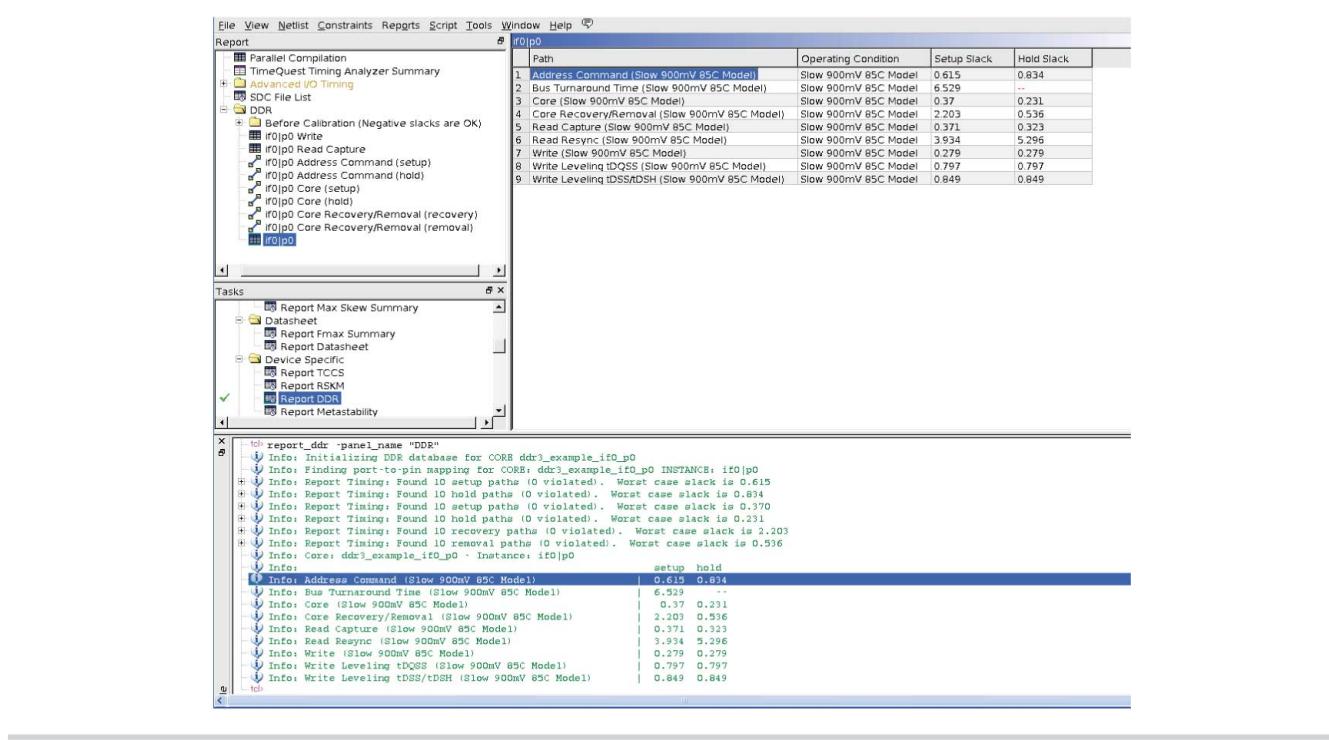
To generate a timing margin report, follow these steps:

1. Compile your design in the Quartus Prime software.
  2. Launch the TimeQuest Timing Analyzer.
  3. Double-click **Report DDR** from the **Tasks** pane. This action automatically executes the **Create Timing Netlist**, **Read SDC File**, and **Update Timing Netlist** tasks for your project.
- The **.sdc** may not be applied correctly if the variation top-level file is the top-level file of the project. You must have the top-level file of the project instantiate the variation top-level file.

The **Report DDR** feature creates a new DDR folder in the TimeQuest Timing Analyzer **Report** pane.

Expanding the DDR folder reveals the detailed timing information for each PHY timing path, in addition to an overall timing margin summary for the UniPHY instance, as shown in the following figure.

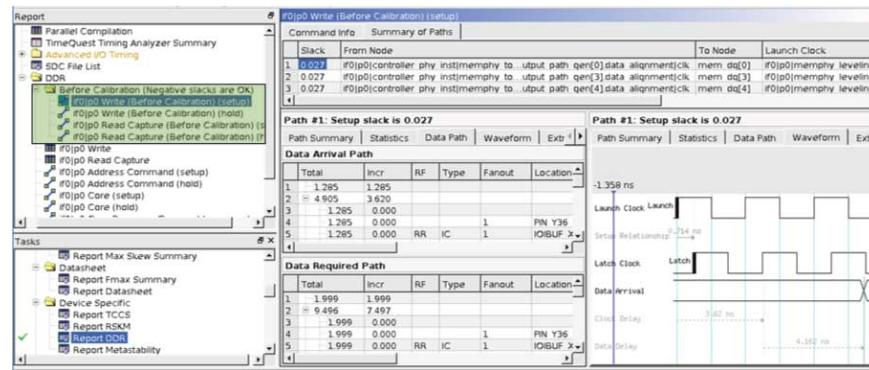
**Figure 9-15: Timing Margin Summary Window Generated by Report DDR Task**



**Note:** Bus turnaround time shown in the above figure is available in all UniPHY IPs and devices except in QDR II and QDR II+ SRAM memory protocols and Stratix III devices.

The following figure shows the timing analysis results calculated using FPGA timing model before adjustment in the **Before Calibration** panel.

Figure 9-16: Read and Write Before Calibration



The following two figures show the read capture and write margin summary window generated by the Report DDR Task for a DDR3 core. It first shows the timing results calculated using the FPGA timing model. The `<variation_name>_report_timing_core.tcl` then adjusts these numbers to account for effects that are not modeled by either the timing model or by TimeQuest Timing Analyzer. The read and write timing margin analysis for Stratix III devices does not need any adjustments.

Figure 9-17: Read Capture Margin Summary Window

if0 p0 Read Capture			
	Operation	Setup Slack	Hold Slack
1	After Calibration Read Capture	0.371	0.323
2	Before Calibration Read Capture	0.280	0.273
3	Memory Calibration	0.078	0.150
4	Deskew Read	0.157	0.044
5	Quantization error	-0.050	-0.050
6	Calibration uncertainty	-0.093	-0.093

Figure 9-18: Write Capture Margin Summary Window

if0 p0 Write			
	Operation	Setup Slack	Hold Slack
1	After Calibration Write	0.279	0.279
2	Before Calibration Write	0.027	0.026
3	Memory Calibration	0.135	0.113
4	Deskew Write and/or more clock pessimism removal	0.216	0.240
5	Quantization error	-0.050	-0.050
6	Calibration uncertainty	-0.050	-0.050

#### Related Information

[Timing Model Assumptions and Design Rules](#) on page 9-33

## Report SDC

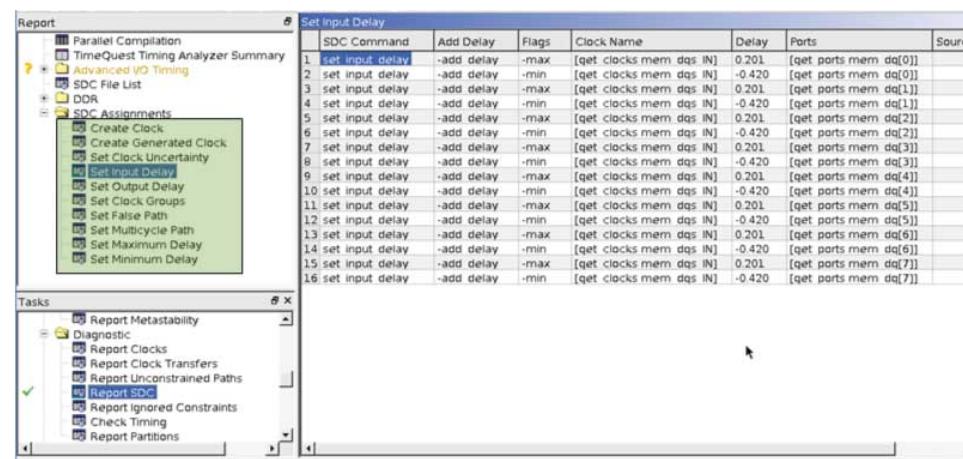
The **Report SDC** task in the TimeQuest Timing Analyzer generates the SDC assignment reports for your design. The TimeQuest Timing Analyzer generates this constraint report by sourcing the `.sdc`. The SDC assignment reports show the constraint applied in the design.

For example, the reports may include the following constraints:

- Create Clock
- Create Generated Clock
- Set Clock Uncertainty
- Set Input Delay
- Set Output Delay
- Set False Path
- Set Multicycle Path
- Set Maximum Delay
- Set Minimum Delay

The following figure shows the SDC assignments generated by the **Report SDC** task for a DDR3 SDRAM core design. The timing analyzer uses these constraint numbers in analysis to calculate the timing margin. Refer to the **.sdc** files of each constraints number.

**Figure 9-19: SDC Assignments Report Window**



## Calibration Effect in Timing Analysis

Timing analysis for Arria II, Arria V, Arria V GZ, Cyclone IV, Cyclone V, Stratix IV, and Stratix V devices take into account the calibration effects to improve the timing margin.

The following topics discuss ways to include the calibration effects in timing analysis.

### Calibration Emulation for Calibrated Path

In conventional static timing analysis, calibration paths do not include calibration effects. To account for the calibration effects, the timing analyzer emulates the calibration process and integrates it into the timing analysis.

Normally the calibration process involves adding or subtracting delays to a path. The analyzer uses the delay obtained through static timing analysis in the emulation algorithm to estimate the extra delay added during calibration. With these estimated delays, the timing analysis emulates hardware calibration and obtains a better estimate timing margin.

**Note:** Refer to **<phy\_variation\_name>\_report\_timing.tcl** and **<phy\_variation\_name>\_report\_timing\_core.tcl** for the files that determine the timing margin after calibration.

## Calibration Error or Quantization Error

Hardware devices use calibration algorithms when delay information is unknown or incomplete. If the delay information is unknown, the timing analysis of the calibrated paths has to work with incomplete data. This unknown information may cause the timing analysis calibration operations to pick topologies that are different than what would actually occur in hardware.

The differences between what can occur in hardware and what occurs in the timing analysis are quantified and included in the timing analysis of the calibrated paths as quantization error or calibration error.

## Calibration Uncertainties

Calibration results may change or reduce due to one or more of the following uncertainties:

- Jitter and DCD effects
- Voltage and temperature variations
- Board trace delays changing due to noise on terminated supply voltages

These calibration uncertainties are accounted for in the timing analysis.

## Memory Calibration

All the timing paths reported include one or more memory parameters, such as  $t_{DQSS}$  and  $t_{DQSQ}$ . These specifications indicate the amount of variation that occurs in various timing paths in the memory and abstracts them into singular values so that they can be used by others when interfacing with the memory device.

JEDEC defines these parameters in their specification for memory standards, and every memory vendor must meet this specification or improve it. However, there is no proportion of each specification due to different types of variations. Variations that are of interest are typically grouped into three different types: process variations (P), voltage variations (V), and temperature variations (T). These together compose PVT variations that typically define the JEDEC specification. You can determine the maximum P variation by comparing different dies, and you can determine the maximum V and T variations by operating a design at the endpoints of the range of voltage and temperature. P variations do not change once the chip has been fabricated, while V and T variations change over time.

The timing analysis for Stratix V FPGAs at 667 MHz of various paths (if the analysis is comprehensive and includes all the sources of noise) indicate that there is no timing margin available. However, the designs do actually work in practice with a reasonable amount of margin. The reason for this behavior is that the memory devices typically have specifications that easily beat the JEDEC specification and that our calibration algorithms calibrate out the process portion of the JEDEC specification, leaving only the V and T portions of the variations.

The memory calibration figure determination includes noting what percentage of the JEDEC specification of various memory parameters is caused by process variations for which UniPHY calibration algorithms can calibrate out, and to apply that to the full JEDEC specification. The remaining portion of the variation is caused by voltage and temperature variations which cannot be calibrated out.

You can find the percentage of the JEDEC specification that is due to process variation is set in `<variation_name>_report_timing.tcl`.

## Timing Model Assumptions and Design Rules

External memory interfaces using Altera IP are optimized for highest performance, and use a high-performance timing model to analyze calibrated and source-synchronous, double-data rate I/O timing paths. This timing model applies to designs that adhere to a set of predefined assumptions.

These timing model assumptions include memory interface pin-placement requirements, PLL and clock network usage, I/O assignments (including I/O standard, termination, and slew rate), and many others.

For example, the read and write datapath timing analysis is based on the FPGA pin-level  $t_{TCCS}$  and  $t_{SW}$  specifications, respectively. While calculating the read and write timing margins, the Quartus Prime software analyzes the design to ensure that all read and write timing model assumptions are valid for your variation instance.

**Note:** Timing model assumptions only apply to Stratix III devices.

When the Report DDR task or **report\_timing.tcl** script is executed, the timing analysis assumptions checker is invoked with specific variation configuration information. If a particular design rule is not met, the Quartus Prime software reports the failing assumption as a Critical Warning message.

The following figure shows a sample set of messages generated when the memory interface DQ, DQS, and CK/CK# pins are not placed in the same edge of the device.

**Figure 9-20: Read and Write Timing Analysis Assumption Verification**

A screenshot of a terminal window titled "tcl: report\_ddr -panel name "DDR"" showing a series of critical warning messages. The messages indicate that various memory data (mem\_dq) and control (mem\_dqs, mem\_clk) pins were placed on different edges of the device, which violates timing assumptions. The window also shows the command "Info: Report Timing: Found 24 setup paths (0 violated). Worst case slack is 1.155".

```
477 tcl: report_ddr -panel name "DDR"
478 Critical Warning: Pin mem_clk[0], mem_dq[0], mem_dq[10], mem_dq[11], mem_dq[12], mem_dq[13], mem_dq[14], mem_dq[15], mem_dq[16], mem_dq[17], mem_dq[18] was placed on the left edge of the device
479 Critical Warning: mem_dq[0] was placed on the bottom edge of the device
480 Critical Warning: mem_dq[10] was placed on the bottom edge of the device
481 Critical Warning: mem_dq[11] was placed on the bottom edge of the device
482 Critical Warning: mem_dq[12] was placed on the bottom edge of the device
483 Critical Warning: mem_dq[13] was placed on the bottom edge of the device
484 Critical Warning: mem_dq[14] was placed on the bottom edge of the device
485 Critical Warning: mem_dq[15] was placed on the bottom edge of the device
486 Critical Warning: mem_dq[16] was placed on the bottom edge of the device
487 Critical Warning: mem_dq[17] was placed on the bottom edge of the device
488 Critical Warning: mem_dq[18] was placed on the bottom edge of the device
489 Critical Warning: mem_dq[9] was placed on the bottom edge of the device
551 Critical Warning: mem_dqs[0] was placed on the bottom edge of the device
552 Critical Warning: mem_dqs[1] was placed on the bottom edge of the device
553 Critical Warning: mem_dqs[2] was placed on the bottom edge of the device
554 Critical Warning: mem_dqs[3] was placed on the bottom edge of the device
555 Critical Warning: mem_dqs[4] was placed on the bottom edge of the device
556 Critical Warning: mem_dqs[5] was placed on the bottom edge of the device
557 Critical Warning: mem_dqs[6] was placed on the bottom edge of the device
558 Critical Warning: mem_dqs[7] was placed on the bottom edge of the device
559 Critical Warning: mem_dqs[8] was placed on the bottom edge of the device
560 Critical Warning: Memory_clock_pin mem_clk[0], mem_clk[1], mem_clk[2] must be placed on the same edge of the device
561 Critical Warning: mem_clk[0] was placed on the left edge of the device
562 Critical Warning: mem_clk[1] was placed on the top edge of the device
563 Critical Warning: mem_clk[2] was placed on the bottom edge of the device
565 Critical Warning: Read Capture and Write timing analyses may not be valid due to violated timing model assumptions
566 Info: Report Timing: Found 24 setup paths (0 violated). Worst case slack is 1.155
```

## Memory Clock Output Assumptions

To verify the quality of the FPGA clock output to the memory device (CK/CK# or K/K#), which affects FPGA performance and quality of the read clock/strobe used to read data from the memory device, the following assumptions are necessary:

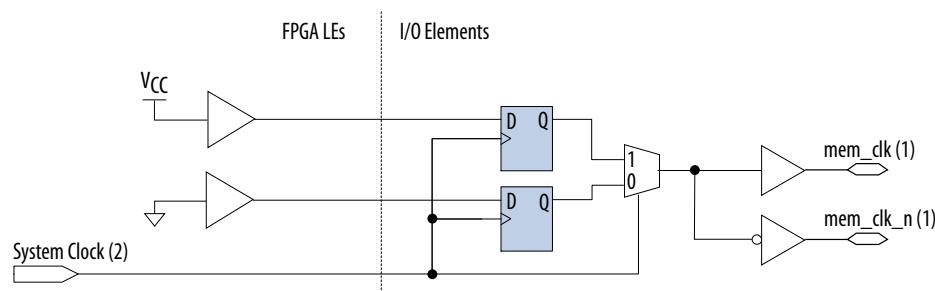
- The slew rate setting must be **Fast** or an on-chip termination (OCT) setting must be used.
- The output delay chains must all be **0** (the default value applied by the Quartus Prime software). These delay chains include the Stratix III D5 and D6 output delay chains.
- The output open-drain parameter on the memory clock pin `IO_OBUF` atom must be **Off**. The **Output Open Drain** logic option must not be enabled.
- The weak pull-up on the CK and CK# pads must be **Off**. The **Weak Pull Up Resistor** logic option must not be enabled.
- The bus hold on the CK and CK# pads must be **Off**. The **Enable Bus Hold Circuitry** logic option must not be enabled.
- All CK and CK# pins must be declared as output-only pins or bidirectional pins with the output enable set to  $V_{CC}$ .

## Memory Clock Assumptions for Stratix III Devices

For Stratix III devices the following additional memory clock assumptions are necessary:

- All memory clock output pins must be placed on `DIFFOUT` pin pairs on the same edge of the device.
- For DDR3 SDRAM interfaces:
  - The CK pins must be placed on FPGA output pins marked DQ, DQS, or DQSn.
  - The CK pin must be fed by an `OUTPUT_PHASE_ALIGNMENT` WYSIWYG with a  $0^\circ$  phase shift.
  - The PLL clock driving CK pins must be the same as the clock driving the DQS pins.
  - The T4 (`DDIO_MUX`) delay chains setting for the memory clock pins must be the same as the settings for the DQS pins.
- For non-DDR3 interfaces, the T4 (`DDIO_MUX`) delay chains setting for the memory clock pins must be greater than 0.
- The programmable rise and fall delay chain settings for all memory clock pins must be set to 0.
- The memory output clock signals must be generated with the DDIO configuration shown in the following figure, with a signal splitter to generate the n- pin pair and a regional clock network-to-clock to output DDIO block.

**Figure 9-21: DDIO Configuration with Signal Splitter**



## Write Data Assumptions

To verify the memory interface using the FPGA TCCS output timing specifications, the following assumptions are necessary:

- For QDRII and QDRII+ memory interfaces, the write clock output pins (such as K/K# or DK/DK#) must be placed in DQS/DQSn pin pairs.
- The PLL clock used to generate the write-clock signals and the PLL clock used to generate the write-data signals must come from the same PLL.
- The slew rate for all write clocks and write data pins must be set to **Fast** or OCT must be used.
- When auto deskew is not enabled, the output delay chains and output enable delay chains must all be set to the default values applied by the Quartus Prime software. These delay chains include the Stratix III D5 and D6 delay chains.
- The output open drain for all write clocks and write data pins' IO\_OBUF atom must be set to **Off**. The **Output Open Drain** logic option must not be enabled.
- The weak pull-up for all write clocks and write data pins must be set to **Off**. The **Weak Pull Up Resistor** logic option must not be enabled.
- The Bus Hold for all write clocks and write data pins must be set to **Off**. The **Enable Bus Hold Circuitry** logic option must not be enabled.

## Write Data Assumptions for Stratix III Devices

For Stratix III devices the following additional write data assumptions are necessary:

- Differential write clock signals (DQS/DQSn) must be generated using the signal splitter.
- The write data pins (including the DM pins) must be placed in related DQ pins associated with the chosen DQS pin. The only exception to this rule is for QDRII and QDRII+ ×36 interfaces emulated using two ×18 DQ groups. For such interfaces, all of the write data pins must be placed on the same edge of the device (left, right, top, or bottom). Also, the write clock K/K# pin pair should be placed on one of the DQS/DQSn pin pairs on the same edge.
- All write clock pins must have similar circuit structure.
  - For DDR2 SDRAM interfaces and DDR3 SDRAM with leveling interfaces, all DQS/DQS# write strobes must be fed by DDIO output registers clocked by the write-leveling delay chain in the OUTPUT\_PHASE\_ALIGNMENT block.
  - For DDR and DDR2 SDRAM interfaces, all write clock pins must be fed by DDIO output registers clocked by a global or regional clock network.
- All write data pins must have similar circuit structure.
  - For DDR3 SDRAM interfaces, all write data pins must be fed by either DDIO output registers clocked by the OUTPUT\_PHASE\_ALIGNMENT block, V<sub>CC</sub>, or GND.
  - For DDR and DDR2 SDRAM interfaces, all write data pins must be fed by either DDIO output registers clocked by a global or regional clock network, V<sub>CC</sub>, or GND.
- The write clock output must be 72°, 90°, or 108° more than the write data output.
  - For DDR2 SDRAM and DDR3 SDRAM with leveling interfaces, the write-leveling delay chain in the OUTPUT\_PHASE\_ALIGNMENT block must implement a phase shift of 72°, 90°, or 108° to center-align write clock with write data.
  - For DDR and DDR2 SDRAM interfaces, the phase shift of the PLL clock used to clock the write clocks must be 72 to 108° more than the PLL clock used to clock the write data clocks to generated center-aligned clock and data.
- The T4 (DDIO\_MUX) delay chains must all be set to 3. When differential DQS (using splitter) is used, T4 must be set to 2.
- The programmable rise and fall delay chain settings for all memory clock pins must be set to 0.

The following table lists I/O standards supported for the write clock and write data signals for each memory type and pin location.

**Table 9-6: I/O standards**

Memory Type	Placement	Legal I/O Standards for DQS	Legal I/O Standards for DQ
DDR3 SDRAM	Row I/O	Differential 1.5-V SSTL Class I	1.5-V SSTL Class I
DDR3 SDRAM	Column I/O	Differential 1.5-V SSTL Class I Differential 1.5-V SSTL Class II	1.5-V SSTL Class I 1.5-V SSTL Class II
DDR2 SDRAM	Any	SSTL-18 Class I SSTL-18 Class II Differential 1.8V SSTL Class I Differential 1.8V SSTL Class II	SSTL-18 Class I SSTL-18 Class II
DDR SDRAM	Any	SSTL-2 Class I SSTL-2 Class II	SSTL-2 Class I SSTL-2 Class II
QDRII and QDR II + SRAM	Any	HSTL-1.5 Class I HSTL-1.8 Class I	HSTL-1.5 Class I HSTL-1.8 Class I
RDRAM II	Any	HSTL-1.5 Class I HSTL-1.8 Class I	HSTL-1.5 Class I HSTL-1.8 Class I

## Read Data Assumptions

To verify that the external memory interface can use the FPGA Sampling Window (SW) input timing specifications, the following assumptions are necessary:

- The read clocks input pins must be placed on DQS pins. DQS/DQS# inputs must be placed on differential DQS/DQSn pins on the FPGA.
- Read data pins (DQ) must be placed on the DQ pins related to the selected DQS pins.
- For QDR II and QDR II+ SRAM interfaces, the complementary read clocks must have a single-ended I/O standard setting of HSTL-18 Class I or HSTL-15 Class I.
- For RDRAM II interfaces, the differential read clocks must have a single ended I/O standard setting of HSTL 18 Class I or HSTL 15 Class I.

## Read Data Assumptions for Stratix III Devices

For Stratix III devices the following additional read data and mimic pin assumptions are necessary:

- For DDR3, DDR2, and DDR SDRAM interfaces, the read clock pin can only drive a DQS bus clocking a  $\times 4$  or  $\times 9$  DQ group.
- For QDR II, QDR II+ SRAM, and RLDRAM II interfaces, the read clock pin can only drive a DQS bus clocking a  $\times 9$ ,  $\times 18$ , or  $\times 36$  DQ group.
- For non-wraparound DDR, DDR2, and DDR3 interfaces, the mimic pin, all read clock, and all read data pins must be placed on the same edge of the device (top, bottom, left, or right). For wraparound interfaces, these pins can be placed on adjacent row I/O and column I/O edges and operate at reduced frequencies.
- All read data pins and the mimic pin must feed DDIO\_IN registers and their input delay chains D1, D2, and D3 set to default values.
- DQS phase-shift setting must be either  $72^\circ$  or  $90^\circ$  (supports only one phase shift for each operating band and memory standard).
- All read clock pins must have the `dqs_ctrl_latches_enable` parameter of its DQS\_DELAY\_CHAIN WYSIWYG set to false.
- The read clocks pins must have their D4 delay chain set to the Quartus Prime software default value of **0**.
- The read data pins must have their T8 delay chain set to the Quartus Prime software default value of **0**.
- When differential DQS strobes are used (DDR3 and DDR2 SDRAM), the mimic pin must feed a true differential input buffer. Placing the memory clock pin on a `DIFFIO_RX` pin pair allows the mimic path to track timing variations on the DQS input path.
- When single ended DQS strobes are used, the mimic pin must feed a single ended input buffer.

## DLL Assumptions

The following DLL assumptions are necessary:

- The DLL must directly feed its `delayctrlout[]` outputs to all DQS pins without intervening logic or inversions.
- The DLL must be in a valid frequency band of operation as defined in the corresponding device data sheet.
- The DLL must have jitter reduction mode and dual-phase comparators enabled.

## PLL and Clock Network Assumptions for Stratix III Devices

To verify that the memory interface's PLL is configured correctly, the following assumptions are necessary:

- The PLL that generates the memory output clock signals and write data and clock signals must be set to **No compensation** mode to minimize output clock jitter.
- The reference input clock signal to the PLL must be driven by the dedicated clock input pin located adjacent to the PLL, or from the clock output signal from the adjacent PLL. If the reference clock cascades from another PLL, that upstream PLL must be in **No compensation** mode and **Low bandwidth** mode.
- For DDR3 and DDR2 SDRAM interfaces, use only regional or dual regional clock networks to route PLL outputs that generate the write data, write clock, and memory output clock signals. This requirement ensures that the memory output clocks (CK/CK#) meet the memory device input clock jitter specifications, and that output timing variations or skews are minimized.
- For other memory types, the same clock tree type (global, regional, or dual regional) is recommended for PLL clocks generating the write clock, write data, and memory clock signals to minimize timing variations or skew between these outputs.

## Common Timing Closure Issues

The following topics describe potential timing closure issues that can occur when using the UniPHY IP.

For possible timing closure issues with UniPHY variations, refer to the *Quartus Prime Software Release Notes* for the software version that you are using. You can solve some timing issues by moving registers or changing the project fitting setting to **Standard** (from **Auto**).

The *Quartus Prime Software Release Notes* list common timing issues that can be encountered in a particular version of the Quartus Prime software.

**Note:** In UniPHY-based memory controllers, the `derive_pll_clocks` command can affect timing closure if it is called before the memory controller files are loaded. Ensure that the Quartus Prime IP File (**.qip**) appears in the file list before any Synopsys Design Constraint Files (**.sdc**) files that contain `derive_pll_clocks`.

### Related Information

[Quartus Prime Software and Device Support Release Notes](#)

## Missing Timing Margin Report

The UniPHY timing margin reports may not be generated during compilation if the **.sdc** does not appear in the Quartus Prime project settings.

Timing margin reports are not generated if you specify the UniPHY variation as the top-level project entity. Instantiate the UniPHY variation as a lower level module in your user design or memory controller.

## Incomplete Timing Margin Report

The timing report may not include margin information for certain timing paths if certain memory interface pins are optimized away during synthesis.

Verify that all memory interface pins appear in the `<variation>_all_pins.txt` file generated during compilation, and ensure that they connect to the I/O pins of the top-level FPGA design.

## Read Capture Timing

In Stratix III and Stratix IV devices, read capture timing may fail if the DQS phase shift selected is not optimal or if the board skew specified is large.

- You can adjust the effective DQS phase shift implemented by the DLL to balance setup and hold margins on the read timing path. The DQS phase shift can be adjusted in coarse PVT-compensated steps of 22.5°, 30°, 36°, or 45° by changing the number of delay buffers (range 1 to 4), or in fine steps using the DQS phase offset feature that supports uncompensated delay addition and subtraction in approximately 12 ps steps.
- To adjust the coarse phase shift selection, determine the supported DLL modes for your chosen memory interface frequency by referencing the DLL and DQS Logic Block Specifications tables in the *Switching Characteristics* section of the device data sheet. For example, a 400 MHz DDR2 interface on a -2 speed grade device can use DLL mode 5 (resolution 36°, range 290 – 450 MHz) to implement a 72° phase shift, or DLL mode 6 (resolution 45°, range 360–560 MHz) to implement a 90° phase shift.

**Note:** Ensure that you specify the appropriate board-skew parameter when you parameterize the controllers in the parameter editor. The default board trace length mismatch used is 20 ps.

## Write Timing

Negative timing margins may be reported for write timing paths if the PLL phase shift used to generate the write data signals is not optimal.

Adjust the PLL phase shift selection on the write clock PLL output using the PLL parameter editor.

**Note:** Regenerating the UniPHY-based controller overwrites changes made using the PLL parameter editor.

## Address and Command Timing

You can optimize the timing margins on the address and command timing path by changing the PLL phase shift used to generate these signals. In the DDR2 or DDR3 SDRAM Controllers with UniPHY IP cores, modify the **Additional CK/CK# phase** and **Additional Address and Command clock phase** parameters.

The DDR2 and DDR3 SDRAM memory controllers use 1T memory timing even in half-rate mode, which may affect the address command margins for DDR2 or DDR3 SDRAM designs that use memory DIMMs. DDR2 SDRAM designs have a greater impact because the address command bus fans out to all the memory devices on a DIMM increasing the loading effect on the bus. Make sure your board designs are robust enough to have the memory clock rising edge within the 1T address command window. You can also use the **Additional Address and Command clock phase** parameter to adjust the phase of the address and command if needed.

The far-end load value and board trace delay differences between address and command and memory clock pins can result in timing failures if they are not accounted for during timing analysis.

The Quartus Prime Fitter may not optimally set output delay chains on the address and command pins. To ensure that any PLL phase-shift adjustments are not negated by delay chain adjustments, create logic assignments using the Assignment Editor to set all address and command output pin D5 delay chains to 0.

For Stratix III and Stratix IV devices, some corner cases of device family and memory frequency may require an increase to the address and command clock phase to meet core timing. You can identify this situation, if the DDR timing report shows a `PHY setup` violation with the `phy_clk` launch clock, and the address and command latch clock—clock 0 (half-rate `phy_clk`) or 2 (full-rate `phy_clk`), and 6, respectively.

If you see this timing violation, you may fix it by advancing the address and command clock phase as required. For example, a 200-ps violation for a 400-MHz interface represents 8% of the clock period, or 28.8. Therefore, advance the address and command phase from 270 to 300. However, this action reduces the setup and hold margin at the DDR device.

## PHY Reset Recovery and Removal

A common cause for reset timing violations in UniPHY designs is the selection of a global or regional clock network for a reset signal.

The UniPHY IP does not require any dedicated clock networks for reset signals. Only UniPHY PLL outputs require clock networks, and any other PHY signal using clock networks may result in timing violations.

You can correct such timing violations by:

- Setting the Global Signal logic assignment to **Off** for the problem path (using the Assignment Editor), or
- Adjusting the logic placement (using the Assignment Editor or Chip Planner)

## Clock-to-Strobe (for DDR and DDR2 SDRAM Only)

Memory output clock signals and DQS strobes are generated using the same PLL output clock. Therefore, no timing optimization is possible for this path and positive timing margins are expected for interfaces running at or below the FPGA data sheet specifications.

For DDR3 interfaces, the timing margin for this path is reported as **Write Leveling**.

## Read Resynchronization and Write Leveling Timing (for SDRAM Only)

These timing paths apply only to Arria II GX, Stratix III, and Stratix IV devices, and are implemented using calibrated clock signals driving dedicated IOE registers. Therefore, no timing optimization is possible for these paths, and positive timing margin is expected for interfaces running at or below the FPGA data sheet specifications.

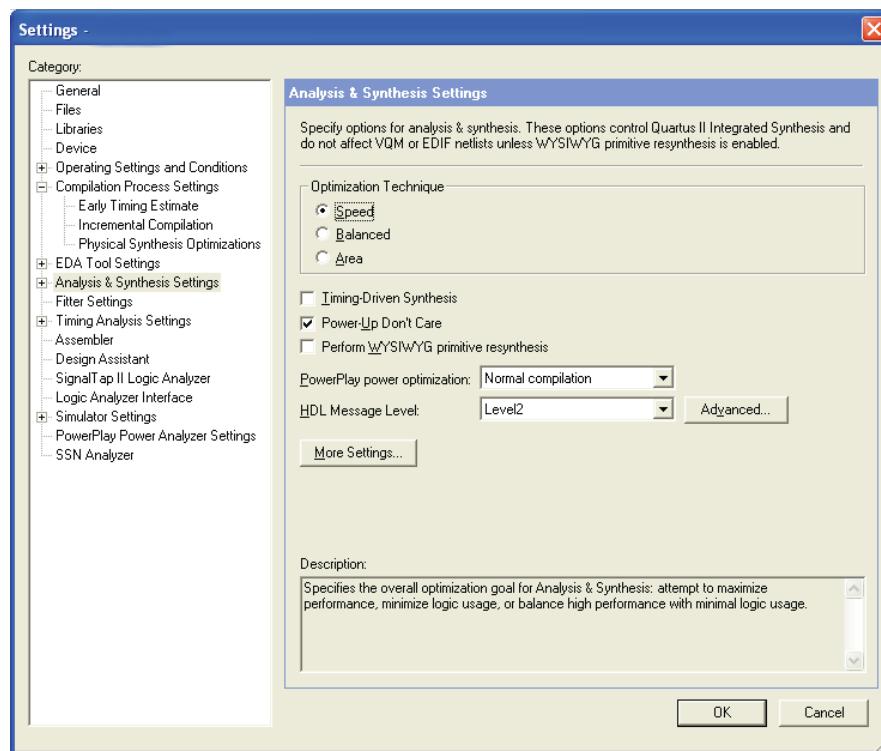
Ensure that you specify the correct memory device timing parameters ( $t_{DQSK}$ ,  $t_{DSS}$ ,  $t_{DSH}$ ) and board skew ( $t_{EXT}$ ) in the DDR2 and DDR3 SDRAM Controllers with UniPHY parameter editor.

## Optimizing Timing

For full-rate designs you may need to use some of the Quartus Prime advanced features, to meet core timing, by following these steps:

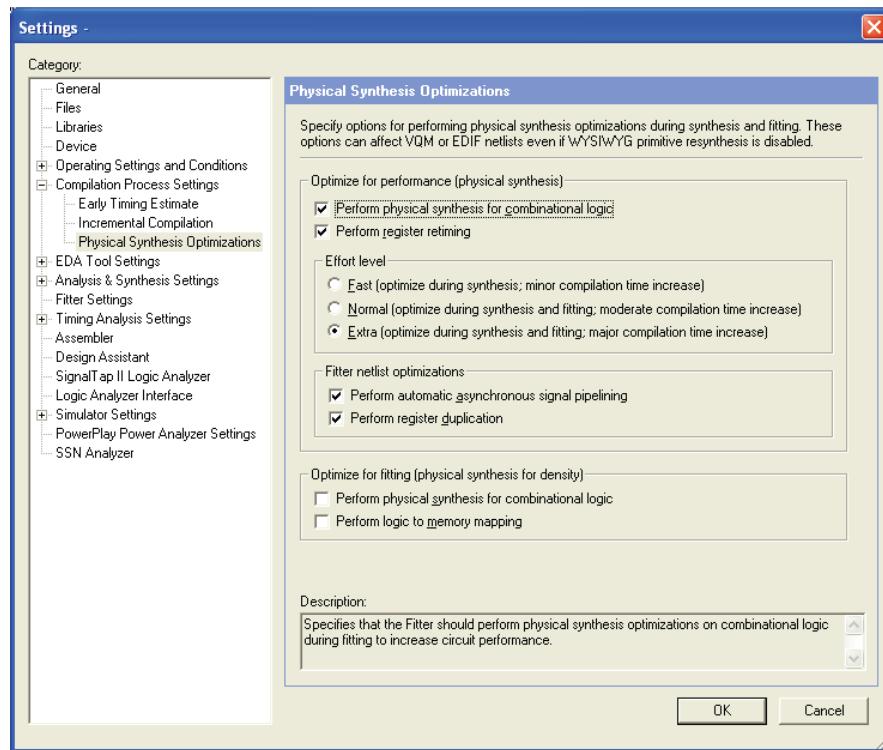
1. On the Assignments menu click **Settings**. In the **Category** list, click **Analysis & Synthesis Settings**. For **Optimization Technique** select **Speed**.

**Figure 9-22: Optimization Technique**



- Turn on **Perform physical synthesis for combinational logic**.  
For more information about physical synthesis, refer to the *Netlist and Optimizations and Physical Synthesis* chapter in the *Quartus Prime Software Handbook*.
- Turn on **Perform register retiming**
- Turn on **Perform automatic asynchronous signal pipelining**
- Turn on **Perform register duplication**
- You can initially select **Normal** for **Effort level**, then if the core timing is still not met, select **Extra**.

**Figure 9-23: Physical Synthesis Optimizations**



#### Related Information

[Netlist Optimizations and Physical Synthesis](#)

## Timing Derivation Methodology for Multiple Chip Select DDR2 and DDR3 SDRAM Designs

In a multiple chip select system, each individual rank has its own chip select signal. Consequently, you must change the **Total Memory chip selects**, **Number of chip select** (for discrete components) or **Number of chip select per slot** (DIMMs) in the **Preset Editor** of the UniPHY-based parameter editors.

In the **Preset Editor**, you must leave the baseline non-derated  $t_{DS}$ ,  $t_{DH}$ ,  $t_{IS}$ ,  $t_{IH}$  values, because the settings on the **Board Settings** page account for multiple chip select slew rate deration.

The following topics explain two timing deration methodologies for multiple chip-select DDR2 and DDR3 SDRAM designs:

- Timing Deration using the Board Settings
- Timing Deration Using the Excel-Based Calculator

For Arria II GX, Arria II GZ, Arria V GZ, Cyclone V, Stratix IV, and Stratix V devices, the UniPHY-based controller parameter editor has an option to select multiple chip-select deration.

**Note:** To perform multiple chip-select timing deration for other Altera devices (such as Stratix III devices), Altera provides an Excel-based calculator available from the Altera website.

Timing deration in this section applies to either discrete components or DIMMs.

**Note:** You can derate DDR SDRAM multiple chip select designs by using the DDR2 SDRAM section of the Excel-based calculator, but Altera does not guarantee the results.

This section assumes you know how to obtain data on PCB simulations for timing deration from HyperLynx or any other PCB simulator.

#### Related Information

- [Timing Deration using the Board Settings](#) on page 9-43
- [Multi Chip Select Calculator](#)

## Multiple Chip Select Configuration Effects

A DIMM contains one or several RAM chips on a small PCB with pins that connect it to another system such as a motherboard or router.

Nonregistered (unbuffered) DIMMs (or UDIMMs) connect address and control buses directly from the module interface to the DRAM on the module.

Registered DIMMs (RDIMMs) and load-reduced DIMMs (LRDIMMs) improve signal integrity (and hence potential clock rates and/or overall memory size) by electrically buffering address and command signals as well as the data bus (for LRDIMMs) at a cost of additional latency. Both RDIMMs and LRDIMMs use parity on the address and command bus for increased robustness.

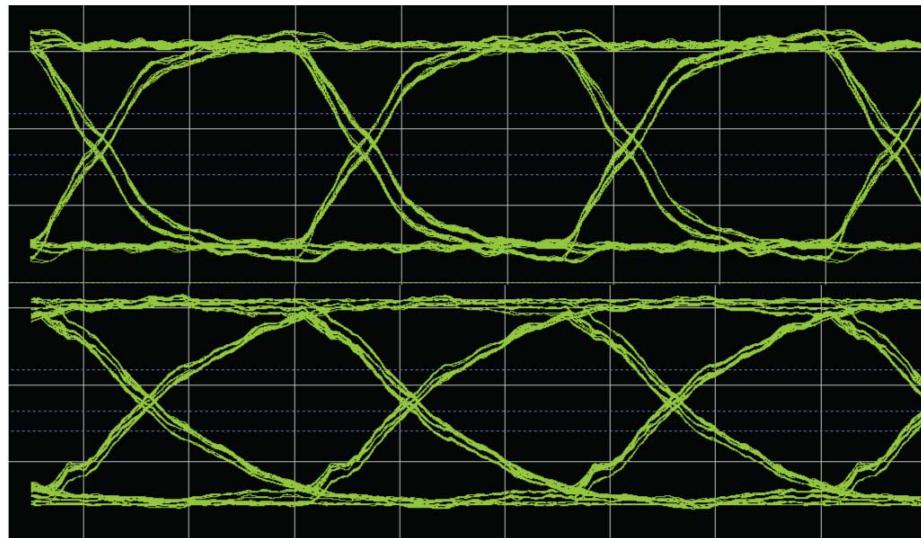
Multiple chip select configurations allow for one set of data pins (and address pins for UDIMMs) to be connected to two or more memory ranks. Multiple chip select configurations have a number of effects on the timing analysis including the intersymbol interference (ISI) effects, board effects, and calibration effects.

### ISI Effects

With multiple chip selects and possible slots loading the far end of the pin, there may be ISI effects on a signal causing the eye openings for DQ, DQS, and address and command signals to be smaller than for single-rank designs.

The following figure shows the eye shrinkage for DQ signal of a single rank system (top) and multiple chip select system (bottom). The ISI eye reductions reduce the timing window available for both the write path and the address and command path analysis. You must specify them as output delay constraints in the **.sdc**.

Extra loading from the additional ranks causes the slew rate of signals from the FPGA to be reduced. This reduction in slew rate affects some of the memory parameters including data, address, command and control setup and hold times ( $t_{DS}$ ,  $t_{DH}$ ,  $t_{IS}$ , and  $t_{IH}$ ).

**Figure 9-24: Eye Shrinkage for DQ Signal**

## Calibration Effects

In addition to SI effects, multiple chip select topologies change the way that the FPGA calibrates to the memories.

In single rank without leveling situations, the calibration algorithm centers the resynchronization or capture phase such that it is optimum for the single rank. When there are two or more ranks in a system, the calibration algorithms must calibrate to the average point of the ranks.

## Board Effects

Unequal length PCB traces result in delays reducing timing margins. Skews between different memory ranks can further reduce the timing margins in multiple chip select topologies.

Board skews can also affect the extent to which the FPGA can calibrate to the different ranks. If the skew between various signals for different ranks is large enough, the timing margin on the fully calibrated paths such as write leveling and resynchronization changes.

To account for all these board effects for Arria II GX, Arria II GZ, Arria V, Cyclone V, Stratix IV, and Stratix V devices, refer to the **Board Settings** page in the UniPHY-based controller parameter editors.

**Note:** To perform multiple chip select timing deration for other Altera devices (for example, Stratix III devices), use the Excel-based calculator available from the Altera website.

### Related Information

[Multi Chip Select Calculator](#)

## Timing Deration using the Board Settings

When you target Arria II GX, Arria II GZ, Arria V, Cyclone V, Stratix IV, or Stratix V devices, the UniPHY-based parameter editors include the **Board Settings** page, to automatically account for the timing deration caused by the multiple chip selects in your design.

When you target Stratix III devices, you can derate single chip-select designs using the parameter editors to account for the skews, ISI, and slew rates in the **Board Settings** page.

If you are targeting Stratix III devices you see the following warning:

"Warning: Calibration performed on all chip selects, timing analysis only performed on first chip select.  
Manual timing derating is required"

**Note:** You must perform manual timing deration using the Excel-based calculator.

The **Board Settings** page allows you to enter the parameters related to the board design including skews, signal integrity, and slew rates. The **Board Settings** page also includes the board skew setting parameter, **Addr/Command to CK skew**.

## Slew Rates

You can obtain the slew rates in one of the following ways:

- Altera performs PCB simulations on internal Altera boards to compute the output slew rate and ISI effects of various multiple chip select configurations. These simulation numbers are prepopulated in the **Slew Rates** based on the number of ranks selected. The address and command setup and hold times ( $t_{DS}$ ,  $t_{DH}$ ,  $t_{IS}$ ,  $t_{IH}$ ) are then computed from the slew rate values and the baseline nonderated  $t_{DS}$ ,  $t_{DH}$ ,  $t_{IS}$ ,  $t_{IH}$  numbers entered in the **Preset Editor**. The parameter editor shows the computed values in **Slew Rates**. If you do not have access to a simulator to obtain accurate slew rates for your specific system, Altera recommends that you use these prepopulated numbers for an approximate estimate of your actual board parameters.
- Alternatively, you can update these default values, if dedicated board simulation results are available for the slew rates. Custom slew rates cause the  $t_{DS}$ ,  $t_{DH}$ ,  $t_{IS}$ ,  $t_{IH}$  values to be updated. Altera recommends performing board level simulations to calculate the slew rate numbers that accounts for accurate board-level effects for your board.
- You can modify the auto-calculated  $t_{DS}$ ,  $t_{DH}$ ,  $t_{IS}$ ,  $t_{IH}$  values with more accurate dedicated results direct from the vendor data sheets, if available.

## Slew Rate Setup, Hold, and Derating Calculation

Slew rate is calculated based on the nominal slew rate for setup and hold times. The total  $t_{IS}$  (setup time) and  $t_{IH}$  (hold time) required is calculated by adding the Micron data sheet  $t_{IS}$  (base) and  $t_{IH}$  (base) values to the delta  $t_{IS}$  and delta  $t_{IH}$  derating values, respectively.

For more information about slew rate calculation, setup, hold, and derating values, download the data sheet specifications from the following vendor websites:

- Micron ([www.micron.com](http://www.micron.com)) For example, refer to *Command and Address Setup, Hold, and Derating* section in the Micron DDR3 data sheet.
- JEDEC ([www.jedec.org](http://www.jedec.org)) For example, refer to the DDR2 SDRAM Standard data sheet.

The following section describes the timing derating algorithms and shows you where to obtain the setup, hold, and derating values in the Micron data sheet.

The slew rate derating process uses the following timing derating algorithms, which is similar to the JEDEC specification:

$$t_{DS} = t_{DS}(\text{base}) + \Delta t_{DS} + (V_{IHAC} - V_{REF}) / (\text{DQ slew rate})$$

$$t_{DH} = t_{DH}(\text{base}) + \Delta t_{DH} + (V_{IHDC} - V_{REF}) / (\text{DQ slew rate})$$

$$t_{IS} = t_{IS}(\text{base}) + \Delta t_{IS} + (V_{IHAC} - V_{REF}) / (\text{Address/Command slew rate})$$

$$t_{IH} = t_{IH}(\text{base}) + \Delta t_{IH} + (V_{IHDC} - V_{REF}) / (\text{Address/Command slew rate})$$

where:

- The setup and hold values for tDS(base), tDH(base), tIS(base), and tIH(base) are obtained from the Micron data sheet. The following figure shows an example of the values from the Micron data sheet.

**Figure 9-25: Setup and Hold Values from Micron Data sheet**

Parameter	Symbol	DDR3-800		DDR3-1066		DDR3-1333		DDR3-1600		Units	Notes
		Min	Max	Min	Max	Min	Max	Min	Max		
<b>DQ Input Timing</b>											
Data setup time to DQS, DQS#	<sup>1</sup> tDS <sub>V<sub>REF</sub> @ 1 V/ns</sub>	75	-	25	=	-	-	-	-	ps	18, 19
Data setup time to DQS, DQS#	<sup>1</sup> tDS <sub>V<sub>REF</sub> @ 1 V/ns</sub>	125	-	200	-	-	-	-	-	ps	19, 20
Data setup time to DQS, DQS#	<sup>1</sup> tDS <sub>V<sub>REF</sub> @ 1 V/ns</sub>	AC150	2/5	-	250	-	180	-	160	-	ps 18, 19, 20
Data setup time to DQS, DQS#	<sup>1</sup> tDS <sub>V<sub>REF</sub> @ 1 V/ns</sub>	AC135	-	-	-	-	-	-	-	ps	18, 19, 20
Data hold time from DQS, DQS#	<sup>1</sup> tDH <sub>V<sub>REF</sub> @ 1 V/ns</sub>	150	-	100	-	65	-	45	-	ps	18, 19
Data hold time from DQS, DQS#	<sup>1</sup> tDH <sub>V<sub>REF</sub> @ 1 V/ns</sub>	DC100	250	-	200	-	165	-	145	-	ps 19, 20
Minimum data pulse width	<sup>1</sup> tIPW	600	-	490	-	400	-	360	-	ps	41
<b>Command and Address Timing</b>											
DLL locking time	<sup>1</sup> tDLLK	512	-	512	-	512	-	512	-	CK	28
CTRL, CMD, ADDR setup to CK, CK#	<sup>1</sup> tS <sub>V<sub>REF</sub> @ 1 V/ns</sub>	200	-	125	-	65	-	45	-	ps	29, 30
CTRL, CMD, ADDR setup to CK, CK#	<sup>1</sup> tS <sub>V<sub>REF</sub> @ 1 V/ns</sub>	AC175	375	-	300	-	240	-	220	-	ps 20, 30
CTRL, CMD, ADDR setup to CK, CK#	<sup>1</sup> tS <sub>V<sub>REF</sub> @ 1 V/ns</sub>	AC150	350	-	275	-	190	-	170	-	ps 29, 30
CTRL, CMD, ADDR hold time from CK, CK#	<sup>1</sup> tH <sub>V<sub>REF</sub> @ 1 V/ns</sub>	275	-	200	-	140	-	120	-	ps	29, 30
Minimum CTRL, CMD, ADDR pulse width	<sup>1</sup> tPW	DC100	375	-	300	-	240	-	220	-	ps 20, 30
Minimum CTRL, CMD, ADDR pulse width	<sup>1</sup> tPW	900	-	780	-	620	-	560	-	ps	41

- The JEDEC defined logic trip points for DDR3 SDRAM memory standard are as follow:
  - $V_{IHAC} = V_{REF} + 0.175\text{ V}$
  - $V_{IHDC} = V_{REF} + 0.1\text{ V}$
  - $V_{ILAC} = V_{REF} - 0.175\text{ V}$
  - $V_{ILDC} = V_{REF} - 0.1\text{ V}$
- The derating values for delta tIS, tIH, tDH, and tDS are obtained from the Micron data sheet.

The following figure shows an image of the derating values from the Micron data sheet.

**Figure 9-26: Derating Values from Micron Data sheet**

$\Delta tIS, \Delta tIH$ Derating (ps) – AC/DC-Based																
CMD/ ADDR Slew Rate V/ns		CK, CK# Differential Slew Rate														
		4.0 V/ns		3.0 V/ns		2.0 V/ns		1.8 V/ns		1.6 V/ns		1.4 V/ns		1.2 V/ns		1.0 V/ns
$\Delta tIS$	$\Delta tIH$	$\Delta tIS$	$\Delta tIH$	$\Delta tIS$	$\Delta tIH$	$\Delta tIS$	$\Delta tIH$	$\Delta tIS$	$\Delta tIH$	$\Delta tIS$	$\Delta tIH$	$\Delta tIS$	$\Delta tIH$	$\Delta tIS$	$\Delta tIH$	
2.0	88	50	88	50	88	50	96	58	104	66	112	74	120	84	128	100
1.5	59	34	59	34	59	34	67	42	75	50	83	58	91	68	99	84
1.0	0	0	0	0	0	0	8	8	16	16	24	24	32	34	40	50
0.9	-2	-4	-2	-4	-2	-4	6	4	14	12	22	20	30	30	38	46
0.8	-6	-10	-6	-10	-6	-10	2	-2	10	6	18	14	26	24	34	40
0.7	-11	-16	-11	-16	-11	-16	-3	-8	5	0	13	8	21	18	29	34
0.6	-17	-26	-17	-26	-17	-26	-9	-18	-1	-10	7	-2	15	8	23	24
0.5	-35	-40	-35	-40	-35	-40	-27	-32	-19	-24	-11	-16	-2	-6	5	10
0.4	-62	-60	-62	-60	-62	-60	-54	-52	-46	-44	-38	-36	-30	-26	-22	-10

Shaded cells indicate slew rate combinations not supported													
$\Delta tDS, \Delta tDH$ Derating (ps) – AC/DC-Based													
DQ Slew Rate V/ns		DQS, DQS# Differential Slew Rate											
		4.0 V/ns	3.0 V/ns	2.0 V/ns	1.8 V/ns	1.6 V/ns	1.4 V/ns	1.2 V/ns	1.0 V/ns	$\Delta tDS$	$\Delta tDH$	$\Delta tDS$	$\Delta tDH$
2.0	88	50	88	50	88	50							
1.5	59	34	59	34	59	34	67	42					
1.0	0	0	0	0	0	0	8	8	16	16			
0.9			-2	-4	-2	-4	6	4	14	12	22	20	
0.8					-6	-10	2	-2	10	6	18	14	26
0.7							-3	-8	5	0	13	8	21
0.6									1	10	7	3	15

#### Related Information

- [www.micron.com](http://www.micron.com)
- [Micron DDR3 Data Sheet](#)
- [www.jedec.org](http://www.jedec.org)

### Intersymbol Interference

ISI parameters are similarly auto-populated based on the number of ranks you select with Altera's PCB simulations. You can update these autopopulated typical values, if more accurate dedicated simulation results are available.

Altera recommends performing board-level simulations to calculate the slew rate and ISI deltas that account for accurate board level effects for your board. You can use HyperLynx or similar simulators to obtain these simulation numbers. The default values have been computed using HyperLynx simulations for Altera boards with multiple DDR2 and DDR3 SDRAM slots.

The wizard writes these parameters for the slew rates and the ISI into the .sdc and they are used during timing analysis.

### Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time

This topic describes how to measure eye reduction for address/command, DQ, and DQS.

Channel signal integrity is a measure of the distortion of the eye due to intersymbol interference or crosstalk or other effects. Typically, when going from a single-rank configuration to a multi-rank configuration there is an increase in the channel loss due to reflections caused by multiple stubs. Although the Quartus Prime timing model includes some channel uncertainty, you must perform your own channel signal integrity simulations and enter the additional channel uncertainty, as compared to the reference eye, into the parameter editor.

For details about measuring channel signal integrity, refer to *Measuring Channel Signal Integrity*, on the Altera Wiki.

#### Related Information

- [http://alterawiki.com/wiki/Measuring\\_Channel\\_Signal\\_Integrity](http://alterawiki.com/wiki/Measuring_Channel_Signal_Integrity)

### Early I/O Timing Estimation for Arria 10 EMIF IP

Early I/O timing analysis allows you to run I/O timing analysis without first compiling your design. You can use early I/O timing analysis to quickly evaluate whether adequate timing margin exists on the I/O interface between the FPGA and external memory device.

Early I/O timing analysis performs the following analyses:

- Read analysis
- Write analysis
- Address and command analysis
- DQS gating analysis
- Write leveling analysis

Early I/O timing analysis takes into consideration the following factors:

- The timing parameters of the memory device
- The speed and topology of the memory interface
- The board timing and ISI characteristics
- The timing of the selected FPGA device

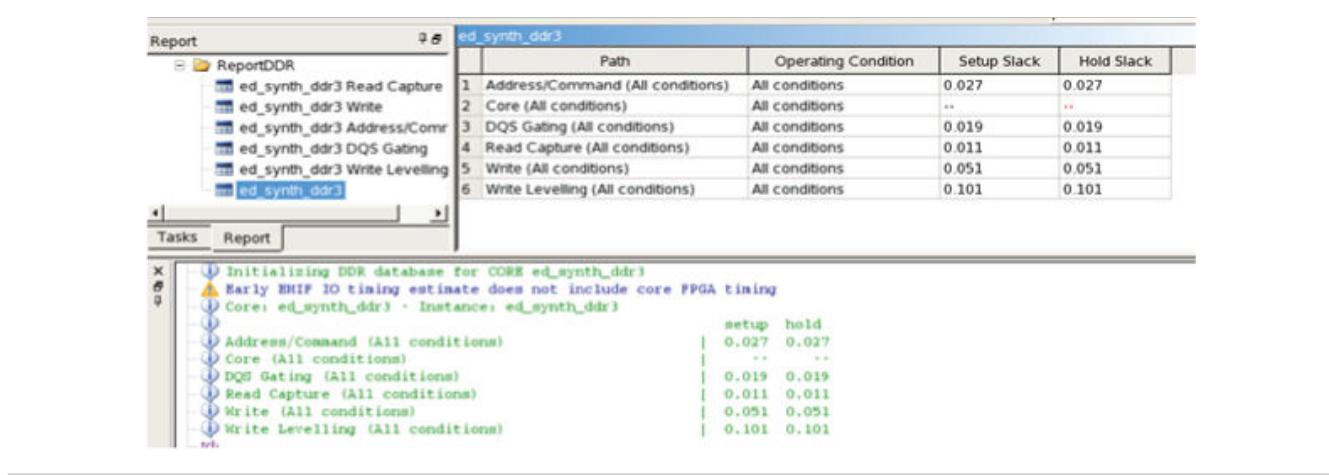
## Performing Early I/O Timing Analysis for Arria 10 EMIF IP

To perform early I/O timing analysis, follow these steps:

1. Instantiate an Arria 10 EMIF IP core.
  - a. On the **Memory Timing** tab, enter accurate memory parameters.
  - b. On the **Board Timing** tab, enter accurate values for Slew Rate, Intersymbol Interference, and Board and Package Skews.
2. After generating your IP core, create a Quartus Prime project and select your device from the **Available devices** list.
3. To launch the TimeQuest Timing Analyzer, select **TimeQuest Timing Analyzer** from the **Tools** menu.
4. To run early I/O timing analysis:
  - a. Select **Run Tcl Script** from the **Script** menu.
  - b. Run **submodule\<variation\_name>\_report\_io\_timing.tcl**.

The following figure shows an early I/O timing analysis from the TimeQuest Timing Analyzer using a DDR3 example design.

**Figure 9-27: Report DDR Timing Results**



Report DDR details the read capture, write, address and command, DQS gating, and write leveling timing analyses, which are identical to those obtained after a full design compilation. Core FPGA timing paths are not included in early I/O timing analysis.

## Performing I/O Timing Analysis

For accurate I/O timing analysis, the Quartus Prime software must be made aware of the board trace and loading information. This information must be derived and refined during your PCB development process of pre-layout (line) and post-layout (board) simulations.

For external memory interfaces that use memory modules (DIMMs), the board trace and loading information must include the trace and loading information of the module in addition to the main and host platform, which you can obtain from your memory vendor.

You can use the following I/O timing analysis methods for your memory interface:

- Perform I/O Timing Analysis with 3rd Party Simulation Tools
- Perform Advanced I/O Timing Analysis with Board Trace Delay Model

### Related Information

- [Performing I/O Timing Analysis with 3rd Party Simulation Tools](#) on page 9-48
- [Performing Advanced I/O Timing Analysis with Board Trace Delay Model](#) on page 9-48

## Performing I/O Timing Analysis with 3rd Party Simulation Tools

Altera recommends that you perform I/O timing analysis using the 3<sup>rd</sup> party simulation tool flow because this flow involves post layout simulation that can capture more accurate I/O timing. This method is also easier because it only requires you to enter the slew rates, board skews, and ISI values in the parameter editor.

To perform I/O timing analysis using 3<sup>rd</sup> party simulation tools, follow these steps:

1. Use a 3<sup>rd</sup> party simulation tool such as HyperLynx to simulate the full path for DQ, DQS, CK, Address, and Command signals.
2. Under the **Board Settings** tab of the parameter editor, enter the slowest slew rate, ISI, and board skew values.

## Performing Advanced I/O Timing Analysis with Board Trace Delay Model

You should use this method only if you are unable to perform post-layout simulation on the memory interface signals to obtain the slew rate parameters, and/or when no simulation tool is available.

To perform I/O timing analysis using board trace delay model, follow these steps:

1. After the instantiation is complete, analyze and synthesize your design.
2. Add pin and DQ group assignment by running the <variation\_name>**\_p0\_pin\_assignments.tcl** script.

**Note:** The naming of the pin assignment file may vary depending on the Quartus Prime software version that you are using.

3. Enter the pin location assignments.
4. Assign the virtual pins, if necessary.
5. Enter the board trace model information. To enter board trace model information, follow these steps:

- a. In the Pin Planner, select the pin or group of pins for which you want to enter board trace parameters.
  - b. Right-click and select **Board Trace Model**.
6. Compile your design. To compile the design, on the Processing menu, click **Start Compilation**.
  7. After successfully compiling the design, perform timing analysis in the TimeQuest timing analyzer. To perform timing analysis, follow these steps:
    - a. In the Quartus Prime software, on the Tools menu, click **TimeQuest Timing Analyzer**.
    - b. On the **Tasks** pane, click **Report DDR**.
    - c. On the **Report** pane, select **Advanced I/O Timing>Signal Integrity Metrics**.
    - d. In the **Signal Integrity Metrics** window, right-click and select **Regenerate** to regenerate the signal integrity metrics.
    - e. In the **Signal Integrity Metrics** window, note the 10–90% rise time (or fall time if fall time is worse) at the far end for CK/CK#, address, and command, DQS/DQS#, and DQ signals.
    - f. In the DDR3 SDRAM controller parameter editor, in the **Board Settings** tab, type the values you obtained from the signal integrity metrics.
    - g. For the board skew parameters, set the maximum skew within DQS groups of your design. Set the other board parameters to 0 ns.
    - h. Compile your design.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Removed occurrences of MegaWizard Plug-In Manager.
December 2013	2013.12.16	<ul style="list-style-type: none"><li>• Removed references to ALTMEMPHY.</li><li>• Removed references to HardCopy.</li><li>• Removed references to Stratix II devices.</li><li>• Removed references to Cyclone III and Cyclone IV devices.</li><li>• Added information for Arria 10 support.</li></ul>
November 2012	6.0	Changed chapter number from 10 to 11.
June 2012	5.0	Added Feedback icon.
November 2011	4.0	<ul style="list-style-type: none"><li>• Added Arria V and Cyclone V information.</li><li>• Added <i>Performing I/O Timing Analysis</i> section.</li><li>• Added <i>Measuring Eye Reduction for Address/Command, DQ, and DQS Setup and Hold Time</i> section.</li></ul>

Date	Version	Changes
June 2011	3.0	Updated for 11.0 release.
December 2010	2.1	Added Arria II GZ and Stratix V, updated board skews table.
July 2010	2.0	Added information about UniPHY-based IP and controllers.
January 2010	1.2	Corrected typos.
December 2009	1.1	Added <i>Timing Derivation</i> section.
November 2009	1.0	Initial release.



# Debugging Memory IP

# 10

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The following topics describe the tools and processes for debugging external memory interfaces.

The discussion focuses on issues pertaining to the Altera<sup>®</sup> DDR, DDR2, DDR3, DDR4, LPDDR2, LPDDR3, QDRII, QDRII+, QDRII+ Xtreme, QDR-IV, RLDRAM II, and RLDRAM 3 IP.

In general, memory debugging issues can be categorized as follows:

- Resource and planning issues
- Interface configuration issues
- Functional issues
- Timing issues

Some issues may not be directly related to interface operation; problems can also occur at the Quartus<sup>®</sup> Prime Fitter stage, or in timing analysis.

## Resource and Planning Issues

Typically, single stand-alone interfaces should not present Quartus Prime Fitter or timing problems.

You may find that fitter, timing, and hardware operation can sometimes become a challenge, as multiple interfaces are combined into a single project, or as the device utilization increases. In such cases, interface configuration is not the issue; rather, the placement and total device resource requirements can create problems.

### Resource Issue Evaluation

External memory interfaces typically require the following resource types, which you must consider when trying to place logic manually. You might also use additional constraints to force the placement or location of external memory interface IP:

- Dedicated IOE DQS group resources and pins
- Dedicated DLL resources
- Specific PLL resources
- Specific global, regional, and dual-regional clock net resources

## Dedicated IOE DQS Group Resources and Pins

Fitter issues can occur with even a single interface, if you do not size the interface to fit within the specified constraints and requirements. A typical requirement includes containing assignments for the interface within a single bank or possibly side of the chosen device.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

Such a constraint requires that the chosen device meets the following conditions:

- Sufficient DQS groups and sizes to support the required number of common I/O (CIO) or separate I/O (SIO) data groups.
- Sufficient remaining pins to support the required number of address, command, and control pins.

Failure to evaluate these fundamental requirements can result in suboptimal interface design, if the chosen device cannot be modified. The resulting wraparound interfaces or suboptimal pseudo read and write data groups artificially limit the maximum operating frequency.

Multiple blocks of IP further complicate the issue, if other IP has either no specified location constraints or incompatible location constraints.

The Quartus Prime fitter may first place other components in a location required by your memory IP, then error at a later stage because of an I/O assignment conflict between the unconstrained IP and the constrained memory IP.

Your design may require that one instance of IP is placed anywhere on one side of the device, and that another instance of IP is placed at a specific location on the same side.

While the two individual instances may compile in isolation, and the physical number of pins may appear sufficient for both instances, issues can occur if the instance without placement constraints is placed before the instance with placement constraints.

In such circumstances, Altera recommends manually placing each individual pin, or at least try using more granular placement constraints.

For more information about the pin number and DQS group capabilities of your chosen device, refer to device data sheets or the Quartus Prime Pin Planner.

## Dedicated DLL Resources

Altera devices typically use DLLs to enhance data capture at the FPGA. While multiple external memory interfaces can usually share DLL resources, fitter issues can occur when there is insufficient planning before HDL coding.

If DLL sharing is required, Altera gives the following recommendations for each instance of the IP that shares the DLL resources:

- Must have compatible DLL requirements—same frequency and mode.
- Exports its autogenerated DLL instance out of its own dedicated PHY hierarchy and into the top-level design file. This procedure allows easy comparison of the generated DLL's mode, and allows you to explicitly show the required DLL sharing between two IP blocks in the HDL

**Note:** The Quartus Prime fitter does not dynamically merge DLL instances.

## Specific PLL Resources

When only a single interface resides on one side or one quadrant of a device, PLL resources are typically not an issue. However if multiple interfaces or IP are required on a single side or quadrant, consider the specific PLL used by each IP, and the sharing of any PLL resources.

The Quartus Prime software automerges PLL resources, but not for any dynamically controlled PLL components. Use the following PLL resource rules:

- Ensure that the PLL located in the same bank or side of the device is available for your memory controller.
- If multiple PLLs are required for multiple controllers that cannot be shared, ensure that enough PLL resources are available within each quadrant to support your interface number requirements.
- Try to limit multiple interfaces to a single quadrant. For example, if two complete same size interfaces can fit on a single side of the device, constrain one interface entirely in one bank of that side, and the other controller in the other bank.

For more information about using multiple PHYs or controllers, refer to the design tutorials on the *List of designs using Altera External Memory IP* page of the Altera Wiki website.

#### Related Information

[List of designs using Altera External Memory IP](#)

## Specific Global, Regional and Dual-Regional Clock Net Resources

Memory PHYs typically have specific clock resource requirements for each PLL clock output.

For example because of characterization data, the PHY may require that the `phy_clk` is routed on a global clock net. The remaining clocks may all be routed on a global or a regional clock net. However, they must all be routed on the same type. Otherwise, the operating frequency of the interface is lowered, because of the increased uncertainty between two different types of clock nets. The design may still fit, but not meet timing.

## Planning Your Design

It is important to understand your design and to plan its resource usage and layout. Include the following steps in your design planning:

1. Plan the total number of DQS groups and total number of other pins required in your shared area. Use the Pin Planner to assist with this activity.
2. Decide which PLLs or clock networks can be shared between IP blocks, then ensure that sufficient resources are available. For example, if an IP core requires a regional clock network, a PLL located on the opposite side of the device cannot be used.
3. Calculate the number of total clock networks and types required when trying to combine multiple instances of IP.
4. You must understand the number of quadrants that the IP uses and if this number can be reduced. For example, an interface may be autoplated across an entire side of the device, but may actually be constrained to fit in a single bank.

By optimizing physical placement, you ensure that the system uses regional clock networks wherever possible. The use of regional clock networks instead of dual-regional clock networks can help maintain clock net resources and simplify routing.

## Optimizing Design Utilization

As device utilization increases, the Quartus Prime software may have difficulty placing the core. To optimize design utilization, follow these steps:

1. Review any fitter warning messages in multiple IP designs to ensure that clock networks or PLL modes are not modified to achieve the desired fit.
2. Use the Quartus Prime Fitter resource section to compare the types of resources used in a successful standalone IP implementation to those used in an unreliable multiple IP implementation.
3. Use this information to better constrain the project to achieve the same results as the standalone project.
4. Use the Chip Planner (Floorplan and Chip Editor) to compare the placement of the working stand-alone design to the multiple IP project. Then use LogicLock™ or Design Partitions to better guide the Quartus Prime software to the required results.
5. When creating LogicLock regions, ensure that they encompass all required resources. For example, if constraining the read and write datapath hierarchy, ensure that your LogicLock region includes the IOE blocks used for your datapath pin out.

## Interface Configuration Performance Issues

There are a large number of interface combinations and configurations possible in an Altera design, therefore it is impractical for Altera to explicitly state the achievable  $f_{MAX}$  for every combination.

Altera seeks to provide guidance on typical performance, but this data is subject to memory component timing characteristics, interface widths, depths directly affecting timing deration requirements, and the achieved skew and timing numbers for a specific PCB.

FPGA timing issues should generally not be affected by interface loading or layout characteristics. In general, the Altera performance figures for any given device family and speed-grade combination should usually be achievable.

To resolve FPGA (PHY and PHY reset) timing issues, refer to the *Analyzing Timing of Memory IP* chapter.

Achievable interface timing (address and command, half-rate address and command, read and write capture) is directly affected by any layout issues (skew), loading issues (deration), signal integrity issues (crosstalk timing deration), and component speed grades (memory timing size and tolerance). Altera performance figures are typically stated for the default (single rank, unbuffered DIMM) case. Altera provides additional expected performance data where possible, but the  $f_{MAX}$  is not achievable in all configurations. Altera recommends that you optimize the following items whenever interface timing issues occur:

- Improve PCB layout tolerances
- Use a faster speed grade of memory component
- Ensure that the interface is fully and correctly terminated
- Reduce the loading (reduce the deration factor)

### Related Information

[Analyzing Timing of Memory IP](#) on page 9-1

## Interface Configuration Bottleneck and Efficiency Issues

Depending on the transaction types, efficiency issues can exist where the achieved data rate is lower than expected. Ideally, these issues should be assessed and resolved during the simulation stage because they are sometimes impossible to solve later without rearchitecting the product.

Any interface has a maximum theoretical data rate derived from the clock frequency, however, in practise this theoretical data rate can never be achieved continuously due to protocol overhead and bus turnaround times.

Simulate your desired configuration to ensure that you have specified a suitable external memory family and that your chosen controller configuration can achieve your required bandwidth.

Efficiency can be assessed in several different ways, and the primary requirement is an achievable continuous data rate. The local interface signals combined with the memory interface signals and a command decode trace should provide adequate visibility of the operation of the IP to understand whether your required data rate is sufficient and the cause of the efficiency issue.

To show if under ideal conditions the required data rate is possible in the chosen technology, follow these steps:

1. Use the memory vendors own testbench and your own transaction engine.
2. Use either your own driver, or modify the provided example driver, to replicate the transaction types typical of your system.
3. Simulate this performance using your chosen memory controller and decide if the achieved performance is still acceptable.

Observe the following points that may cause efficiency or bottleneck issues at this stage:

- Identify the memory controller rate (full, half, or quarter) and commands, which may take two or four times longer than necessary
- Determine whether the memory controller is starved for data by observing the appropriate request signals.
- Determine whether the memory controller processor transactions at a rate sufficient to meet throughput requirements by observing appropriate signals, including the local ready signal.

Altera has several versions and types of memory controller, and where possible you can evaluate different configurations based on the results of the first tests.

Consider using either a faster interface, or a different memory type to better align your data rate requirements to the IP available directly from Altera.

Altera also provides stand-alone PHY configurations so that you may develop custom controllers or use third-party controllers designed specifically for your requirements.

## Functional Issue Evaluation

Functional issues occur at all frequencies (using the same conditions) and are not altered by speed grade, temperature, or PCB changes. You should use functional simulation to evaluate functional issues.

The Altera IP includes the option to autogenerate a testbench specific to your IP configuration, which provides an easy route to functional verification.

The following issues should be considered when trying to debug functional issues in a simulation environment.

## Correct Combination of the Quartus Prime Software and ModelSim-Altera Device Models

When running any simulations, ensure that you are using the correct combination of the Quartus Prime software and device models.

Altera only tests each release of software and IP with the aligned release of device models. Failure to use the correct RTL and model combination may result in unstable simulation environments.

The ModelSim® -Altera edition of the ModelSim simulator comes precompiled with the Altera device family libraries included. You must always install the correct release of ModelSim-Altera to align with your Quartus Prime software and IP release.

If you are using a full version of ModelSim-SE or PE, or any other supported simulation environment, ensure that you are compiling the current Quartus Prime supplied libraries. These libraries are located in the <Quartus Prime install path>/quartus/eda/sim\_lib/ directory.

## Altera IP Memory Model

Altera memory IP autogenerated a generic simplified memory model that works in all cases. This simple read and write model is not designed or intended to verify all entered IP parameters or transaction requirements.

The Altera-generated memory model may be suitable to evaluate some limited functional issues, but it does not provide comprehensive functional simulation.

## Vendor Memory Model

Contact the memory vendor directly, because many additional models are available from the vendor's support system.

When using memory vendor models, ensure that the model is correctly defined for the following characteristics:

- Speed grade
- Organization
- Memory allocation
- Maximum memory usage
- Number of ranks on a DIMM
- Buffering on the DIMM
- ECC

**Note:** Refer to the **readme.txt** file supplied with the memory vendor model, for more information about how to define this information for your configuration. Also refer to Transcript Window Messages, for more information.

**Note:** Altera does not provide support for vendor-specific memory models.

During simulation vendor models output a wealth of information regarding any device violations that may occur because of incorrectly parameterized IP.

## Insufficient Memory in Your PC

If you are running the ModelSim-Altera simulator, the limitation on memory size may mean that you have insufficient memory to run your simulation. Or, if you are using a 32-bit operating system, your PC may have insufficient memory.

Typical simulation tool errors include: "Iteration limit reached" or "Error out of memory".

When using either the Altera generic memory model, or a vendor specific model quite large memory depths can be required if you do not specify your simulation carefully.

For example, if you simulate an entire 4-GB DIMM interface, the hardware platform that performs that simulation requires at least this amount of memory just for the simulation contents storage.

**Note:** Refer to Memory Allocation and Max Memory Usage in the vendor's **readme.txt** files for more information.

## Transcript Window Messages

When you are debugging a functional issue in simulation, vendor models typically provide much more detailed checks and feedback regarding the interface and their operational requirements than the Altera generic model.

In general, you should use a vendor-supplied model whenever one is available. Consider using second-source vendor models in preference to the Altera generic model.

Many issues can be traced to incorrectly configured IP for the specified memory components. Component data sheets usually contain settings information for several different speed grades of memory. Be aware data sheet specify parameters in fixed units of time, frequencies, or clock cycles.

The Altera generic memory model always matches the parameters specified in the IP, as it is generated using the same engine. Because vendor models are independent of the IP generation process, they offer a more robust IP parameterization check.

During simulation, review the transcript window messages and do not rely on the Simulation Passed message at the end of simulation. This message only indicates that the example driver successfully wrote and then read the correct data for a single test cycle.

Even if the interface functionally passes in simulation, the vendor model may report operational violations in the transcript window. These reported violations often specifically explain why an interface appears to pass in simulation, but fails in hardware.

Vendor models typically perform checks to ensure that the following types of parameters are correct:

- Burst length
- Burst order
- tMRD
- tMOD
- tRFC
- tREFPDEN
- tRP
- tRAS
- tRC
- tACTPDEN
- tWR
- tWRPDEN
- tRTP
- tRDPDEN
- tINIT
- tXPDLL
- tCKE
- tRRD
- tCCD
- tWTR
- tXPR
- PRECHARGE
- CAS length
- Drive strength
- AL
- tDQS
- CAS\_WL

- Refresh
- Initialization
- tIH
- tIS
- tDH
- tDS

If a vendor model can verify all these parameters are compatible with your chosen component values and transactions, it provides a specific insight into hardware interface failures.

## Passing Simulation

Passing simulation means that the interface calibrates and successfully completes a single test complete cycle without asserting pass not fail (pnf).

It does not take into account any warning messages that you may receive during simulation. If you are debugging an interface issue, review and, if necessary, correct any warning messages from the transcript window before continuing.

## Modifying the Example Driver to Replicate the Failure

Often during debugging, you may discover that the example driver design works successfully, but that your custom logic is observing data errors.

This information indicates that the issue is either one of the following:

- Related to the way that the local interface transactions are occurring. Altera recommends you probe and compare using the SignalTap™ II analyzer.
- Related to the types or format of transactions on the external memory interface. Altera recommends you modify the example design to replicate the issue.

Typical issues on the local interface side include:

- Incorrect local address to memory address translation causing the word order to be different than expected. Refer to *Burst Definition* in your memory vendor data sheet.
- Incorrect timing on the local interface. When your design requests a transaction, the local side must be ready to service that transaction as soon as it is accepted without any pause.
- For more information, refer to the *Avalon® Interface Specification*.

The default example driver only performs a limited set of transaction types, consequently potential bus contention or preamble and postamble issues can often be masked in its default operation. For successful debugging, isolate the custom logic transaction types that are causing the read and write failures and modify the example driver to demonstrate the same issue. Then, you can try to replicate the failure in RTL simulation with the modified driver.

For Arria 10 interfaces, you can enable the Traffic Generator 2.0 in the example design, allowing you to use the EMIF Debug Toolkit to configure different traffic pattern for debug purposes.

An issue that you can replicate in RTL simulation indicates a potential bug in the IP. You should recheck the IP parameters. An issue that you can not replicate in RTL simulation indicates a timing issue on the PCB. You can try to replicate the issue on an Altera development platform to rule out a board issue.

**Note:** Ensure that all PCB timing, loading, skew, and deration information is correctly defined in the Quartus Prime software, as the timing report is inaccurate if this initial data is not correct.

Functional simulation allows you to identify any issues with the configuration of either the Altera memory controller and or PHY. You can then easily check the operation against both the memory vendor

data sheet and the respective JEDEC specification. After you resolve functional issues, you can start testing hardware.

For more information about simulation, refer to the *Simulating Memory IP* chapter.

#### Related Information

- [Avalon Interface Specification](#)
- [Simulating Memory IP](#) on page 8-1

## Timing Issue Characteristics

The Altera PHY and controller combinations autogenerate timing constraint files to ensure that the PHY and external interface are fully constrained and that timing is analyzed during compilation. However, timing issues can still occur. This topic discusses how to identify and resolve any timing issues that you may encounter.

Timing issues typically fall into two distinct categories:

- FPGA core timing reported issues
- External memory interface timing issues in a specific mode of operation or on a specific PCB

TimeQuest reports timing issues in two categories: core to core and core to IOE transfers. These timing issues include the PHY and PHY reset sections in the TimeQuest Report DDR subsection of timing analysis. External memory interface timing issues are specifically reported in the TimeQuest Report DDR subsection, excluding the PHY and PHY reset. The Report DDR PHY and PHY reset sections only include the PHY, and specifically exclude the controller, core, PHY-to-controller and local interface. Quartus Prime timing issues should always be evaluated and corrected before proceeding to any hardware testing.

PCB timing issues are usually Quartus Prime timing issues, which are not reported in the Quartus Prime software, if incorrect or insufficient PCB topology and layout information is not supplied. PCB timing issues are typically characterized by calibration failure, or failures during user mode when the hardware is heated or cooled. Further PCB timing issues are typically hidden if the interface frequency is lowered.

## Evaluating FPGA Timing Issues

Usually, you should not encounter timing issues with Altera-provided IP unless your design exceeds Altera's published performance range or you are using a device for which the Quartus Prime software offers only preliminary timing model support. Nevertheless, timing issues can occur in the following circumstances:

- The **.sdc** files are incorrectly added to the Quartus Prime project
- Quartus Prime analysis and synthesis settings are not correct
- Quartus Prime Fitter settings are not correct

For all of these issues, refer to the correct user guide for more information about recommended settings and follow these steps:

1. Ensure that the IP generated **.sdc** files are listed in the Quartus Prime TimeQuest Timing Analyzer files to include in the project window.
2. Ensure that **Analysis and Synthesis Settings** are set to **Optimization Technique Speed**.
3. Ensure that **Fitter Settings** are set to **Fitter Effort Standard Fit**.
4. Use **TimeQuest Report Ignored Constraints**, to ensure that **.sdc** files are successfully applied.
5. Use **TimeQuest Report Unconstrained Paths**, to ensure that all critical paths are correctly constrained.

More complex timing problems can occur if any of the following conditions are true:

- The design includes multiple PHY or core projects
- Devices where the resources are heavily used
- The design includes wide, distributed, maximum performance interfaces in large die sizes

Any of the above conditions can lead to suboptimal placement results when the PHY or controller are distributed around the FPGA. To evaluate such issues, simplify the design to just the autogenerated example top-level file and determine if the core meets timing and you see a working interface. Failure implies that a more fundamental timing issue exists. If the standalone design passes core timing, evaluate how this placement and fit is different than your complete design.

Use LogicLock regions, or design partitions to better define the placement of your memory controllers. When you have your interface standalone placement, repeat for additional interfaces, combine, and finally add the rest of your design.

Additionally, use fitter seeds and increase the placement and router effort multiplier.

## Evaluating External Memory Interface Timing Issues

External memory interface timing issues usually relate to the FPGA input and output characteristics, PCB timing, and the memory component characteristics.

The FPGA input and output characteristics are usually fixed values, because the IOE structure of the devices is fixed. Optimal PLL characteristics and clock routing characteristics do have an effect. Assuming the IP is correctly constrained with autogenerated assignments, and you follow implementation rules, the design should reach the stated performance figures.

Memory component characteristics are fixed for any given component or DIMM. Consider using faster components or DIMMs in marginal cases when PCB skew may be suboptimal, or your design includes multiple ranks when deration may cause read capture or write timing challenges. Using faster memory components often reduces the memory data output skew and uncertainty easing read capture, and lowering the memory's input setup and hold requirement, which eases write timing.

Increased PCB skew reduces margins on address, command, read capture and write timing. If you are narrowly failing timing on these paths, consider reducing the board skew (if possible), or using faster memory. Address and command timing typically requires you to manually balance the reported setup and hold values with the dedicated address and command phase in the IP.

Refer to the respective IP user guide for more information.

Multiple-slot multiple-rank UDIMM interfaces can place considerable loading on the FPGA driver. Typically a quad rank interface can have thirty-six loads. In multiple-rank configurations, Altera's stated maximum data rates are not likely to be achievable because of loading deration. Consider using different topologies, for example registered DIMMs, so that the loading is reduced.

Deration because of increased loading, or suboptimal layout may result in a lower than desired operating frequency meeting timing. You should close timing in the Quartus Prime software using your expected loading and layout rules before committing to PCB fabrication.

Ensure that any design with an Altera PHY is correctly constrained and meets timing in the Quartus Prime software. You must address any constraint or timing failures before testing hardware.

For more information about timing constraints, refer to the *Analyzing Timing of Memory IP* chapter.

#### Related Information

[Analyzing Timing of Memory IP](#) on page 9-1

## Verifying Memory IP Using the SignalTap II Logic Analyzer

The SignalTap II logic analyzer shows read and write activity in the system.

For more information about using the SignalTap II logic analyzer, refer to the *Design Debugging Using the SignalTap II Embedded Logic Analyzer* chapter in volume 3 of the *Quartus Prime Handbook*.

To add the SignalTap II logic analyzer, follow these steps:

1. On the Tools menu click **SignalTap II Logic Analyzer** .
2. In the **Signal Configuration** window next to the **Clock** box, click ... (Browse Node Finder).
3. Type the memory interface system clock (typically \* phy\_clk) in the **Named** box, for **Filter** select **SignalTap II: presynthesis** and click **List**.
4. Select the memory interface clock that is exposed to the user logic.
5. Click **OK**.
6. Under Signal Configuration, specify the following settings:
  - For **Sample depth**, select **512**
  - For **RAM type**, select **Auto**
  - For **Trigger flow control**, select **Sequential**
  - For **Trigger position**, select **Center trigger position**
  - For **Trigger conditions** , select **1**
7. On the Edit menu, click **Add Nodes**.
8. Search for specific nodes that you want to monitor, and click **Add**.

**Note:** SignalTap can probe only nodes that are exposed to FPGA core logic. Refer to pin descriptions in the in the External Memory Interface Handbook for help in deciding which signals to monitor.

9. Decide which signal and event you want to trigger on, and set the corresponding trigger condition.
10. On the File menu, click **Save**, to save the SignalTap II .stp file to your project.

**Note:** If you see the message **Do you want to enable SignalTap II file “stp1.stp” for the current project**, click **Yes**.

11. After you add signals to the SignalTap II logic analyzer, recompile your design by clicking **Start Compilation** on the **Processing** menu.
12. Following compilation, verify that TimeQuest timing analysis passes successfully.
13. Connect the development board to your computer.
14. On the Tools menu, click **SignalTap II Logic Analyzer**.
15. Add the correct <project\_name>.sof file to the SOF Manager:

- a. Click ... to open the **Select Program Files** dialog box.
  - b. Select <your\_project\_name>.sof.
  - c. Click **Open**.
  - d. To download the file, click the **Program Device** button.
16. When the example design including SignalTap II successfully downloads to your development board, click **Run Analysis** to run once, or click **Autorun Analysis** to run continuously.

#### Related Information

##### [Design Debugging Using the SignalTap II Logic Analyzer](#)

## Signals to Monitor with the SignalTap II Logic Analyzer

This topic lists the memory controller signals you should consider analyzing for different memory interfaces. This list is not exhaustive, but is a starting point.

For a description of each signal, refer to *Volume 3: Reference Material of the External Memory Interface Handbook*.

Monitor the following signals for UniPHY designs:

- avl\_addr
- avl\_rdata
- avl\_rdata\_valid
- avl\_read\_req
- avl\_ready
- avl\_wdata
- avl\_write\_req
- fail
- pass
- afi\_cal\_fail
- afi\_cal\_success
- test\_complete
- be\_reg (QDRII only)
- pnf\_per\_bit
- rdata\_reg
- rdata\_valid\_reg
- data\_out
- data\_in
- written\_data\_fifo|data\_out
- usequencer|state \*
- usequencer|phy\_seq\_rdata\_valid
- usequencer|phy\_seq\_read\_fifo\_q
- usequencer|phy\_read\_increment\_vfifo \*
- usequencer|phy\_read\_latency\_counter
- uread\_datapath|afi\_rdata\_en
- uread\_datapath|afi\_rdata\_valid
- uread\_datapath|ddio\_phy\_dq
- qvld\_wr\_address \*
- qvld\_rd\_address \*

**Related Information**  
[Volume 3: Reference Material](#)

## Hardware Debugging Guidelines

Before debugging your design, confirm that it follows the Altera recommended design flow. Refer to the *Design Flow* chapter in volume 1 of the *External Memory Interface Handbook*.

Always keep a record of tests, to avoid repeating the same tests later. To start debugging the design, perform the following initial steps.

**Related Information**  
[Recommended Design Flow](#)

### Create a Simplified Design that Demonstrates the Same Issue

To help debugging create a simple design that replicates the issue.

A simple design should compile quickly and be easy to understand. Altera's external memory interface IP generates an example top-level file that is ideal for debugging. The example top-level file uses all the same parameters, pin-outs, and so on.

If you are using the Arria 10 example design and the Traffic Generator 2.0, you can configure the traffic pattern using the EMIF Debug Toolkit.

**Related Information**  
[External Memory Interface Debug Toolkit](#)

### Measure Power Distribution Network

Measure voltages of the various power supplies on their hardware development platform over a suitable time base and with a suitable trigger.

Ensure that you use an appropriate probe and grounding scheme. In addition, take the measurements directly on the pins or vias of the devices in question, and with the hardware operational.

### Measure Signal Integrity and Setup and Hold Margin

Measure the signals on the PCB. When measuring any signal, consider the edge rate of the signal, not just its frequency. Modern FPGA devices have very fast edge rates, therefore you must use a suitable oscilloscope, probe, and grounding scheme when you measure the signals.

You can take measurements to capture the setup and hold time of key signal classes with respect to their clock or strobe. Ensure that the measured setup and hold margin is at least better than that reported in the Quartus Prime software. A worse margin indicates a timing discrepancy somewhere in the project; however, this issue may not be the cause of your problem.

### Vary Voltage

Vary the voltage of your system, if you suspect a marginality issue.

Increasing the voltage usually causes devices to operate faster and also usually provides increased noise margin.

## Use Freezer Spray and Heat Gun

If you have an intermittent marginal issue, apply cold or heat to the interface to stress the components.

Cooling ICs causes them to run faster, which makes timing easier. Conversely, heating ICs causes them to run slower, which makes timing more difficult.

If cooling or heating corrects the problem, it is probably a timing issue.

## Operate at a Lower Speed

Test the interface at a lower speed. If the interface works at a lower speed, the interface is correctly pinned out and functional.

If the interface fails at a lower speed, determine if the test is valid. Many high-speed memory components have a minimal operating frequency, or require subtly different configurations when operating at a lower speeds.

For example, DDR, DDR2, or DDR3 SDRAM typically requires modification to the following parameters if you want to operate the interface at lower speeds:

- $t_{MRD}$
- $t_{WTR}$
- CAS latency and CAS write latency

## Determine Whether the Issue Exists in Previous Versions of Software

Hardware that works before an update to either the Quartus Prime software or the memory IP indicates that the development platform is not the issue.

However, the previous generation IP may be less susceptible to a PCB issue, masking the issue.

## Determine Whether the Issue Exists in the Current Version of Software

Designs are often tested using previous generations of Altera software or IP.

Projects may not be upgraded for various reasons:

- Multiple engineers are on the same project. To ensure compatibility, a common release of Altera software is used by all engineers for the duration of the product development. The design may be several releases behind the current Quartus Prime software version.
- Many companies delay before adopting a new release of software so that they can first monitor Internet forums to get a feel for how successful other users say the software is.
- Many companies never use the latest version of any software, preferring to wait until the first service pack is released that fixes the primary issues.
- Some users may only have a license for the older version of the software and can only use that version until their company makes the financial decision to upgrade.
- The local interface specification from Altera IP to the customer's logic sometimes changes from software release to software release. If you have already spent resources designing interface logic, you may be reluctant to repeat this exercise. If a block of code is already signed off, you may be reluctant to modify it to upgrade to newer IP from Altera..

In all of the above scenarios, you must determine if the issue still exists in the latest version of the Altera software. Bug fixes and enhancements are added to the Altera IP every release. Depending on the nature of the bug or enhancement, it may not always be clearly documented in the release notes.

Finally, if the latest version of the software resolves the issue, it may be easier to debug the version of software that you are using.

## Try A Different PCB

If you are using the same Altera IP on several different hardware platforms, determine whether the problem occurs on all platforms or just on one.

Multiple instances of the same PCB, or multiple instances of the same interface, on physically different hardware platforms may exhibit different behavior. You can determine if the configuration is fundamentally not working, or if some form of marginality is involved in the issue.

Issues are often reported on the alpha build of a development platform. These are produced in very limited numbers and often have received limited bare-board testing, or functional testing. These early boards are often more unreliable than production quality PCBs.

Additionally, if the IP is from a previous project to help save development resources, determine whether the specific IP configuration works on a previous platform.

## Try Other Configurations

Designs are often quite large, using multiple blocks of IP in many different combinations. Determine whether any other configurations work correctly on the development platform.

The full project may have multiple external memory controllers in the same device, or may have configurations where only half the memory width or frequency is required. Find out what does and does not work to help the debugging of the issue.

## Debugging Checklist

The following checklist is a good starting point when debugging an external memory interface.

**Table 10-1: CheckItem**

Check	Item
<input type="checkbox"/>	Try a different fit.
<input type="checkbox"/>	Check IP parameters at the operating frequency ( $t_{MRD}$ , $t_{WTR}$ for example).
<input type="checkbox"/>	Ensure you have constrained your design with proper timing derivation and have closed timing.
<input type="checkbox"/>	Simulate the design. If it fails in simulation, it will fail in hardware.
<input type="checkbox"/>	Analyze timing.
<input type="checkbox"/>	Place and assign $R_{UP}$ and $R_{DN}$ (OCT).
<input type="checkbox"/>	Measure the power distribution network (PDN).
<input type="checkbox"/>	Measure signal integrity.
<input type="checkbox"/>	Measure setup and hold timing.
<input type="checkbox"/>	Measure FPGA voltages.

Check	Item
<input type="checkbox"/>	Vary voltages.
<input type="checkbox"/>	Heat and cool the PCB.
<input type="checkbox"/>	Operate at a lower or higher frequency.
<input type="checkbox"/>	Check board timing and trace Information.
<input type="checkbox"/>	Check LVDS and clock sources, I/O voltages and termination.
<input type="checkbox"/>	Check PLL clock source, specification, and jitter.
<input type="checkbox"/>	Ensure the correct number of PLL phase steps take place during calibration. If the number stated in the IP does not match the number, you may have manually altered the PLL.
<input type="checkbox"/>	Retarget to a smaller interface width or a single bank.

## Catagorizing Hardware Issues

The following topics divide issues into categories. By determining which category (or categories) an issue belongs in, you may be able to better focus on the cause of the issue.

Hardware issues fall into three categories:

- Signal integrity issues
- Hardware and calibration issues
- Intermittent issues

### Signal Integrity Issues

Many design issues, including some at the protocol layer, can be traced back to signal integrity problems. You should check circuit board construction, power systems, command, and data signaling to determine if they meet specifications.

If infrequent, random errors exist in the memory subsystem, product reliability suffers. Check the bare circuit board or PCB design file. Circuit board errors can cause poor signal integrity, signal loss, signal timing skew, and trace impedance mismatches. Differential traces with unbalanced lengths or signals that are routed too closely together can cause crosstalk.

### Characteristics of Signal Integrity Issues

Signal integrity problems often appear when the performance of the hardware design is marginal.

The design may not always initialize and calibrate correctly, or may exhibit occasional bit errors in user mode. Severe signal integrity issues can result in total failure of an interface at certain data rates, and sporadic component failure because of electrical stress. PCB component variance and signal integrity issues often show up as failures on one PCB, but not on another identical board. Timing issues can have a similar characteristic. Multiple calibration windows or significant differences in the calibration results from one calibration to another can also indicate signal integrity issues.

## Evaluating Signal Integrity Issues

Signal integrity problems can only really be evaluated in two ways:

- direct measurement using suitable test equipment like an oscilloscope and probe
- simulation using a tool like HyperLynx or Allegro PCB SI

Compare signals to the respective electrical specification. You should look for overshoot and undershoot, non-monotonicity, eye height and width, and crosstalk.

### Skew

Ensure that all clocked signals, commands, addresses, and control signals arrive at the memory inputs at the same time.

Trace length variations cause data valid window variations between the signals, reducing margin. For example, DDR2-800 at 400 MHz has a data valid window that is smaller than 1,250 ps. Trace length skew or crosstalk can reduce this data valid window further, making it difficult to design a reliably operating memory interface. Ensure that the skew figure previously entered into the Altera IP matches that actually achieved on the PCB, otherwise Quartus Prime timing analysis of the interface is accurate.

### Crosstalk

Crosstalk is best evaluated early in the memory design phase.

Check the clock-to-data strobes, because they are bidirectional. Measure the crosstalk at both ends of the line. Check the data strobes to clock, because the clocks are unidirectional, these only need checking at the memory end of the line.

### Power System

Some memory interfaces draw current in spikes from their power delivery system as SDRAMs are based on capacitive memory cells.

Rows are read and refreshed one at a time, which causes dynamic currents that can stress any power distribution network (PDN). The various power rails should be checked either at or as close as possible to the SDRAM power pins. Ideally, you should use a real-time oscilloscope set to fast glitch triggering to check the power rails.

### Clock Signals

The clock signal quality is important for any external memory system.

Measurements include frequency, digital core design (DCD), high width, low width, amplitude, jitter, rise, and fall times.

### Read Data Valid Window and Eye Diagram

The memory generates the read signals. Take measurements at the FPGA end of the line.

To ease read diagram capture, modify the example driver to mask writes or modify the PHY to include a signal that you can trigger on when performing reads.

### Write Data Valid Window and Eye Diagram

The FPGA generates the write signals. Take measurements at the memory device end of the line.

To ease write diagram capture, modify the example driver to mask reads or modify the PHY export a signal that is asserted when performing writes.

## OCT and ODT Usage

Modern external memory interface designs typically use OCT for the FPGA end of the line, and ODT for the memory component end of the line. If either the OCT or ODT are incorrectly configured or enabled, signal integrity problems occur.

If the design uses OCT, R<sub>UP</sub> or R<sub>DN</sub> pins must be placed correctly for the OCT to work. If you do not place these pins, the Quartus Prime software allocates them automatically with the following warning:

```
Warning: No exact pin location assignment(s) for 2 pins of 110 total pins
Info: Pin termination_blk0~_rup_pad not assigned to an exact location on the device
Info: Pin termination_blk0~_rdn_pad not assigned to an exact location on the device
```

If you see these warnings, the R<sub>UP</sub> and R<sub>DN</sub> pins may have been allocated to a pin that does not have the required external resistor present on the board. This allocation renders the OCT circuit faulty, resulting in unreliable UniPHY calibration and or interface behavior. The pins with the required external resistor must be specified in the Quartus Prime software.

For the FPGA, ensure that you perform the following:

- Specify the R<sub>UP</sub> and R<sub>DN</sub> pins in either the projects HDL port list, or in the assignment editor (termination\_blk0~\_rup\_pad/ termination\_blk0~\_rdn\_pad).
- Connect the R<sub>UP</sub> and R<sub>DN</sub> pins to the correct resistors and pull-up and pull-down voltage in the schematic or PCB.
- Contain the R<sub>UP</sub> and R<sub>DN</sub> pins within a bank of the device that is operating at the same VCCIO voltage as the interface that is terminated.
- Check that only the expected number of R<sub>UP</sub> and R<sub>DN</sub> pins exists in the project pin-out file. Look for Info: Created on-chip termination messages at the fitter stage for any calibration blocks not expected in your design.
- Review the Fitter Pin-Out file for R<sub>UP</sub> and R<sub>DN</sub> pins to ensure that they are on the correct pins, and that only the correct number of calibration blocks exists in your design.
- Check in the fitter report that the input, output, and bidirectional signals with calibrated OCT all have the termination control block applicable to the associated R<sub>UP</sub> and R<sub>DN</sub> pins.

For the memory components, ensure that you perform the following:

- Connect the required resistor to the correct pin on each and every component, and ensure that it is pulled to the correct voltage.
- Place the required resistor close to the memory component.
- Correctly configure the IP to enable the desired termination at initialization time.
- Check that the speed grade of memory component supports the selected ODT setting.
- Check that the second source part that may have been fitted to the PCB, supports the same ODT settings as the original

## Hardware and Calibration Issues

Hardware and calibration issues have the following definitions:

- Calibration issues result in calibration failure, which usually causes the ctl\_cal\_fail signal to be asserted.
- Hardware issues result in read and write failures, which usually causes the pass not fail (pnf) signal to be asserted.

**Note:** Ensure that functional, timing, and signal integrity issues are not the direct cause of your hardware issue, as functional, timing or signal integrity issues are usually the cause of any hardware issue.

## Evaluating Hardware and Calibration Issues

Use the following methods to evaluate hardware and calibration issues:

- Evaluate hardware issues using the SignalTap II logic analyzer to monitor the local side read and write interface with the pass or fail or error signals as triggers
- Evaluate calibration issues using the SignalTap II logic analyzer to monitor the various calibration, configuration with the pass or fail or error signals as triggers, but also use the debug toolkit and system consoles when available
- For more information about debug toolkit and the type of signals for debugging external memory interfaces, refer to the *External Memory Interface Debug Toolkit* chapter in volume 3 of the *External Memory Interface Handbook*.

Consider adding core noise to your design to aggravate margin timing and signal integrity issues. Steadily increasing the stress on the external memory interface is an ideal way to assess and understand the cause of any previously intermittent failures that you may observe in your system. Using the SignalTap II probe tool can provide insights into the source or cause of operational failure in the system.

Steadily increasing stress on the external memory interface allows you to assess and understand the impact that such factors have on the amount of timing margin and resynchronization window. Take measurements with and without the additional stress factor to allow evaluation of the overall effect.

Steadily increase the stress on the interface in the following order:

1. Increase the interface utilization by modifying the example driver to focus on the types of transactions that exhibit the issue. (For Arria 10 interfaces, you can implement an example design with Traffic Generator 2.0 enabled, and then employ the EMIF Debug Toolkit to configure the data transaction and traffic pattern.)
2. Increase the SSN or aggressiveness of the data pattern by modifying the example driver to output in synchronization PRBS data patterns, or hammer patterns.
3. Increase the stress on the PDN by adding more and more core noise to your system. Try sweeping the fundamental frequency of the core noise to help identify resonances in your power system.

### Related Information

[External Memory Interface Debug Toolkit](#)

## Write Timing Margin

Determine the write timing margin by phase sweeping the write clock from the PLL.

Use sources and probes to dynamically control the PLL phase offset control, to increase and decrease the write clock phase adjustment so that the write window size may be ascertained.

Remember that when sweeping PLL clock phases, the following two factors may cause operational failure:

- The available write margin.
- The PLL phase in a multi-clock system.

The following code achieves this adjustment. You should use sources and probes to modify the respective output of the PLL. Ensure that the example driver is writing and reading from the memory while observing the `pnf_per_byte` signals to see when write failures occur:

```
///////////
wire [7:0] Probe_sig;
wire [5:0] Source_sig;
PhaseCount PhaseCounter (
    .resetn (1'b1),
    .clock (pll_ref_clk),
    .step (Source_sig[5]),
    .updown (Source_sig[4]),
```

**Read Timing Margin**

```

.offset (Probe_sig)
);
CheckoutPands freq_Pands (
.probe (Probe_sig),
.source (Source_sig)
);
ddr2_dimm_phy_alt_mem_phy_pll_siii pll (
.inclk0 (pll_ref_clk),
.areset (pll_reset),
.c0 (phy_clk_1x), // hR
.c1 (mem_clk_2x), // FR
.c2 (aux_clk), // FR
.c3 (write_clk_2x), // FR
.c4 (resync_clk_2x), // FR
.c5 (measure_clk_1x), // hR
.c6 (ac_clk_1x), // hR
.phasecounterselect (Source_sig[3:0]),
.phasestep (Source_sig[5]),
.phaseupdown (Source_sig[4]),
.scanclk (scan_clk),
.locked (pll_locked_src),
.phasedone (pll_phase_done)
);

```

**Read Timing Margin**

Assess the read timing margin by using sources and probes to manually control the DLL phase offset feature.

Open the autogenerated DLL using ALT\_DLL and add the additionally required offset control ports. This action allows control and observation of the following signals:

```

dll_delayctrlout[5:0], // Phase output control from DLL to DQS pins (Gray Coded)
dll_offset_ctrl_a_addnsub, // Input add or subtract the phase offset value
dll_offset_ctrl_a_offset[5:0], // User Input controlled DLL offset value (Gray Coded)
dll_aload, // User Input DLL load command
dll_dqsupdate, // DLL Output update required signal.

```

In examples where the applied offset applied results in the maximum or minimum dll\_delayctrlout[5:0] setting without reaching the end of the read capture window, regenerate the DLL in the next available phase setting, so that the full capture window is assessed.

Modify the example driver to constantly perform reads (mask writes). Observe the pnf\_per\_byte signals while the DLL capture phase is manually modified to see when failures begin, which indicates the edge of the window.

A resynchronization timing failure can indicate failure at that capture phase, and not a capture failure. You should recalibrate the PHY with the calculated phase offset to ensure that you are using the true read-capture margin.

**Address and Command Timing Margin**

You set the address and command clock phase directly in the IP. Assuming you enter the correct board trace model information into the Quartus Prime software, the timing analysis should be correct.

If you want to evaluate the address and command timing margin, use the same process as in “Write Timing Margin”, only phase step the address and command PLL output (c6 ac\_clk\_1x). You can achieve this effect using the debug toolkit or system console.

Refer to the *External Memory Interface Debug Toolkit* chapter in volume 3 of the *External Memory Interface Handbook*.

### Related Information

- [Write Timing Margin](#) on page 10-19
- [External Memory Interface Debug Toolkit](#)

### Resynchronization Timing Margin

Observe the size and margins available for resynchronization using the debug toolkit or system console.

Refer to *External Memory Interface Debug Toolkit* chapter in volume 3 of the *External Memory Interface Handbook*.

Additionally for PHY configurations that use a dedicated PLL clock phase (as opposed to a resynchronization FIFO buffer), use the same process as described in “Write Timing Margin”, to dynamically sweep resynchronization margin (`c4_resynch_clk_2x`).

### Related Information

- [Write Timing Margin](#) on page 10-19
- [External Memory Interface Debug Toolkit](#)

### Postamble Timing Issues and Margin

The postamble timing is set by the PHY during calibration.

You can diagnose postamble issues by viewing the `pnf_per_byte` signal from the example driver. Postamble timing issues mean only read data is corrupted during the last beat of any read request.

### Intermittent Issue Evaluation

Intermittent issues are typically the hardest type of issue to debug—they appear randomly and are hard to replicate.

Errors that occur during run-time indicate a data-related issue, which you can identify by the following actions:

- Add the SignalTap II logic analyzer and trigger on the post-trigger `pnf`
- Use a stress pattern of data or transactions, to increase the probability of the issue
- Heat up or cool down the system
- Run the system at a slightly faster frequency

If adding the SignalTap II logic analyzer or modifying the project causes the issue to go away, the issue is likely to be placement or timing related.

Errors that occur at start-up indicate that the issue is related to calibration, which you can identify by the following actions:

- Modify the design to continually calibrate and reset in a loop until the error is observed
- Where possible, evaluate the calibration margin either from the debug toolkit or system console.

**Note:** Refer to the *External Memory Interface Debug Toolkit* chapter in volume 3 of the *External Memory Interface Handbook*.

- Capture the calibration error stage or error code, and use this information with whatever specifically occurs at that stage of calibration to assist with your debugging of the issue.

### Related Information

- [External Memory Interface Debug Toolkit](#)

## EMIF Debug Toolkit Overview

The EMIF Debug Toolkit is a Tcl-based interface that runs on your PC and communicates via a JTAG connection to enable you to debug your external memory interface on the circuit board, retrieve calibration status, and perform margining activities. The EMIF Debug Toolkit does not support MAX 10 devices.

**Note:** For more information about the EMIF Debug Toolkit, refer to the *External Memory Interface Debug Toolkit* in volume 3 of the *External Memory Interface Handbook*.

### Related Information

#### [External Memory Interface Debug Toolkit](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"> <li>• Added DDR4, LPDDR2, LPDDR3, QDRII+ Xtreme, QDR-IV, and RLDRAM 3 to the list of IP in <i>Debugging Memory IP</i>.</li> <li>• Added a paragraph about using the Traffic Generator 2.0 with Arria 10 interfaces, to <i>Modifying the Example Driver to Replicate the Failure</i>.</li> <li>• Added a paragraph about using the Traffic Generator 2.0 with Arria 10 interfaces, to <i>Create a Simplified Design that Demonstrates the Same Issue</i>.</li> <li>• Added a comment about using the EMIF Debug Toolkit with Arria 10 example designs using the Traffic Generator 2.0, to item 1 in <i>Evaluating Hardware and Calibration Issues</i>.</li> <li>• Added DDR3, DDR4, and LPDDR3 controllers for Arria 10 EMIF IP to the list of toolkit components in <i>EMIF Debug Toolkit Overview and Usage Flow</i>.</li> <li>• </li> </ul>
November 2015	2015.11.02	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Maintenance release.
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>• Removed references to ALTMEMPHY.</li> <li>• Removed local_wdata_req from step 9 of <i>Verifying Memory IP Using SignalTap II Logic Analyzer</i>.</li> </ul>
November 2012	4.2	Changed chapter number from 11 to 12.

Date	Version	Changes
June 2012	4.1	Added Feedback icon.
November 2011	4.0	Added <i>Debug Toolkit</i> section.
June 2011	3.0	Removed leveling information from <i>ALTMEMPHY Calibration Stages</i> and <i>UniPHY Calibration Stages</i> chapter.
December 2010	2.1	<ul style="list-style-type: none"><li>Added new chapter: <i>UniPHY Calibration Stages</i>.</li><li>Added new chapter: <i>DDR2 and DDR3 SDRAM Controllers with UniPHY EMIF Toolkit</i>.</li></ul>
July 2010	2.0	Updated for 10.0 release.
January 2010	1.2	Corrected typos.
December 2009	1.1	Added <i>Debug Toolkit for DDR2 and DDR3 SDRAM High-Performance Controllers</i> chapter and <i>ALTMEMPHY Calibration Stages</i> chapter.
November 2009	1.0	Initial release.

# Optimizing the Controller 11

2016.05.02

EMI\_DG



Subscribe



Send Feedback

It is important that you understand how to increase the efficiency and bandwidth of the memory controller when you design any external memory interface.

The following topics discuss factors that affect controller efficiency and ways to increase the efficiency of the controller.

## Controller Efficiency

Controller efficiency varies depending on data transaction. The best way to determine the efficiency of the controller is to simulate the memory controller for your specific design.

Controller efficiency is expressed as:

Efficiency = number of active cycles of data transfer/total number of cycles

The total number of cycles includes the number of cycles required to issue commands or other requests.

**Note:** You calculate the number of active cycles of data transfer in terms of local clock cycles. For example, if the number of active cycles of data transfer is 2 memory clock cycles, you convert that to the local clock cycle which is 1.

The following cases are based on a DDR2 SDRAM high-performance controller design targeting a Stratix<sup>®</sup> IV device that has a CAS latency of 3, and burst length of 4 on the memory side (2 cycles of data transfer), with accessed bank and row in the memory device already open. The Stratix IV device has a command latency of 9 cycles in half-rate mode. The `local_ready` signal is high.

- Case 1: The controller performs individual reads.

Efficiency =  $1/(1 + \text{CAS} + \text{command latency}) = 1/(1+1.5+9) = 1/11.5 = 8.6\%$

- Case 2: The controller performs 4 back to back reads.

In this case, the number of data transfer active cycles is 8. The CAS latency is only counted once because the data coming back after the first read is continuous. Only the CAS latency for the first read has an impact on efficiency. The command latency is also counted once because the back to back read commands use the same bank and row.

Efficiency =  $4/(4 + \text{CAS} + \text{command latency}) = 4/(4+1.5+9) = 1/14.5 = 27.5\%$

## Factors Affecting Efficiency

The two main factors that affect controller efficiency are the interface standard specified by the memory vendor, and the way that you transfer data.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

**ALTERA**  
now part of Intel

The following sections discuss these two factors in detail.

## Interface Standard

Complying with certain interface standard specifications affects controller efficiency.

When interfacing the memory with the DDR2 or DDR3 SDRAM controllers, you must follow certain timing specifications and perform the following bank management operations:

- **Activate**

Before you issue any read (RD) or write (WR) commands to a bank within a DDR2 SDRAM device, you must open a row in that bank using the activate (ACT) command. After you open a row, you can issue a read or write command to that row based on the  $t_{RCD}$  specification. Reading or writing to a closed row has negative impact on the efficiency as the controller has to first activate that row and then wait until  $t_{RCD}$  time to perform a read or write.

- **Precharge**

To open a different row in the same bank, you must issue a precharge (PCH) command. The precharge command deactivates the open row in a particular bank or the open row in all banks. Switching a row has a negative impact on the efficiency as you must first precharge the open row, then activate the next row and wait  $t_{RCD}$  time to perform any read or write operation to the row.

- **Device CAS latency**

The higher the CAS latency, the less efficient an individual access. The memory device has its own read latency, which is about 12 ns to 20 ns regardless of the actual frequency of the operation. The higher the operating frequency, the longer the CAS latency is in number of cycles.

- **Refresh**

A refresh, in terms of cycles, consists of the precharge command and the waiting period for the auto refresh. Based on the memory data sheet, these components require the following values:

- $t_{RP} = 12$  ns, 3 clock cycles for a 200-MHz operation (5 ns period for 200 MHz)
- $t_{RFC} = 75$  ns, 15 clock cycles for a 200-MHz operation.

Based on this calculation, a refresh pauses read or write operations for 18 clock cycles. So, at 200 MHz, you lose 1.15% ( $18 \times 5$  ns/7.8 us) of the total efficiency.

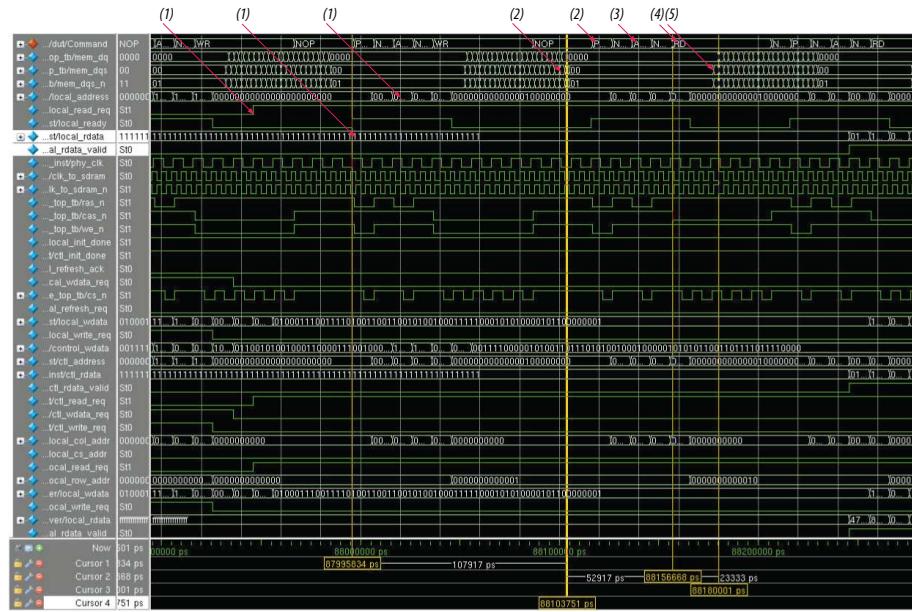
## Bank Management Efficiency

The following figures show examples of how the bank management operations affect controller efficiency.

The first figure shows a read operation in which you have to change a row in a bank. This figure shows how CAS latency and precharge and activate commands affect efficiency.

The following figure illustrates a read-after-write operation. The controller changes the row address after the write-to-read from a different row.

Figure 11-1: Read Operation—Changing A Row in A Bank



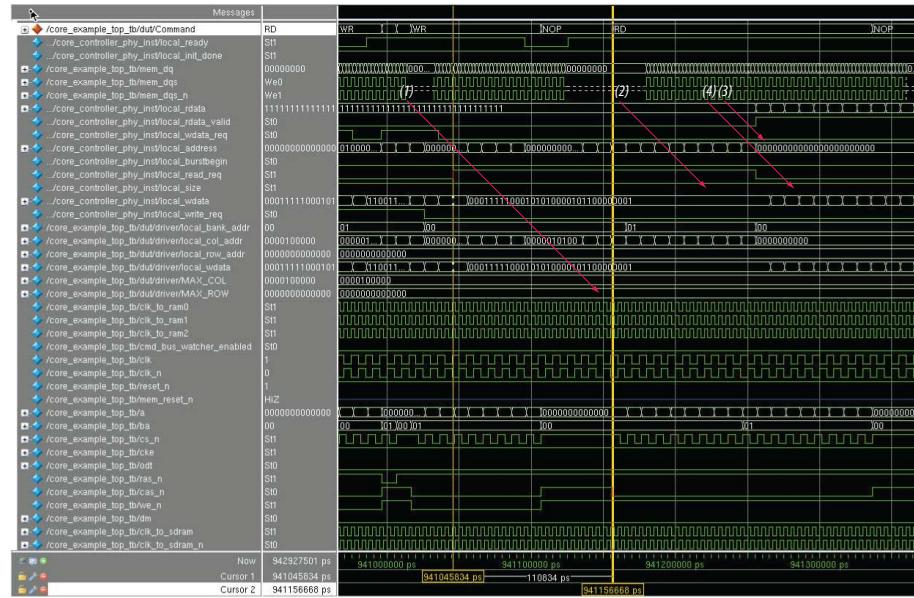
The following sequence of events describes the above figure:

1. The `local_read_req` signal goes high, and when the `local_ready` signal goes high, the controller accepts the read request along with the address.
2. After the memory receives the last write data, the row changes for read. Now you require a precharge command to close the row opened for write. The controller waits for  $t_{WTR}$  time (3 memory clock cycles) to give the precharge command after the memory receives the last write data.
3. After the controller issues the precharge command, it must wait for  $t_{RP}$  time to issue an activate command to open a row.
4. After the controller gives the activate command to activate the row, it needs to wait  $t_{RCD}$  time to issue a read command.
5. After the memory receives the read command, it takes the memory some time to provide the data on the pin. This time is known as CAS latency, which is 3 memory clock cycles in this case.

For this particular case, you need approximately 17 local clock cycles to issue a read command to the memory. Because the row in the bank changes, the read operation takes a longer time, as the controller has to issue the precharge and activate commands first. You do not have to take into account  $t_{WTR}$  for this case because the precharge and activate operations already exceeded  $t_{WTR}$  time.

The following figure shows the case where you use the same the row and bank address when the controller switches from write to read. In this case, the read command latency is reduced.

Figure 11-2: Changing From Write to Read—Same Row and Bank Address



The following sequence of events describes the above figure:

1. The `local_read_req` signal goes high and the `local_ready` signal is high already. The controller accepts the read request along with the address.
2. When switching from write to read, the controller has to wait  $t_{WTR}$  time before it gives a read command to the memory.
3. The SDRAM device receives the read command.
4. After the SDRAM device receives the read command, it takes some time to give the data on the pin. This time is called CAS latency, which is 3 memory clock cycles in this case.

For the case illustrated in the second figure above, you need approximately 11 local clock cycles to issue a read command to the memory. Because the row in the bank remains the same, the controller does not have to issue the precharge and activate commands, which speeds up the read operation and in turn results in a better efficiency compared to the case in the first figure above.

Similarly, if you do not switch between read and write often, the efficiency of your controller improves significantly.

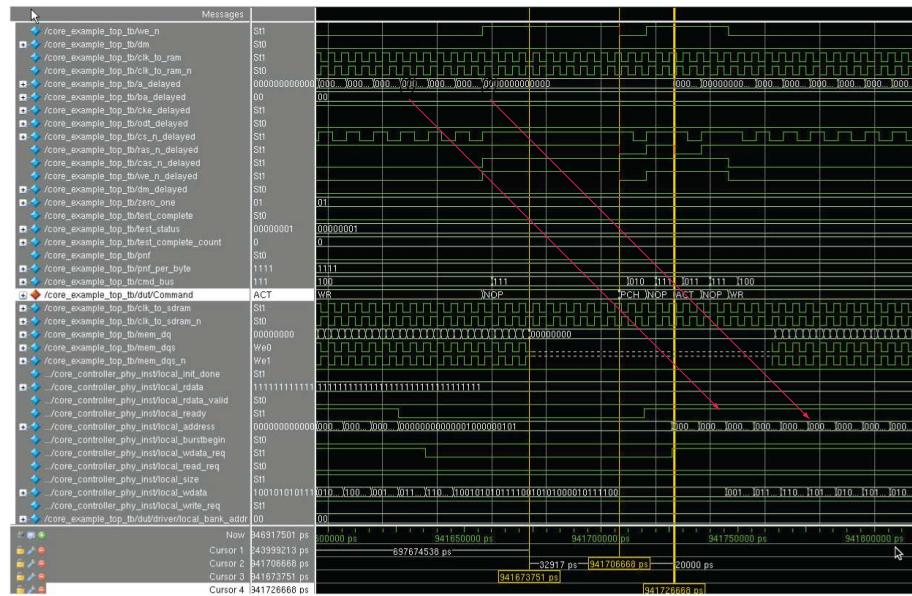
## Data Transfer

The following methods of data transfer reduce the efficiency of your controller:

- Performing individual read or write accesses is less efficient.
- Switching between read and write operation has a negative impact on the efficiency of the controller.
- Performing read or write operations from different rows within a bank or in a different bank—if the bank and a row you are accessing is not already open—also affects the efficiency of your controller.

The following figure shows an example of changing the row in the same bank.

Figure 11-3: Changing Row in the Same Bank



The following sequence of events describes the above figure:

1. You have to wait  $t_{WR}$  time before giving the precharge command
2. You then wait  $t_{RP}$  time to give the activate command.

## Ways to Improve Efficiency

To improve the efficiency of your controller, you can use the following tools and methods:

- DDR2 SDRAM Controller
- Auto-Precharge Commands
- Additive Latency
- Bank Interleaving
- Command Queue Look-Ahead Depth
- Additive Latency and Bank Interleaving
- User-Controlled Refresh
- Frequency of Operation
- Burst Length
- Series of Reads or Writes

The following sections discuss these methods in detail.

## DDR2 SDRAM Controller

The DDR2 SDRAM controller maintains up to eight open banks; one row in each bank is open at a time.

Maintaining more banks at one time helps avoid bank management commands. Ensure that you do not change a row in a bank frequently, because changing the row in a bank causes the bank to close and reopen to open another row in that bank.

## Auto-Precharge Commands

The auto-precharge read and write commands allow you to indicate to the memory device that a given read or write command is the last access to the currently opened row.

The memory device automatically closes or auto-precharges the page that is currently being accessed, so that the next access to the same bank is faster. The Auto-Precharge command is useful when you want to perform fast random memory accesses.

The Timer Bank Pool (TBP) block supports the dynamic page policy, where depending on user input on local autoprecharge input would keep a page open or close. In a closed-page policy, a page is always closed after it is accessed with auto-precharge command. When the data pattern consists of repeated reads or writes to addresses not within the same page, the optimal system achieves the maximum efficiency allowed by continuous page miss limited access. Efficiency losses are limited to those associated with activating and refreshing. An efficiency of 10-20% should be expected for this closed-page policy.

In an open-page policy, the page remains open after it is accessed for incoming commands. When the data pattern consists of repeated reads or writes to sequential addresses within the same page, the optimal system can achieve 100% efficiency for page-open transactions (ignoring the effects of periodic refreshes, which typically consume around 2-3% of total efficiency), with minimum latency for highest priority single transactions.

If you turn on **Enable Auto-Precharge Control**, you can instruct the controller to issue an autoprecharge read or write command. The next time you access that bank, the access will be faster because the controller does not have to precharge the bank before activating the row that you want to access.

The controller-derived autoprecharge logic evaluates the pending commands in the command buffer and determines the most efficient autoprecharge operation to perform. The autoprecharge logic can reorder commands if necessary. When all TBP are occupied due to tracking an open page, TBP uses a scheme called on-demand flush, where it stops tracking a page to create space for an incoming command.

The following figure compares auto-precharge with and without look-ahead support.

**Figure 11-4: Comparison With and Without Look-ahead Auto-Precharge**

Without Look-ahead Auto-Precharge			Look-ahead Auto-Precharge		
Cycle	Command	Data	Cycle	Command	Data
1	WRITE		1	WRITE with AP	
2	NOP	DATA0 (Burst 0, Burst 1)	2	NOP	DATA0 (Burst 0, Burst 1)
3	ACT	DATA0 (Burst 2, Burst 3)	3	ACT	DATA0 (Burst 2, Burst 3)
4	NOP	DATA0 (Burst 4, Burst 5)	4	NOP	DATA0 (Burst 4, Burst 5)
5	WRITE	DATA0 (Burst 6, Burst 7)	5	WRITE	DATA0 (Burst 6, Burst 7)
6	NOP	DATA1 (Burst 0, Burst 1)	6	NOP	DATA1 (Burst 0, Burst 1)
7	ACT	DATA1 (Burst 2, Burst 3)	7	ACT	DATA1 (Burst 2, Burst 3)
8	NOP	DATA1 (Burst 4, Burst 5)	8	NOP	DATA1 (Burst 4, Burst 5)
9	WRITE	DATA1 (Burst 6, Burst 7)	9	WRITE	DATA1 (Burst 6, Burst 7)
10	NOP	DATA2 (Burst 0, Burst 1)	10	NOP	DATA2 (Burst 0, Burst 1)
11	PCH	DATA2 (Burst 2, Burst 3)	11	ACT	DATA2 (Burst 2, Burst 3)
12	NOP	DATA2 (Burst 4, Burst 5)	12	NOP	DATA2 (Burst 4, Burst 5)
13	ACT	DATA2 (Burst 6, Burst 7)	13	WRITE	DATA2 (Burst 6, Burst 7)
14	NOP	Wasted Cycle	14	NOP	DATA3 (Burst 0, Burst 1)
15	WRITE	Wasted Cycle	15	NOP	DATA3 (Burst 2, Burst 3)
16	NOP	DATA3 (Burst 0, Burst 1)	16	NOP	DATA3 (Burst 4, Burst 5)
17	NOP	DATA3 (Burst 2, Burst 3)	17	NOP	DATA3 (Burst 6, Burst 7)
18	NOP	DATA3 (Burst 4, Burst 5)			
19	NOP	DATA3 (Burst 6, Burst 7)			

Command	Bank	Row	Condition
Write	Bank 0	Row 0	
Write	Bank 1	Row 0	Activate required
Write	Bank 2	Row 0	Activate required
Write	Bank 0	Row 1	Precharge required

Without using the look-ahead auto-precharge feature, the controller must precharge to close and then open the row before the write or read burst for every row change. When using the look-ahead precharge feature, the controller decides whether to do auto-precharge read/write by evaluating the incoming command; subsequent reads or writes to same bank/different row will require only an activate command.

As shown in the preceding figure, the controller performs an auto-precharge for the write command to bank 0 at cycle 1. The controller detects that the next write at cycle 13 is to a different row in bank 0, and hence saves 2 data cycles.

The following efficiency results apply to the above figure:

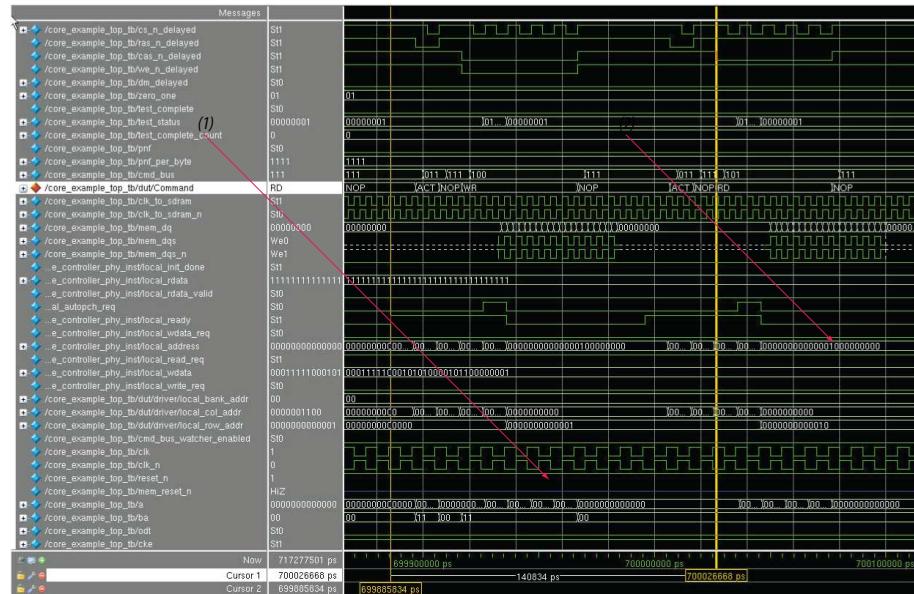
**Table 11-1: Comparative Efficiencies With and Without Look-Ahead Auto-Precharge Feature**

	Without Look-ahead Auto-precharge	With Look-ahead Auto-precharge
Active cycles of data transfer	16	16
Total number of cycles	19	17
Approximate efficiency	84%	94%

The look-ahead auto-precharge used increases efficiency by approximately 10%.

The following figure shows how you can improve controller efficiency using the auto-precharge command.

Figure 11-5: Improving Efficiency Using Auto-Precharge Command



The following sequence of events describes the above figure:

1. The controller accepts a read request from the local side as soon as the `local_ready` signal goes high.
2. The controller gives the activate command and then gives the read command. The read command latency is approximately 14 clock cycles for this case as compared to the similar case with no auto precharge which had approximately 17 clock cycles of latency (described in the "data Transfer" topic).

When using the auto-precharge option, note the following guidelines:

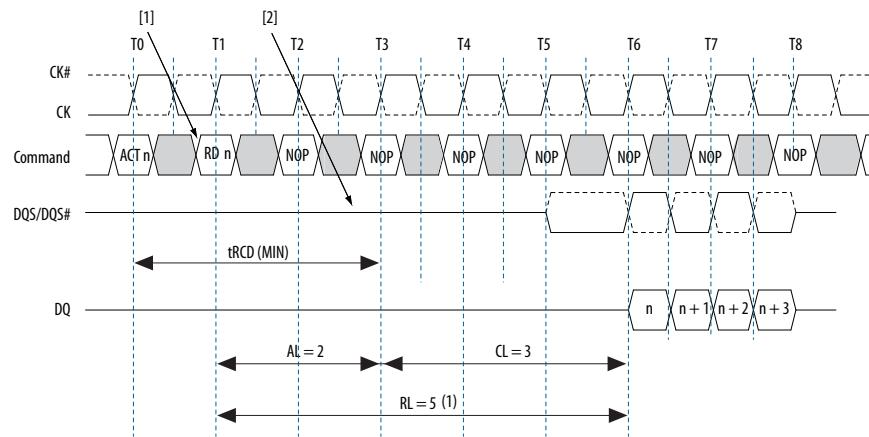
- Use the auto-precharge command if you know the controller is issuing the next read or write to a particular bank and a different row.
- Auto-precharge does not improve efficiency if you auto-precharge a row and immediately reopen it.

## Additive Latency

Additive latency increases the efficiency of the command and data bus for sustainable bandwidths.

You may issue the commands externally but the device holds the commands internally for the duration of additive latency before executing, to improve the system scheduling. The delay helps to avoid collision on the command bus and gaps in data input or output bursts. Additive latency allows the controller to issue the row and column address commands—activate, and read or write—in consecutive clock cycles, so that the controller need not hold the column address for several ( $t_{RCD}$ ) cycles. This gap between the activate and the read or write command can cause bubbles in the data stream.

The following figure shows an example of additive latency.

**Figure 11-6: Additive Latency—Read**

The following sequence of events describes the above figure:

1. The controller issues a read or write command before the  $t_{RCD}$  (MIN) requirement—additive latency less than or equal to  $t_{RCD}$  (MIN).
2. The controller holds the read or write command for the time defined by additive latency before issuing it internally to the SDRAM device.

Read latency = additive latency + CAS latency

Write latency = additive latency + CAS latency -  $t_{CK}$

## Bank Interleaving

You can use bank interleaving to sustain bus efficiency when the controller misses a page, and that page is in a different bank.

**Note:** Page size refers to the minimum number of column locations on any row that you access with a single activate command. For example: For a 512Mb x8 DDR3 SDRAM with 1024 column locations (column address A[9:0]), page size = 1024 columns x 8 = 8192 bits = 8192/8 bytes = 1024 bytes (1 KB)

Without interleaving, the controller sends the address to the SDRAM device, receives the data requested, and then waits for the SDRAM device to precharge and reactivate before initiating the next data transaction, thus wasting several clock cycles.

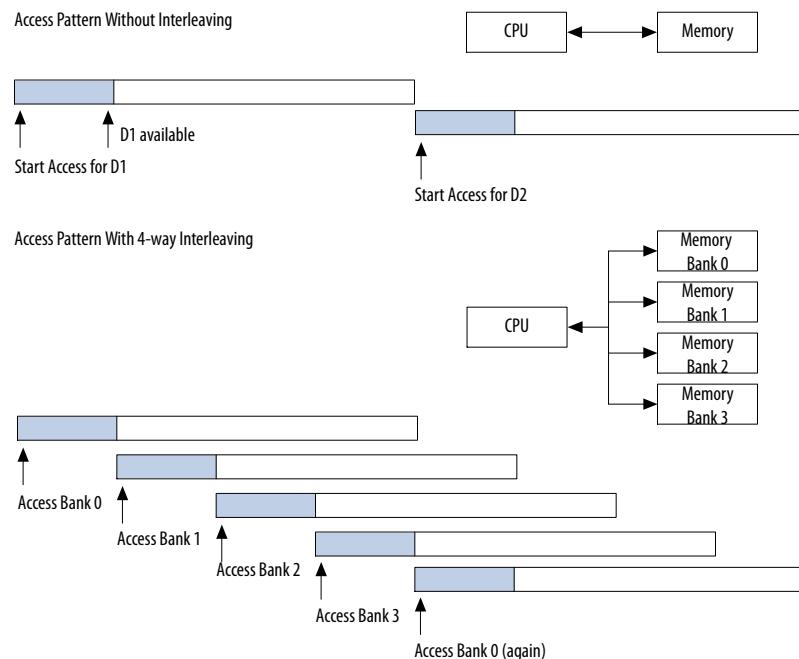
Interleaving allows banks of the SDRAM device to alternate their background operations and access cycles. One bank undergoes its precharge/activate cycle while another is being accessed. By alternating banks, the controller improves its performance by masking the precharge/activate time of each bank. If there are four banks in the system, the controller can ideally send one data request to each of the banks in consecutive clock cycles.

For example, in the first clock cycle, the CPU sends an address to Bank 0, and then sends the next address to Bank 1 in the second clock cycle, before sending the third and fourth addresses to Banks 2 and 3 in the third and fourth clock cycles respectively. The sequence is as follows:

1. Controller sends address 0 to Bank 0.
2. Controller sends address 1 to Bank 1 and receives data 0 from Bank 0.
3. Controller sends address 2 to Bank 2 and receives data 1 from Bank 1.
4. Controller sends address 3 to Bank 3 and receives data 2 from Bank 2.
5. Controller receives data 3 from Bank 3.

The following figure shows how you can use interleaving to increase bandwidth.

**Figure 11-7: Using Interleaving to Increase Bandwidth**



The Altera controller supports three interleaving options:

**Chip-Bank-Row-Col** – This is a noninterleaved option. Select this option to improve efficiency with random traffic

**Chip-Row-Bank-Col** – This option uses bank interleaving without chip select interleaving. Select this option to improve efficiency with sequential traffic, by spreading smaller data structures across all banks in a chip.

**Row-Chip-Bank-Col** - This option uses bank interleaving with chip select interleaving. Select this option to improve efficiency with sequential traffic and multiple chip selects. This option allows smaller data structures to spread across multiple banks and chips.

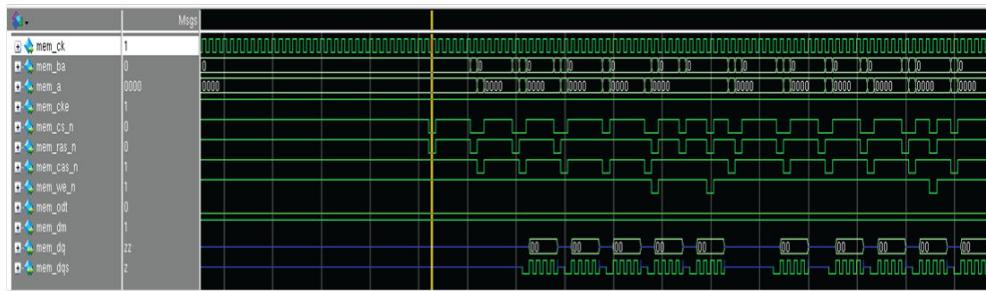
Bank interleaving is a fixed pattern of data transactions, enabling best-case bandwidth and latency, and allowing for sufficient interleaved transactions between opening banks to completely hide  $t_{RC}$ . An optimal system can achieve 100% efficiency for bank interleave transactions with 8 banks. A system with less than 8 banks is unlikely to achieve 100%.

## Command Queue Look-Ahead Depth

The command queue look-ahead depth value determines the number of read or write requests that the look-ahead bank management logic examines. The command queue look-ahead depth value also determines how many open pages the High-Performance Controller II (HPC II) can track.

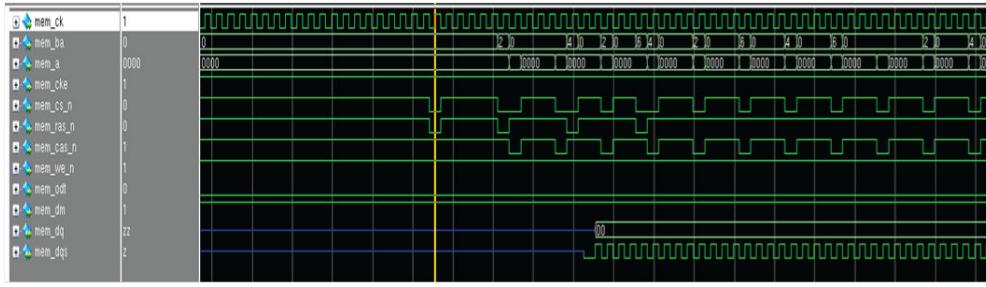
For example, if you set the command queue look-ahead depth value to 4, the HPC II controller can track 4 open pages. In a 4-bank interleaving case, HPCII will receive repeated commands with addresses of bank A, bank B, bank C, and bank D. To receive the next set of commands, the controller issues a precharge command to exit the current page and then issues an activate command to track the new incoming page, leading to a drop in efficiency.

**Figure 11-8: Simulation with Command Queue Look-ahead Depth of 4**



With the command queue look-ahead depth set to 8, the controller can track 8 open pages and overall efficiency is much improved relative to a command queue look-ahead value of 4.

**Figure 11-9: Simulation with Command Queue Look-ahead Depth of 8**



There is a trade-off between efficiency and resource usage. Higher command queue look-ahead values are likely to increase bank management efficiency, but at the cost of higher resource usage. Smaller command queue look-ahead values may be less efficient, but also consume fewer resources. Also, a command queue look-ahead value greater than 4 may cause timing violations for interfaces approaching their maximum frequency.

**Note:** If you set Command Queue Look-ahead depth to a value greater than 4, you may not be able to run the interface at maximum frequency.

To achieve an optimized balance of controller efficiency versus resource usage and frequency, you must understand your traffic patterns. You should simulate your design with a variety of controller settings to observe the results of different settings.

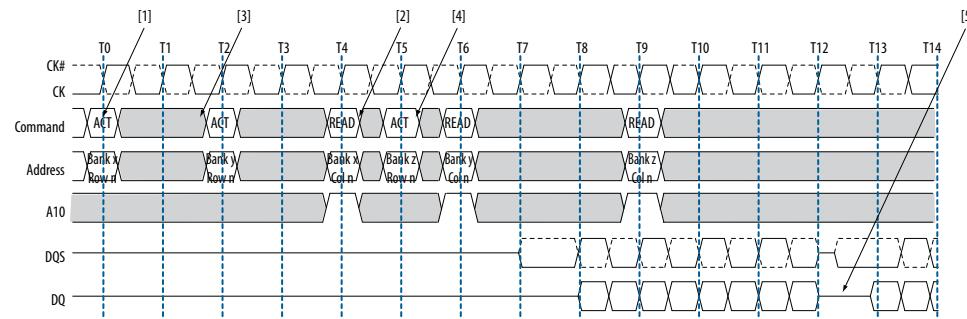
**Note:** User-selectable Command Queue Look-ahead depth is available only when using the soft memory controller. For the hard memory controller, the Command Queue Look-ahead depth value is hard-coded to 8.

## Additive Latency and Bank Interleaving

Using additive latency together with bank interleaving increases the bandwidth of the controller.

The following figure shows an example of bank interleaving in a read operation without additive latency. The example uses DDR2 SDRAM bank interleave reads with CAS latency of 4, and burst length of 4.

**Figure 11-10: Bank Interleaving—Without Additive Latency**



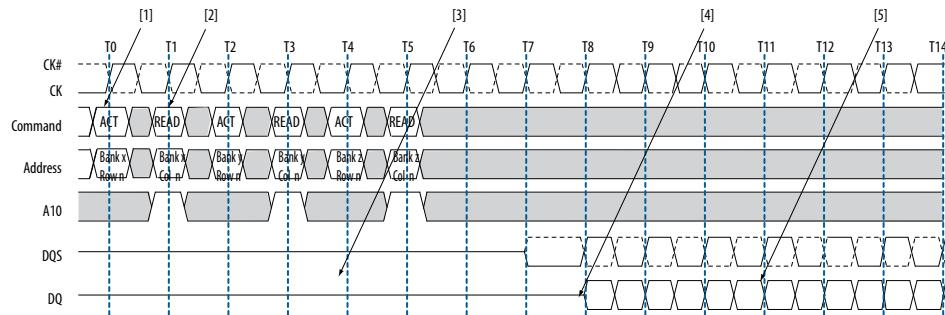
The following sequence of events describes the above figure:

1. The controller issues an activate command to open the bank, which activates bank  $x$  and the row in it.
2. After  $t_{RCD}$  time, the controller issues a read with auto-precharge command to the specified bank.
3. Bank  $y$  receives an activate command after  $t_{RRD}$  time.
4. The controller cannot issue an activate command to bank  $z$  at its optimal location because it must wait for bank  $x$  to receive the read with auto-precharge command, thus delaying the activate command for one clock cycle.
5. The delay in activate command causes a gap in the output data from the DDR2 SDRAM device.

**Note:** If you use additive latency of 1, the latency affects only read commands and not the timing for write commands.

The following figure shows an example of bank interleaving in a read operation with additive latency. The example uses DDR2 SDRAM bank interleave reads with additive latency of 3, CAS latency of 4, and burst length of 4. In this configuration, the controller issues back-to-back activate and read with auto-precharge commands.

**Figure 11-11: Bank Interleaving—With Additive Latency**



The following sequence of events describes the above figure:

1. The controller issues an activate command to bank  $x$ .
2. The controller issues a read with auto precharge command to bank  $x$  right after the activate command, before waiting for the  $t_{RCD}$  time.
3. The controller executes the read with auto-precharge command  $t_{RCD}$  time later on the rising edge T4.
4. 4 cycles of CAS latency later, the SDRAM device issues the data on the data bus.
5. For burst length of 4, you need 2 cycles for data transfer. With 2 clocks of giving activate and read with auto-precharge commands, you get a continuous flow of output data.

Compare the efficiency results in the two preceding figures:

- DDR2 SDRAM bank interleave reads with no additive latency, CAS latency of 4, and burst length of 4 (first figure),

Number of active cycles of data transfer = 6.

Total number of cycles = 15

Efficiency = 40%

- DDR2 SDRAM bank interleave reads with additive latency of 3, CAS latency of 4, and burst length of 4 (second figure),

Number of active cycles of data transfer = 6.

Total number of cycles = 14

Efficiency = approximately 43%

The interleaving reads used with additive latency increases efficiency by approximately 3%.

**Note:** Additive latency improves the efficiency of back-to-back interleaved reads or writes, but not individual random reads or writes.

## User-Controlled Refresh

The requirement to periodically refresh memory contents is normally handled by the memory controller; however, the **User Controlled Refresh** option allows you to determine when memory refresh occurs.

With specific knowledge of traffic patterns, you can time the refresh operations so that they do not interrupt read or write operations, thus improving efficiency.

**Note:** If you enable the auto-precharge control, you must ensure that the average periodic refresh requirement is met, because the controller does not issue any refreshes until you instruct it to.

## Frequency of Operation

Certain frequencies of operation give you the best possible latency based on the memory parameters. The memory parameters you specify through the parameter editor are converted to clock cycles and rounded up.

If you are using a memory device that has  $t_{RCD} = 20$  ns and running the interface at 100 MHz, you get the following results:

- For full-rate implementation ( $t_{Ck} = 10$  ns):  
 $t_{RCD}$  convert to clock cycle =  $20/10 = 2$ .
- For half rate implementation ( $t_{Ck} = 20$  ns):  
 $t_{RCD}$  convert to clock cycle =  $20/20 = 1$

This frequency and parameter combination is not easy to find because there are many memory parameters and frequencies for the memory device and the controller to run. Memory device parameters are optimal for the speed at which the device is designed to run, so you should run the device at that speed.

In most cases, the frequency and parameter combination is not optimal. If you are using a memory device that has  $t_{RCD} = 20$  ns and running the interface at 133 MHz, you get the following results:

- For full-rate implementation ( $t_{Ck} = 7.5$  ns):

$t_{RCD}$  convert to clock cycle =  $20/7.5 = 2.66$ , rounded up to 3 clock cycles or 22.5 ns.

- For half rate implementation ( $t_{Ck} = 15$  ns):

$t_{RCD}$  convert to clock cycle =  $20/15 = 1.33$ , rounded up to 2 clock cycles or 30 ns.

There is no latency difference for this frequency and parameter combination.

## Burst Length

Burst length affects the efficiency of the controller. A burst length of 8 provides more cycles of data transfer, compared to a burst length of 4.

For a half-rate design that has a command latency of 9 half-rate clock cycles, and a CAS latency of 3 memory clock cycles or 1.5 half rate local clock cycles, the efficiency is 9% for burst length of 4, and 16% for burst length of 8.

- Burst length of 4 (2 memory clock cycles of data transfer or 1 half-rate local clock cycle)

Efficiency = number of active cycles of data transfer/total number of cycles

Efficiency =  $1/(1 + \text{CAS} + \text{command latency}) = 1/(1 + 1.5 + 9) = 1/11.5 = 8.6\%$  or approximately 9%

- Burst length of 8 (4 memory clock cycles of data transfer or 2 half-rate local clock cycles)

Efficiency = number of active cycles of data transfer/total number of cycles

Efficiency =  $2/(2 + \text{CAS} + \text{command latency}) = 2/(2 + 1.5 + 9) = 2/12.5 = 16\%$

## Series of Reads or Writes

Performing a series of reads or writes from the same bank and row increases controller efficiency.

The case shown in the second figure in the "Bank Management Efficiency" topic demonstrates that a read performed from the same row takes only 14.5 clock cycles to transfer data, making the controller 27% efficient.

Do not perform random reads or random writes. When you perform reads and writes to random locations, the operations require row and bank changes. To change banks, the controller must precharge the previous bank and activate the row in the new bank. Even if you change the row in the same bank, the controller has to close the bank (precharge) and reopen it again just to open a new row (activate). Because of the precharge and activate commands, efficiency can decrease by as much as 3–15%, as the controller needs more time to issue a read or write.

If you must perform a random read or write, use additive latency and bank interleaving to increase efficiency.

Controller efficiency depends on the method of data transfer between the memory device and the FPGA, the memory standards specified by the memory device vendor, and the type of memory controller.

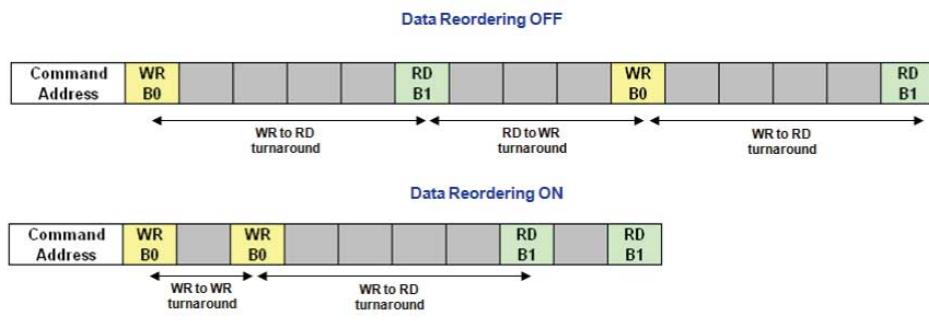
## Data Reordering

Data reordering and command reordering can both contribute towards achieving controller efficiency.

The Data Reordering feature allows the single-port memory controller to change the order of read and write commands to achieve highest efficiency. You can enable data reordering by turning on **Enable Reordering** on the **Controller Settings** tab of the parameter editor.

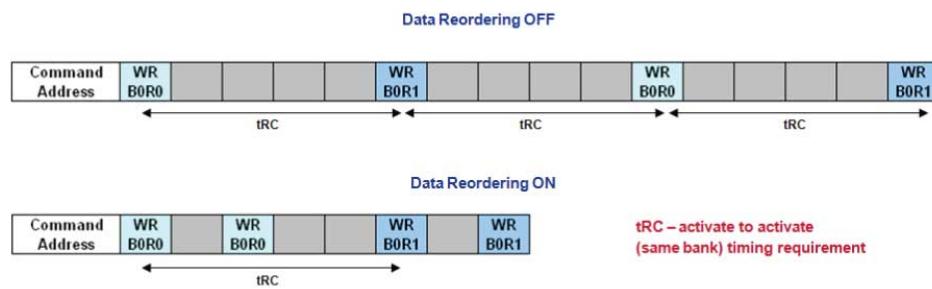
In the soft memory controller, inter-bank data reordering serves to minimize bus turnaround time by optimizing the ordering of read and write commands going to different banks; commands going to the same bank address are not reordered.

**Figure 11-12: Data Reordering for Minimum Bus Turnaround**



In the hard memory controller, inter-row data reordering serves to minimize  $t_{RC}$  by reordering commands going to different bank and row addresses; command going to the same bank and row address are not reordered. Inter-row data reordering inherits the minimum bus turnaround time benefit from inter-bank data reordering.

**Figure 11-13: Data Reordering for Minimum  $t_{RC}$**



## Starvation Control

The controller implements a starvation counter to ensure that lower-priority requests are not forgotten as higher-priority requests are reordered for efficiency.

In starvation control, a counter is incremented for every command served. You can set a starvation limit, to ensure that a waiting command is served immediately upon the starvation counter reaching the specified limit.

For example, if you set a starvation limit of 10, a lower-priority command will be treated as high priority and served immediately, after ten other commands are served before it.

## Command Reordering

Data reordering and command reordering can both contribute towards achieving controller efficiency.

DDR protocols are naturally inefficient, because commands are fetched and processed sequentially. The DDRx command and DQ bus are not fully utilized as few potential cycles are wasted and degrading the efficiency.

The command reordering feature, or look-ahead bank management feature, allows the controller to issue bank management commands early based on incoming patterns, so that when the command reaches the memory interface, the desired page in memory is already open.

The command cycles during the  $t_{RCD}$  period are idle and the bank-management commands are issued to next access banks. When the controller is serving the next command, the bank is already precharged. The command queue look-ahead depth is configurable from 1-16, to specify how many read or write requests the look-ahead bank management logic examines. With the look-ahead command queue, if consecutive write or read requests are to a sequential address with same row, same bank, and column incremental by 1, the controller merges the write or read requests at the memory transaction into a single burst.

**Figure 11-14: Comparison With and Without Look-Ahead Bank Management Feature**

Without Lookahead			With Lookahead		
Cycle	Command	Data	Cycle	Command	Data
1	ACT		1	ACT	
2	NOP		2		
3	NOP		3		
4	READ		4	READ	
5		DATA0 (Burst 0, Burst 1)	5	ACT	DATA0 (Burst 0, Burst 1)
6	NOP	DATA0 (Burst 2, Burst 3)	6	NOP	DATA0 (Burst 2, Burst 3)
7	ACT	DATA0 (Burst 4, Burst 5)	7	ACT	DATA0 (Burst 4, Burst 5)
8	NOP	DATA0 (Burst 6, Burst 7)	8	READ	DATA0 (Burst 6, Burst 7)
9	NOP	Wasted Cycle	9	NOP	DATA1 (Burst 0, Burst 1)
10	READ	Wasted Cycle	10	NOP	DATA1 (Burst 2, Burst 3)
11		DATA1 (Burst 0, Burst 1)	11	NOP	DATA1 (Burst 4, Burst 5)
12	NOP	DATA1 (Burst 2, Burst 3)	12	READ	DATA1 (Burst 6, Burst 7)
13	ACT	DATA1 (Burst 4, Burst 5)	13	NOP	DATA2 (Burst 0, Burst 1)
14	NOP	DATA1 (Burst 6, Burst 7)	14	NOP	DATA2 (Burst 2, Burst 3)
15	NOP	Wasted Cycle	15	NOP	DATA2 (Burst 4, Burst 5)
16	READ	Wasted Cycle	16	NOP	DATA2 (Burst 6, Burst 7)
17	NOP	DATA2 (Burst 0, Burst 1)			
18	NOP	DATA2 (Burst 2, Burst 3)			
19	NOP	DATA2 (Burst 4, Burst 5)			
20	NOP	DATA2 (Burst 6, Burst 7)			

Command	Address	Condition
Read	Bank 0	Activate required
Read	Bank 1	Precharge required
Read	Bank 2	Precharge required

Compare the following efficiency results for the above figure:

**Table 11-2: Efficiency Results for Above Figure**

	Without Look-ahead Bank Management	With Look-ahead Bank Management
Active cycles of data transfer	12	12
Total number of cycles	20	16
Approximate efficiency	60%	75%

In the above table, the use of look-ahead bank management increases efficiency by 15%. The bank look-ahead pattern verifies that the system is able to completely hide the bank precharge and activation for specific sequences in which the minimum number of page-open transactions are placed between transactions to closed pages to allow bank look-ahead to occur just in time for the closed pages. An optimal system would completely hide bank activation and precharge performance penalties for the bank look-ahead traffic pattern and achieve 100% efficiency, ignoring refresh.

## Bandwidth

Bandwidth depends on the efficiency of the memory controller controlling the data transfer to and from the memory device.

You can express bandwidth as follows:

$$\text{Bandwidth} = \text{data width (bits)} \times \text{data transfer rate (1/s)} \times \text{efficiency}$$

$$\text{Data rate transfer (1/s)} = 2 \times \text{frequency of operation (4 for QDR SRAM interfaces)}$$

The following example shows the bandwidth calculation for a 16-bit interface that has 70% efficiency and runs at 200 MHz frequency:

$$\text{Bandwidth} = 16 \text{ bits} \times 2 \text{ clock edges} \times 200 \text{ MHz} \times 70\% = 4.48 \text{ Gbps.}$$

DRAM typically has an efficiency of around 70%, but when you use the Altera® memory controller efficiency can vary from 10 to 92%.

In QDR II+ or QDR II SRAM the IP implements two separate unidirectional write and read data buses, so the data transfer rate is four times the clock rate. The data transfer rate for a 400-MHz interface is 1,600 Mbps. The efficiency is the percentage of time the data bus is transferring data. It is dependent on the type of memory. For example, in a QDR II+ or QDR II SRAM interface with separate write and read ports, the efficiency is 100% when there is an equal number of read and write operations on these memory interfaces.

For information on best-case and worst-case efficiency scenarios, refer to the white paper, *The Efficiency of the DDR & DDR2 SDRAM Controller Compiler*.

### Related Information

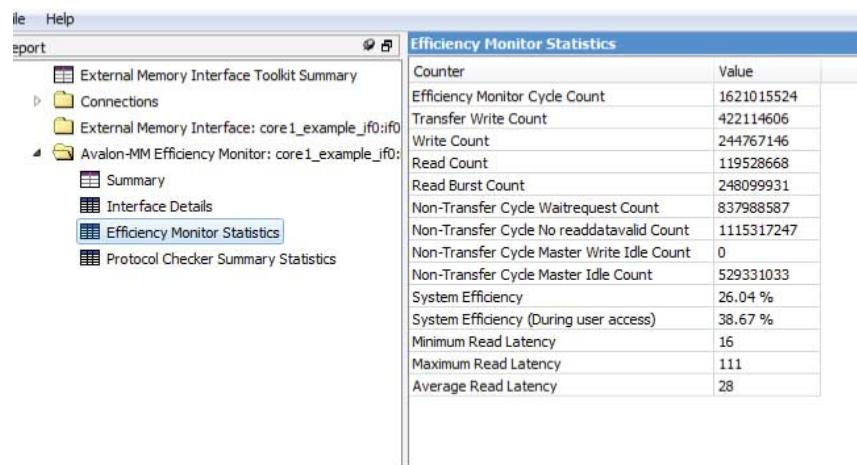
[The Efficiency of the DDR & DDR2 SDRAM Controller Compiler](#)

## Efficiency Monitor

The Efficiency Monitor and Protocol Checker measures traffic efficiency on the Avalon interface between the traffic generator and the controller, and checks that the Avalon protocol is not violated.

The Protocol Checker monitors the controller's Avalon slave interface for any illegal commands presented to it by any master; it does not monitor the legality of the controller's Avalon outputs.

**Figure 11-15: Efficiency Monitor and Protocol Checker Statistics**



The screenshot shows a software interface for the External Memory Interface Toolkit. On the left is a navigation tree with the following structure:

- External Memory Interface Toolkit Summary
- Connections
  - External Memory Interface: core1\_example\_if0:if0
  - Avalon-MM Efficiency Monitor: core1\_example\_if0:if0
    - Summary
    - Interface Details
    - Efficiency Monitor Statistics** (highlighted)
    - Protocol Checker Summary Statistics

To the right is a table titled "Efficiency Monitor Statistics" with the following data:

Counter	Value
Efficiency Monitor Cycle Count	1621015524
Transfer Write Count	422114606
Write Count	244767146
Read Count	119528668
Read Burst Count	248099931
Non-Transfer Cycle Waitrequest Count	837988587
Non-Transfer Cycle No readdatavalid Count	1115317247
Non-Transfer Cycle Master Write Idle Count	0
Non-Transfer Cycle Master Idle Count	529331033
System Efficiency	26.04 %
System Efficiency (During user access)	38.67 %
Minimum Read Latency	16
Maximum Read Latency	111
Average Read Latency	28

**Note:** To enable efficiency measurements to be performed on the controller Avalon interface through UniPHY External Memory Interface Toolkit, turn on **Enable the Efficiency Monitor and Protocol Checker on the Controller Avalon Interface**.

**Note:** For UniPHY-based designs, the efficiency monitor is not available for QDR II and QDR II+ SRAM interfaces, or for the MAX 10 device family, or for Arria V or Cyclone V designs using the Hard Memory Controller.

**Note:** The efficiency monitor does not take refreshes into account.

The Efficiency Monitor counts the number of cycles of command transfers and wait times for the controller interface and provides an Avalon slave port to allow access to this data. The efficiency monitor has an internal 32-bit counter for accessing transactions; its status can be any of the following:

- Not Running
- Not Running: Waiting for pattern start
- Running
- Not Running: Counter Saturation

For example, once the counter saturates the efficiency monitor stops because it can no longer track transactions. In the summary panel, this appears as **Not Running: Counter Saturation**.

The debug toolkit summarizes efficiency monitor statistics as follows:

- **Efficiency Monitor Cycle Count** – counts cycles from first command/start until  $2^{32}$  or a stop request
- **Transfer Count** – counts any data transfer cycle, read or write
- **Write Count** – counts how many writes requested, including those during a burst
- **Read Count** – counts how many reads requested (just commands)
- **Read Burst counter** – counts how many reads requested (total burst requests)

- **Non-Transfer Cycle Waitrequest Count** – counts Non Transfer Cycles due to slave wait request high. A Non-Transfer Cycle is a cycle during which no read data is received and no write data is accepted on the Avalon interface.
- **Non-Transfer Cycle No readdatavalid Count** – counts Non Transfer Cycles due to slave not having read data
- **Non-Transfer Cycle Master Write Idle Count** – counts Non Transfer Cycles due master not issuing command or pause in write burst
- **Non-Transfer Cycle Master Idle Count** – counts Non Transfer Cycles due master not issuing command anytime
- **System Efficiency** – The percentage of all Avalon-MM cycles where the interface is transferring data. Refreshes and idle time are not taken into consideration when calculating efficiency.
- **System Efficiency (During user access)** – Tracks the efficiency when transactions are occurring, which is a reflection on waitrequest. It is defined as: Transfer Count/(Efficiency Monitor Cycle Count - Non-Transfer Cycle Master Idle Count)
- **Minimum Read Latency** – The lowest of all read latencies, which is measured by time between a read command is being accepted by the Controller till the first beat of read data is presented to the driver.
- **Maximum Read Latency** – The highest of all read latencies, which is measured by time between a read command is being accepted by the Controller till the first beat of read data is presented to the driver.
- **Average Read Latency** – The average of all read latencies, which is measured by time between a read command is being accepted by the Controller till the first beat of read data is presented to the driver.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"><li>Added note about efficiency monitor protocol and device support, to <i>Efficiency Monitor</i>.</li></ul>
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Removed occurrence of MegaWizard Plug-In Manager.
December 2013	2013.12.16	<ul style="list-style-type: none"><li>Removed references to ALTMEMPHY.</li><li>Changed chapter number from 14 to 13.</li></ul>
November 2012	3.1	Changed chapter number from 13 to 14.
June 2012	3.0	<ul style="list-style-type: none"><li>Expanded <i>Auto-Precharge Commands</i> section.</li><li>Expanded <i>Bank Interleaving</i> section.</li><li>Added <i>Command Queue Look-Ahead Depth</i> section.</li><li>Added <i>Reordering</i> section.</li><li>Added <i>Efficiency Monitor</i> section.</li><li>Added Feedback icon.</li></ul>

Date	Version	Changes
November 2011	2.0	Reorganized optimizing the controller information into an individual chapter.
June 2011	1.0	Initial release.



# PHY Considerations 12

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The performance of your external memory interface is affected by several factors, including the data rate at which core logic and the memory interface operate, the PHY block in use, the sequencer in use, shared resources, and pin placement.

The following topics describe design considerations that affect the external memory interface performance and the device resource usage for UniPHY-based designs.

## Core Logic and User Interface Data Rate

The clocking operation in the PHY consists of the following two domains:

- PHY-memory domain—the PHY interfaces with the external memory device and is always at full-rate.
- PHY-AFI domain—the PHY interfaces with the memory controller and can either be at full, half or quarter rate of the memory clock depending on your choice of controller and PHY.

For the memory controller to operate at full, half and quarter data rate, the UniPHY IP supports full, half and quarter data rate. The data rate defines the ratio between the frequency of the Altera<sup>®</sup> PHY Interface (AFI) clock and the frequency of the memory device clock.

the following table compares the clock cycles, data bus width and address/command bus width between the full, half, and quarter-rate designs.

**Table 12-1: Ratio between Clock Cycles, Data Bus Width, and Address/Command Bus Width**

Data Rate	Controller Clock Cycles	Bus Width	
		AFI Data	AFI Address/Command
Full	1	2	1
Half	2	4	2
Quarter	4	8	4

In general, full-rate designs require smaller data and address/command bus width. However, because the core logic runs at a high frequency, full rate designs might have difficulty in closing timing. Consequently, for high frequency memory interface designs, Altera recommends that you use half-rate or quarter-rate UniPHY IP and controllers.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

DDR3 SDRAM interfaces can run at much higher frequencies than DDR, DDR2 SDRAM, QDRII, QDRII + SRAM, and RLDRAM II interfaces. For this reason, Altera High-Performance Controller II and UniPHY IPs do not support full rate designs using the DDR3 SDRAM interface. However, the DDR3 hard controller in Arria® V devices supports only full rate. Quarter-rate design support is for DDR3 SDRAM interfaces targeting frequencies higher than 667 MHz.

Although it is easier to close timing for half-rate and quarter-rate designs due to the lower frequency required on the core logic, full-rate interfaces offer better efficiency for low burst-length designs. This is because of 1T addressing mode where the address and command signals are asserted for one memory clock cycle. Typically half-rate and quarter-rate designs operate in 2T and 4T mode, respectively, in which the address and command signals in 2T and 4T mode must be asserted for two and four memory clock cycles, respectively. To improve efficiency, the controller can operate in Quasi-1T half-rate and Quasi-2T quarter-rate modes. In Quasi-1T half-rate mode, two commands are issued to the memory on two memory clock cycles. In Quasi-2T quarter-rate mode, two commands are issued to the memory on four memory clock cycles. The controller is constrained to issue a row command on the first clock phase and a column command on the second clock phase, or vice versa. Row commands include activate and precharge commands; column commands include read and write commands.

## Hard and Soft Memory PHY

The Arria V and Cyclone® V device families support hard and soft memory interfaces. Hard memory interfaces use the hard memory controllers and hard memory PHY blocks in the devices.

The hard memory PHY is instantiated together with the hard memory controller. In addition to the PHY data path that uses the hard IP blocks in the devices (similar to how the soft PHY is implemented for device families supported by UniPHY), the hard memory PHY also uses the dedicated hardware circuitries in the devices for certain component managers in the sequencer, including the read write (RW) and PHY managers.

In soft memory PHY, the UniPHY sequencer implements the Nios® II processor and all the component managers in the core logic. The hard memory PHY uses dedicated hard IP blocks in the Arria V and Cyclone V devices to implement the RW and PHY managers to save LE resources, and to allow better performance and lower latency.

Each Arria V and Cyclone V device has a fixed number of hard PHYs. Dedicated I/O pins with specific functions for data, strobe, address, command, control, and clock must be used together with each hard PHY.

For the list of hard PHY dedicated pins, refer to the device pin-out files for your target device on the *Pin-Out Files for Altera Devices* page of the Altera website.

The soft memory PHY gives you the flexibility to choose the pins to be used for the memory interface. Soft memory PHY also supports wider interfaces than hard memory PHY.

### Related Information

#### [Pin-Out Files for Altera Devices](#)

## Sequencer

The UniPHY IP soft memory PHY supports the following two types of sequencer used for QDRII and QDRII+ SRAM, and RLDRAM II calibration:

- RTL-based sequencer
- Nios II-based sequencer

The RTL-based sequencer performs FIFO calibration that includes adjusting the valid-prediction FIFO (VFIFO) and latency FIFO (LFIFO) length. In addition to the FIFO calibration, the Nios II-based sequencer also performs I/O calibration that includes adjusting delay chains and phase settings to center-align the data pins with respect to the strobes that sample them. I/O calibration is required for memory interfaces running at higher frequencies to increase read and write margins.

Because the RTL-based sequencer performs a relatively simpler calibration process, it does not require a Nios II processor. For this reason, utilization of resources such as LEs and RAMs is lower than with the Nios II-based sequencer.

For more information about the RTL-based sequencer and Nios II-based sequencer, refer to the *Functional Description—UniPHY* chapter in volume 3 of the *External Memory Interface Handbook*.

For more information about the calibration process, refer to the “UniPHY Calibration Stages” section in the *Functional Description—UniPHY* chapter of the *External Memory Interface Handbook*.

#### Related Information

##### [Functional Description - UniPHY](#)

## PLL, DLL and OCT Resource Sharing

By default, each external memory interface in a device needs one PLL, one DLL, and one OCT control block. The number of PLL, DLL and OCT resources in a device is fixed; however, these resources can be shared by two or more memory interfaces when certain criteria are met. This method allows more memory interfaces to fit into a device and allows the remaining resources to be used for other purposes.

By sharing PLLs, fewer PLLs are used, and the number of clock networks and clock input pins required is also reduced. To share PLLs, the memory interfaces must meet the following criteria:

- Run the same memory protocol (for example, DDR3 SDRAM)
- Run at the same frequency
- The controllers or PHYs run at the same rate (for example, half rate)
- Use the same phase requirements (for example, additional core-to-periphery clock phase of 90°)
- The memory interfaces are located on the same side of the device, or adjacent sides of the device if the PLL is able to drive both sides.

Altera devices have up to four DLLs available to perform phase shift on the DQS signal for capturing the read data. The DLLs are located at the device corners and some of the DLLs can access two adjacent sides of the device. To share DLLs, the memory interfaces must meet the following criteria:

- Run at the same frequency
- The memory interfaces are located on the same side of the device, or adjacent sides of the device accessible by the DLL.

Memory interface pins with OCT calibration require the OCT control block to calibrate the OCT resistance value. Depending on the device family, the OCT control block uses either the RUP and RDN, or RZQ pins for OCT calibration. Each OCT control block can only be shared by pins powered by the same VCCIO level. Sharing of the OCT control block by interfaces operating at the same VCCIO level allows other OCT control blocks in the device to support other VCCIO levels. The unused RUP/RDN or RZQ pins can also be used for other purposes. For example, the RUP/RDN pins can be used as DQ or DQS pins. To share an OCT control block, the memory interfaces must operate at the same VCCIO level.

For more information about the resources required for memory interfaces in various device families, refer to the *Planning Pin and FPGA Resources* chapter.

For more information about how to share PLL, DLL and OCT control blocks, refer to the *Functional Description—UniPHY* chapter in volume 3 of the *External Memory Interface Handbook*.

For more information about the DLL, refer to the external memory interface chapters in the respective device handbooks.

For more information about the OCT control block, refer to the I/O features chapters in the respective device handbooks.

#### Related Information

- [Functional Description - UniPHY](#)
- [Planning Pin and FPGA Resources](#) on page 1-1

## Pin Placement Consideration

The Stratix® V, Arria® V, Cyclone® V, and MAX® 10 device families use the PHY clock (PHYCLK) networks to clock the external memory interface pins for better performance.

Each PHYCLK network is driven by a PLL.

- In Cyclone V and Stratix V devices, the PHYCLK network spans across two I/O banks on the same side of the device.
- For Arria V devices, each PHYCLK network spans across one I/O bank.
- For MAX 10 devices, the PHYCLK network is available only for the I/O banks on the right side of the device.

All pins for a memory interface must be placed on the same side of the device.

For more information about pin placement guidelines related to the PHYCLK network, refer to the following documents:

- *External Memory Interfaces in Stratix V Devices* chapter in volume 2 of the *Stratix V Device Handbook*
- *External Memory Interfaces in Arria V Devices* chapter in volume 2 of the *Arria V Device Handbook*
- *External Memory Interfaces in Cyclone V Devices* chapter in volume 2 of the *Cyclone V Device Handbook*
- *MAX 10 External Memory Interface Architecture and Features* chapter of the *MAX 10 External Memory Interface User Guide*

Certain device families do not use the PHYCLK network to allow greater flexibility in pin placement. These device families support the following interface types:

- Wraparound interfaces, in which data pins from a memory interface are placed on two adjacent sides of a device.
- Split interfaces, in which data pins are place on two opposite I/O banks.

The x36 emulated mode is supported in certain device families that do not use the PHY clock network for QDRII and QDRII+ SRAM x36 interfaces. In x36 emulated mode, two x18 DQS groups or four x9 DQS groups can be combined to form a 36-bit wide write data bus. Also, two x18 DQS groups can be combined to form a 36-bit wide read data bus. This method allows a device to support x36 QDRII and QDRII+ SRAM interfaces even if the device does not have the required number of x36 DQS groups.

Some device families might support wraparound or x36 emulated mode interfaces at slightly lower frequencies.

For information about the devices that support wraparound and x36 emulated mode interfaces, and the supported frequency for your design, refer to the External Memory Interface Spec Estimator page on the Altera website.

For more information about x36 emulated mode support for QDRII and QDRII+ SRAM interfaces, refer to the *Planning Pin and FPGA Resources* chapter.

#### Related Information

- [Planning Pin and FPGA Resources](#) on page 1-1
- [External Memory Interfaces in Stratix V Devices](#)
- [External Memory Interfaces in Arria V Devices](#)
- [External Memory Interfaces in Cyclone V Devices](#)
- [MAX 10 PHY Clock \(PHYCLK\) Network, MAX 10 External Memory Interface Architecture](#)
- [External Memory Interface Spec Estimator](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.01	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Updated the <i>Pin Placement Consideration</i> topic to include MAX 10 devices.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Maintenance release.
December 2013	2013.12.16	Changed chapter number from 15 to 14.
November 2012	3.1	Changed chapter number from 14 to 15.
June 2012	3.0	Added Feedback icon.
November 2011	1.0	Initial release.

# Power Estimation Methods for External Memory Interfaces **13**

2016.05.02

EMI\_DG



Subscribe



Send Feedback

The following table lists the Altera® -supported power estimation methods for external memory interfaces.

**Table 13-1: Power Estimation Methods for External Memory Interfaces**

Method	Vector Source	UniPHY Support	Accuracy	Estimation Time <sup>(1)</sup>
Early power estimator (EPE)	Not applicable	v	Lowest	Fastest
Vector-less PowerPlay power analysis (PPPA)	Not applicable	v		
Vector-based PPPA	RTL simulation	v		
	Zero-delay simulation <sup>(2)</sup>	v		
	Timing simulation	(2)		
			Highest	Slowest

Notes to Table:

1. To decrease the estimation time, you can skip power estimation during calibration. Power consumption during calibration is typically equivalent to power consumption during user mode.
2. Power analysis using timing simulation vectors is not supported.

When using Altera IP, you can use the zero-delay simulation method to analyze the power required for the external memory interface. Zero-delay simulation is as accurate as timing simulation for 95% designs (designs with no glitching). For a design with glitching, power may be under estimated.

For more information about zero-delay simulation, refer to the *Power Estimation and Analysis* section in the *Quartus® Prime Handbook*.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

**Note:** The size of the vector file (.vcd) generated by zero-delay simulation of an Altera DDR3 SDRAM High-Performance Controller example design is 400 GB. The .vcd includes calibration and user mode activities. When vector generation of calibration phase is skipped, the vector size decreases to 1 GB.

#### Related Information

[Quartus Prime Handbook](#)

## Performing Vector-Based Power Analysis with the PowerPlay Power Analyzer

To perform vector-based power analysis with the PowerPlay Power Analyzer using zero-delay simulation, follow these steps:

1. Compile your design in the Quartus Prime software to generate a Netlist <project\_name>.vo file for your design.

**Note:**

2. The <project\_name>.vo is generated in the last stage of a compile EDA Netlist Writer.
3. Open the <project\_name>.vo file in a text editor.
4. In the <project\_name>.vo file, locate the include statement for <project\_name>.sdo, and comment-out that include statement. Save the <project\_name>.vo file.
5. Create a simulation script containing device model files and libraries and design specific files:
  - Netlist file for the design, <project\_name>.vo
  - RTL or netlist file for the memory device
  - Testbench RTL file
6. Compile all the files.
7. Invoke the simulator with commands to generate .vcd files.
8. Generate .vcd files for the parts of the design that contribute the most to power dissipation.
9. Run simulation.
10. Use the generated .vcd files in the PowerPlay Power Analyzer as the signal activity input file.
11. Run the PowerPlay Power Analyzer.

**Note:** For more information about estimating power, refer to the *Power Estimation and Analysis* section in the *Quartus Prime Handbook*.

#### Related Information

[Quartus Prime Handbook](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
May 2015	2015.05.04	Maintenance release.

Date	Version	Changes
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Maintenance release.
December 2013	2013.12.16	<ul style="list-style-type: none"><li>Removed references to ALTMEMPHY.</li><li>Changed chapter number from 16 to 15.</li></ul>
November 2012	2.2	Changed chapter number from 15 to 16.
June 2012	2.1	Added Feedback icon.
November 2011	2.0	Reorganized power estimation methods section into an individual chapter.
April 2010	1.0	Initial release.

# External Memory Interface Handbook

**Volume 3: Reference Material**

# Contents

<b>Functional Description—UniPHY.....</b>	<b>1-1</b>
I/O Pads.....	1-2
Reset and Clock Generation.....	1-3
Dedicated Clock Networks.....	1-3
Address and Command Datapath.....	1-4
Write Datapath.....	1-5
Leveling Circuitry.....	1-5
Read Datapath.....	1-7
Sequencer.....	1-8
Nios II-Based Sequencer.....	1-9
RTL-based Sequencer.....	1-13
Shadow Registers.....	1-15
Shadow Registers Operation.....	1-17
UniPHY Interfaces.....	1-17
The DLL and PLL Sharing Interface.....	1-18
The OCT Sharing Interface.....	1-19
UniPHY Signals.....	1-21
PHY-to-Controller Interfaces.....	1-25
Using a Custom Controller.....	1-30
AFI 3.0 Specification.....	1-31
Bus Width and AFI Ratio.....	1-32
AFI Parameters.....	1-32
AFI Signals.....	1-34
Register Maps.....	1-39
UniPHY Register Map.....	1-40
Controller Register Map.....	1-43
Ping Pong PHY.....	1-44
Ping Pong PHY Feature Description.....	1-44
Ping Pong PHY Architecture.....	1-45
Ping Pong PHY Operation.....	1-47
Efficiency Monitor and Protocol Checker.....	1-48
Efficiency Monitor.....	1-48
Protocol Checker.....	1-48
Read Latency Counter.....	1-48
Using the Efficiency Monitor and Protocol Checker.....	1-48
Avalon CSR Slave and JTAG Memory Map.....	1-49
UniPHY Calibration Stages.....	1-51
Calibration Overview.....	1-52
Calibration Stages.....	1-52
Memory Initialization.....	1-53
Stage 1: Read Calibration Part One—DQS Enable Calibration and DQ/DQS Centering...1-53	
Stage 2: Write Calibration Part One.....	1-57

Stage 3: Write Calibration Part Two—DQ/DQS Centering.....	1-59
Stage 4: Read Calibration Part Two—Read Latency Minimization.....	1-59
Calibration Signals.....	1-59
Calibration Time.....	1-59
Document Revision History.....	1-60

## Functional Description—Arria 10 EMIF.....2-1

Supported Memory Protocols.....	2-2
Key Differences Compared to UniPHY IP and Previous Device Families.....	2-2
Migrating from Previous Device Families.....	2-3
Arria 10 EMIF Architecture: Introduction.....	2-3
Arria 10 EMIF Architecture: I/O Subsystem.....	2-4
Arria 10 EMIF Architecture: I/O Column.....	2-5
Arria 10 EMIF Architecture: I/O AUX.....	2-6
Arria 10 EMIF Architecture: I/O Bank.....	2-6
Arria 10 EMIF Architecture: I/O Lane.....	2-10
Arria 10 EMIF Architecture: Input DQS Clock Tree.....	2-12
Arria 10 EMIF Architecture: PHY Clock Tree.....	2-13
Arria 10 EMIF Architecture: PLL Reference Clock Networks.....	2-13
Arria 10 EMIF Architecture: Clock Phase Alignment.....	2-14
Hardware Resource Sharing Among Multiple EMIFs.....	2-15
I/O Aux Sharing.....	2-16
I/O Bank Sharing.....	2-16
PLL Reference Clock Sharing.....	2-17
Core Clock Network Sharing.....	2-18
Arria 10 EMIF IP Component.....	2-18
Instantiating Your Arria 10 EMIF IP in a Qsys Project.....	2-19
File Sets.....	2-22
Customized readme.txt File.....	2-23
Clock Domains.....	2-23
ECC in Arria 10 EMIF IP.....	2-24
Examples of External Memory Interface Implementations for DDR4.....	2-25
Arria 10 EMIF Sequencer.....	2-30
Arria 10 EMIF DQS Tracking.....	2-31
Arria 10 EMIF Calibration.....	2-31
Calibration Stages .....	2-31
Calibration Stages Descriptions.....	2-32
Calibration Algorithms.....	2-33
Calibration Flowchart.....	2-35
Periodic OCT Recalibration.....	2-35
Arria 10 EMIF and SmartVID.....	2-37
Hard Memory Controller Rate Conversion Feature.....	2-38
Compiling Arria 10 EMIF IP with the Quartus Prime Software.....	2-39
Instantiating the Arria 10 EMIF IP.....	2-39
Setting I/O Assignments in Arria 10 EMIF IP.....	2-39
Debugging Arria 10 EMIF IP.....	2-40
External Memory Interface Debug Toolkit.....	2-40
On-Chip Debug for Arria 10.....	2-40

Configuring Your EMIF IP for Use with the Debug Toolkit.....	2-40
Arria 10 EMIF Debugging Examples.....	2-41
Arria 10 EMIF for Hard Processor Subsystem.....	2-43
Restrictions on I/O Bank Usage for Arria 10 EMIF IP with HPS.....	2-44
Arria 10 EMIF Ping Pong PHY.....	2-46
Ping Pong PHY Feature Description.....	2-46
Ping Pong PHY Architecture.....	2-48
Ping Pong PHY Limitations.....	2-49
Ping Pong PHY Calibration.....	2-50
Using the Ping Pong PHY.....	2-51
Ping Pong PHY Simulation Example Design.....	2-51
AFI 4.0 Specification.....	2-52
Bus Width and AFI Ratio.....	2-52
AFI Parameters.....	2-52
AFI Signals.....	2-55
AFI 4.0 Timing Diagrams.....	2-61
Resource Utilization.....	2-75
QDR-IV Resource Utilization in Arria 10 Devices.....	2-75
Arria 10 EMIF Latency.....	2-75
Arria 10 EMIF Calibration Times.....	2-76
Integrating a Custom Controller with the Hard PHY.....	2-77
Memory Mapped Register (MMR) Tables.....	2-77
ctrlcfg0: Controller Configuration.....	2-80
ctrlcfg1: Controller Configuration.....	2-81
ctrlcfg2: Controller Configuration.....	2-83
ctrlcfg3: Controller Configuration.....	2-84
ctrlcfg4: Controller Configuration.....	2-85
ctrlcfg5: Controller Configuration.....	2-87
ctrlcfg6: Controller Configuration.....	2-87
ctrlcfg7: Controller Configuration.....	2-88
ctrlcfg8: Controller Configuration.....	2-88
ctrlcfg9: Controller Configuration.....	2-88
dramtiming0: Timing Parameters.....	2-89
dramodt0: On-Die Termination Parameters.....	2-89
dramodt1: On-Die Termination Parameters.....	2-90
sbcfg0: Sideband Configuration.....	2-90
sbcfg1: Sideband Configuration.....	2-91
sbcfg2: Sideband Configuration.....	2-91
sbcfg3: Sideband Configuration.....	2-91
sbcfg4: Sideband Configuration.....	2-92
sbcfg5: Sideband Configuration.....	2-92
sbcfg6: Sideband Configuration.....	2-92
sbcfg7: Sideband Configuration.....	2-92
sbcfg8: Sideband Configuration.....	2-93
sbcfg9: Sideband Configuration.....	2-93
caltiming0: Command/Address/Latency Parameters.....	2-93
caltiming1: Command/Address/Latency Parameters.....	2-94
caltiming2: Command/Address/Latency Parameters.....	2-94
caltiming3: Command/Address/Latency Parameters.....	2-94

caltiming4: Command/Address/Latency Parameters.....	2-95
caltiming5: Command/Address/Latency Parameters.....	2-95
caltiming6: Command/Address/Latency Parameters.....	2-96
caltiming7: Command/Address/Latency Parameters.....	2-96
caltiming8: Command/Address/Latency Parameters.....	2-96
caltiming9: Command/Address/Latency Parameters.....	2-97
caltiming10: Command/Address/Latency Parameters.....	2-97
dramaddrw: Row/Column/Bank Address Width Configuration.....	2-97
sideband0: Sideband.....	2-98
sideband1: Sideband.....	2-98
sideband2: Sideband.....	2-98
sideband3: Sideband.....	2-98
sideband4: Sideband.....	2-99
sideband5: Sideband.....	2-99
sideband6: Sideband.....	2-99
sideband7: Sideband.....	2-99
sideband8: Sideband.....	2-99
sideband9: Sideband.....	2-100
sideband10: Sideband.....	2-100
sideband11: Sideband.....	2-100
sideband12: Sideband.....	2-100
sideband13: Sideband.....	2-101
sideband14: Sideband.....	2-101
sideband15: Sideband.....	2-101
dramsts: Calibration Status.....	2-101
ecc1: ECC General Configuration.....	2-102
ecc2: Width Configuration.....	2-102
ecc3: ECC Error and Interrupt Configuration.....	2-103
ecc4: Status and Error Information.....	2-104
Document Revision History.....	2-105

<b>Functional Description—MAX 10 EMIF.....</b>	<b>3-1</b>
MAX 10 EMIF Overview.....	3-1
External Memory Protocol Support.....	3-2
MAX 10 Memory Controller.....	3-2
MAX 10 Low Power Feature.....	3-3
MAX 10 Memory PHY.....	3-3
Supported Topologies.....	3-3
Read Datapath.....	3-3
Write Datapath.....	3-4
Address and Command Datapath.....	3-6
Sequencer.....	3-7
Calibration.....	3-9
Read Calibration.....	3-10
Write Calibration.....	3-10
Sequencer Debug Information.....	3-10
Register Maps.....	3-12
Document Revision History.....	3-12

<b>Functional Description—Hard Memory Interface.....</b>	<b>4-1</b>
Multi-Port Front End (MPFE).....	4-2
Multi-port Scheduling.....	4-3
Port Scheduling.....	4-3
DRAM Burst Scheduling.....	4-3
DRAM Power Saving Modes.....	4-4
MPFE Signal Descriptions.....	4-4
Hard Memory Controller.....	4-7
Clocking.....	4-8
Reset.....	4-8
DRAM Interface.....	4-8
ECC.....	4-8
Bonding of Memory Controllers.....	4-9
Hard PHY.....	4-10
Interconnections.....	4-10
Clock Domains.....	4-11
Hard Sequencer.....	4-11
MPFE Setup Guidelines.....	4-11
Soft Memory Interface to Hard Memory Interface Migration Guidelines.....	4-12
Bonding Interface Guidelines.....	4-13
Document Revision History.....	4-13

<b>Functional Description—HPS Memory Controller.....</b>	<b>5-1</b>
Features of the SDRAM Controller Subsystem.....	5-1
SDRAM Controller Subsystem Block Diagram.....	5-2
SDRAM Controller Memory Options.....	5-3
SDRAM Controller Subsystem Interfaces.....	5-4
MPU Subsystem Interface.....	5-4
L3 Interconnect Interface.....	5-4
CSR Interface.....	5-5
FPGA-to-HPS SDRAM Interface.....	5-5
Memory Controller Architecture.....	5-6
Multi-Port Front End.....	5-7
Single-Port Controller.....	5-8
Functional Description of the SDRAM Controller Subsystem.....	5-10
MPFE Operation Ordering.....	5-10
MPFE Multi-Port Arbitration.....	5-10
MPFE SDRAM Burst Scheduling.....	5-13
Single-Port Controller Operation.....	5-14
SDRAM Power Management.....	5-24
DDR PHY.....	5-25
Clocks.....	5-25
Resets.....	5-26
Taking the SDRAM Controller Subsystem Out of Reset .....	5-26
Port Mappings.....	5-26
Initialization.....	5-27

FPGA-to-SDRAM Protocol Details.....	5-27
SDRAM Controller Subsystem Programming Model.....	5-32
HPS Memory Interface Architecture.....	5-32
HPS Memory Interface Configuration.....	5-32
HPS Memory Interface Simulation.....	5-32
Generating a Preloader Image for HPS with EMIF.....	5-33
Debugging HPS SDRAM in the Preloader.....	5-34
Enabling UART or Semihosting Printout.....	5-34
Enabling Simple Memory Test.....	5-35
Enabling the Debug Report.....	5-37
Writing a Predefined Data Pattern to SDRAM in the Preloader.....	5-40
SDRAM Controller Address Map and Register Definitions.....	5-41
SDRAM Controller Address Map.....	5-41
Document Revision History.....	5-73

## **Functional Description—HPC II Controller.....6-1**

HPC II Memory Interface Architecture.....	6-1
HPC II Memory Controller Architecture.....	6-2
Backpressure Support.....	6-4
Command Generator.....	6-5
Timing Bank Pool.....	6-5
Arbiter.....	6-5
Rank Timer.....	6-5
Read Data Buffer and Write Data Buffer.....	6-5
ECC Block.....	6-5
AFI and CSR Interfaces.....	6-6
HPC II Controller Features.....	6-6
Data Reordering.....	6-6
Pre-emptive Bank Management.....	6-6
Quasi-1T and Quasi-2T.....	6-6
User Autoprecharge Commands.....	6-7
Address and Command Decoding Logic.....	6-7
Low-Power Logic.....	6-7
ODT Generation Logic.....	6-7
Burst Merging.....	6-10
ECC.....	6-10
External Interfaces.....	6-13
Clock and Reset Interface.....	6-13
Avalon-ST Data Slave Interface.....	6-13
AXI Data Slave Interface.....	6-13
Controller-PHY Interface.....	6-21
Memory Side-Band Signals.....	6-21
Controller External Interfaces.....	6-22
Top-Level Signals Description.....	6-23
Clock and Reset Signals.....	6-23
Local Interface Signals.....	6-25
Controller Interface Signals.....	6-32
CSR Interface Signals.....	6-34

Soft Controller Register Map.....	6-35
Hard Controller Register Map.....	6-42
Sequence of Operations.....	6-46
Document Revision History.....	6-47
<b>Functional Description—QDR II Controller.....</b>	<b>7-1</b>
Block Description.....	7-1
Avalon-MM Slave Read and Write Interfaces.....	7-1
Command Issuing FSM.....	7-2
AFI.....	7-2
Avalon-MM and Memory Data Width.....	7-2
Signal Descriptions.....	7-3
Document Revision History.....	7-4
<b>Functional Description—QDR-IV Controller.....</b>	<b>8-1</b>
Block Description.....	8-1
Avalon-MM Slave Read and Write Interfaces.....	8-1
AFI.....	8-2
Avalon-MM and Memory Data Width.....	8-3
Signal Descriptions.....	8-3
Document Revision History.....	8-4
<b>Functional Description—RLDRAM II Controller.....</b>	<b>9-1</b>
Block Description.....	9-1
Avalon-MM Slave Interface.....	9-1
Write Data FIFO Buffer.....	9-2
Command Issuing FSM.....	9-2
Refresh Timer.....	9-2
Timer Module.....	9-2
AFI.....	9-2
User-Controlled Features.....	9-3
Error Detection Parity.....	9-3
User-Controlled Refresh.....	9-3
Avalon-MM and Memory Data Width.....	9-3
Signal Descriptions.....	9-3
Document Revision History.....	9-4
<b>Functional Description—RLDRAM 3 PHY-Only IP.....</b>	<b>10-1</b>
Block Description.....	10-1
Features.....	10-1
RLDRAM 3 AFI Protocol.....	10-2
RLDRAM 3 Controller with Arria 10 EMIF Interfaces.....	10-3
Document Revision History.....	10-5
<b>Functional Description—Example Designs.....</b>	<b>11-1</b>

Arria 10 EMIF IP Example Designs Quick Start Guide.....	11-1
Typical Example Design Workflow.....	11-1
Example Designs Interface Tab.....	11-3
Development Kit Preset Workflow.....	11-4
Compiling and Simulating the Design.....	11-5
Compiling and Testing the Design in Hardware.....	11-7
Configuring the Traffic Generator 2.0.....	11-8
EMIF Configurable Traffic Generator 2.0 Reference.....	11-9
Configurable Traffic Generator Parameters.....	11-10
Configurable Traffic Generator 2.0 Configuration Options.....	11-10
Performing Your Own Tests Using Traffic Generator 2.0.....	11-16
UniPHY-Based Example Designs.....	11-18
Synthesis Example Design.....	11-18
Simulation Example Design.....	11-19
Traffic Generator and BIST Engine.....	11-21
Creating and Connecting the UniPHY Memory Interface and the Traffic Generator in Qsys.....	11-25
Document Revision History.....	11-27

## **Introduction to UniPHY IP..... 12-1**

Release Information.....	12-1
Device Support Levels.....	12-2
Device Family and Protocol Support.....	12-2
UniPHY-Based External Memory Interface Features.....	12-3
System Requirements.....	12-5
MegaCore Verification.....	12-6
Resource Utilization.....	12-6
DDR2, DDR3, and LPDDR2 Resource Utilization in Arria V Devices.....	12-6
DDR2 and DDR3 Resource Utilization in Arria II GZ Devices.....	12-7
DDR2 and DDR3 Resource Utilization in Stratix III Devices.....	12-9
DDR2 and DDR3 Resource Utilization in Stratix IV Devices.....	12-11
DDR2 and DDR3 Resource Utilization in Arria V GZ and Stratix V Devices.....	12-13
QDR II and QDR II+ Resource Utilization in Arria V Devices.....	12-16
QDR II and QDR II+ Resource Utilization in Arria II GX Devices.....	12-17
QDR II and QDR II+ Resource Utilization in Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices.....	12-17
RLDRAM II Resource Utilization in Arria V Devices.....	12-18
RLDRAM II Resource Utilization in Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices.....	12-18
Document Revision History.....	12-19

## **Latency for UniPHY IP..... 13-1**

DDR2 SDRAM LATENCY.....	13-1
DDR3 SDRAM LATENCY.....	13-2
LPDDR2 SDRAM LATENCY.....	13-3
QDR II and QDR II+ SRAM Latency.....	13-4
RLDRAM II Latency.....	13-5

RLDRAM 3 Latency.....	13-5
Variable Controller Latency.....	13-6
Document Revision History.....	13-6
<b>Timing Diagrams for UniPHY IP.....</b>	<b>14-1</b>
DDR2 Timing Diagrams.....	14-1
DDR3 Timing Diagrams.....	14-7
QDR II and QDR II+ Timing Diagrams.....	14-15
RLDRAM II Timing Diagrams.....	14-19
LPDDR2 Timing Diagrams.....	14-24
RLDRAM 3 Timing Diagrams.....	14-30
Document Revision History.....	14-34
<b>External Memory Interface Debug Toolkit.....</b>	<b>15-1</b>
User Interface.....	15-1
Communication.....	15-1
Calibration and Report Generation.....	15-2
Setup and Use.....	15-2
General Workflow.....	15-3
Linking the Project to a Device.....	15-3
Establishing Communication to Connections.....	15-4
Selecting an Active Interface.....	15-4
Reports.....	15-4
Operational Considerations.....	15-6
Troubleshooting.....	15-7
Debug Report for Arria V and Cyclone V SoC Devices.....	15-7
Enabling the Debug Report for Arria V and Cyclone V SoC Devices.....	15-7
Determining the Failing Calibration Stage for a Cyclone V or Arria V HPS SDRAM Controller.....	15-8
On-Chip Debug Port for UniPHY-based EMIF IP.....	15-8
Access Protocol.....	15-9
Command Codes Reference.....	15-10
Header Files.....	15-11
Generating IP With the Debug Port.....	15-11
Example C Code for Accessing Debug Data.....	15-12
On-Chip Debug Port for Arria 10 EMIF IP.....	15-13
Access Protocol.....	15-14
EMIF On-Chip Debug Port.....	15-15
On-Die Termination Calibration.....	15-15
Eye Diagram .....	15-16
Driver Margining for Arria 10 EMIF IP.....	15-16
Determining Margin.....	15-16
Read Setting and Apply Setting Commands for Arria 10 EMIF IP.....	15-17
Reading or Applying Calibration Settings.....	15-17
Configuring the Traffic Generator 2.0.....	15-17
The Traffic Generator 2.0 Report.....	15-19
Example Tcl Script for Running the EMIF Debug Toolkit.....	15-19

Document Revision History.....	15-20
--------------------------------	-------

<b>Upgrading to UniPHY-based Controllers from ALTMEMPHY-based Controllers.....</b>	<b>16-1</b>
Generating Equivalent Design.....	16-1
Replacing the ALTMEMPHY Datapath with UniPHY Datapath.....	16-2
Resolving Port Name Differences.....	16-2
Creating OCT Signals.....	16-4
Running Pin Assignments Script.....	16-4
Removing Obsolete Files.....	16-4
Simulating your Design.....	16-4
Document Revision History.....	16-6

# Functional Description—UniPHY

1

2016.05.02

EMI\_RM



Subscribe



Send Feedback

UniPHY is the physical layer of the external memory interface.

The major functional units of the UniPHY layer include the following:

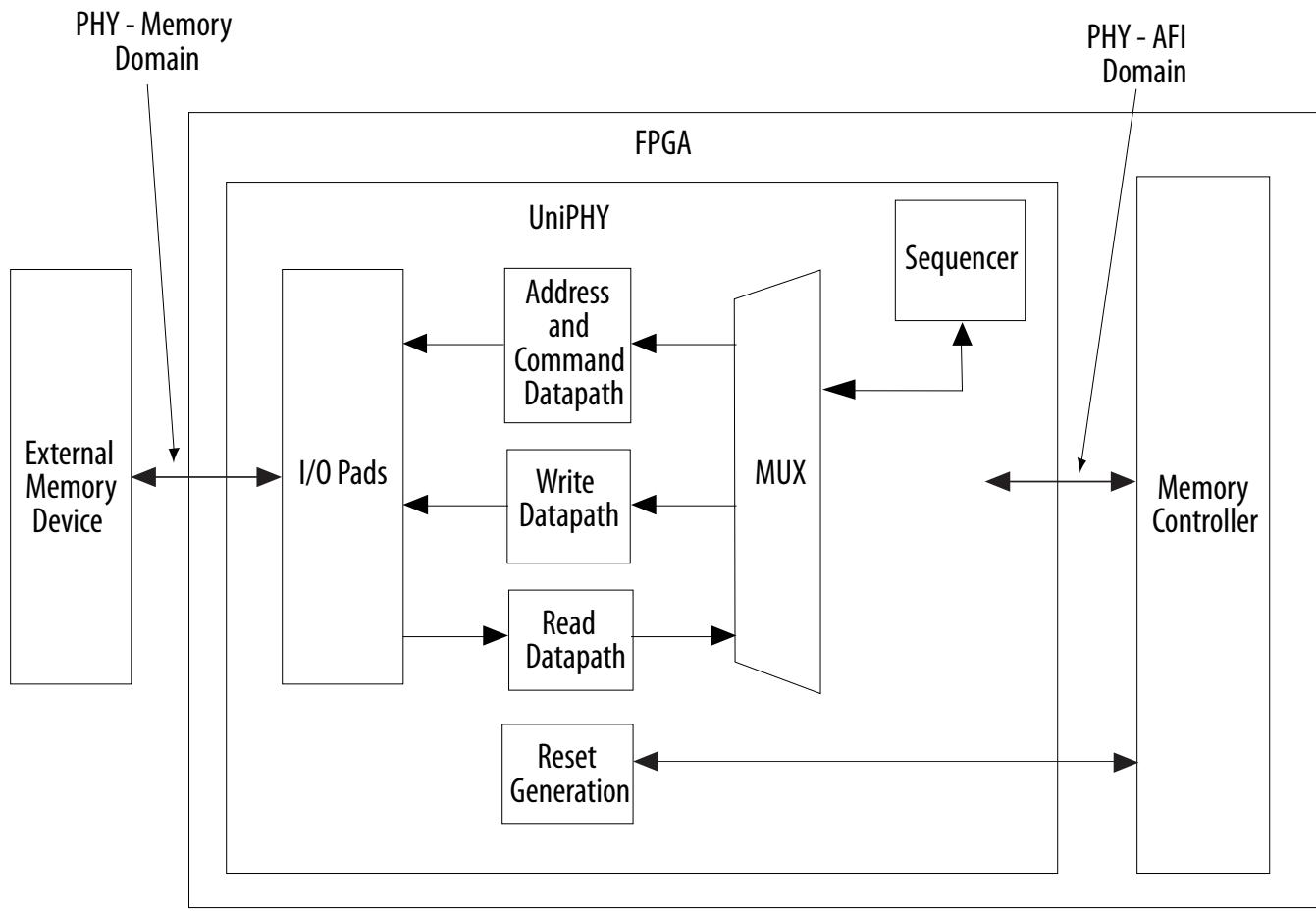
- Reset and clock generation
- Address and command datapath
- Write datapath
- Read datapath
- Sequencer

The following figure shows the PHY block diagram.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

Figure 1-1: PHY Block Diagram



### Related Information

- [UniPHY Block Description](#)
- [UniPHY Interfaces](#) on page 1-17
- [UniPHY Signals](#) on page 1-21
- [PHY-to-Controller Interfaces](#) on page 1-25
- [Using a Custom Controller](#) on page 1-30
- [Using a Vendor-Specific Memory Model](#)
- [AFI 4.0 Specification](#) on page 2-52
- [Register Maps](#) on page 1-39
- [Ping Pong PHY](#) on page 1-44
- [Efficiency Monitor](#) on page 1-48
- [Calibration Stages](#) on page 1-52

## I/O Pads

The I/O pads contain all the I/O instantiations.

## Reset and Clock Generation

At a high level, clocks in the PHY can be classified into two domains: the PHY-memory domain and the PHY-AFI domain.

The PHY-memory domain interfaces with the external memory device and always operate at full-rate. The PHY-AFI domain interfaces with the memory controller and can be a full-rate, half-rate, or quarter-rate clock, based on the controller in use.

The number of clock domains in a memory interface can vary depending on its configuration; for example:

- At the PHY-memory boundary, separate clocks may exist to generate the memory clock signal, the output strobe, and to output write data, as well as address and command signals. These clocks include `p11_dq_write_clk`, `p11_write_clk`, `p11_mem_clk`, and `p11_addr_cmd_clk`. These clocks are phase-shifted as required to achieve the desired timing relationships between memory clock, address and command signals, output data, and output strobe.
- For quarter-rate interfaces, additional clock domains such as `p11_hr_clock` are required to convert signals between half-rate and quarter-rate.
- For high-performance memory interfaces using Arria V, Cyclone V, or Stratix V devices, additional clocks may be required to handle transfers between the device core and the I/O periphery for timing closure. For core-to-periphery transfers, the latch clock is `p11_c2p_write_clock`; for periphery-to-core transfers, it is `p11_p2c_read_clock`. These clocks are automatically phase-adjusted for timing closure during IP generation, but can be further adjusted in the parameter editor. If the phases of these clocks are zero, the Fitter may remove these clocks during optimization. Also, high-performance interfaces using a Nios II-based sequencer require two additional clocks, `p11_avl_clock` for the Nios II processor, and `p11_config_clock` for clocking the I/O scan chains during calibration.

For a complete list of clocks in your memory interface, compile your design and run the **Report Clocks** command in the TimeQuest Timing Analyzer.

## Dedicated Clock Networks

The UniPHY layer employs three types of dedicated clock networks:

- Global clock network
- Dual-regional clock network
- PHY clock network (applicable to Arria V, Cyclone V, and Stratix V devices, and later)

The PHY clock network is a dedicated high-speed, low-skew, balanced clock tree designed for high-performance external memory interface. For device families that support the PHY clock network, UniPHY always uses the PHY clock network for all clocks at the PHY-memory boundary.

For families that do not support the PHY clock network, UniPHY uses either dual-regional or global clock networks for clocks at the PHY-memory boundary. During generation, the system selects dual-regional or global clocks automatically, depending on whether a given interface spans more than one quadrant. UniPHY does not mix the usage of dual-regional and global clock networks for clocks at the PHY-memory boundary; this ensures that timing characteristics of the various output paths are as similar as possible.

The `<variation_name>.pin_assignments.tcl` script creates the appropriate clock network type assignment. The use of the PHY clock network is specified directly in the RTL code, and does not require an assignment.

The UniPHY uses an active-low, asynchronous assert and synchronous de-assert reset scheme. The global reset signal resets the PLL in the PHY and the rest of the system is held in reset until after the PLL is locked.

## Address and Command Datapath

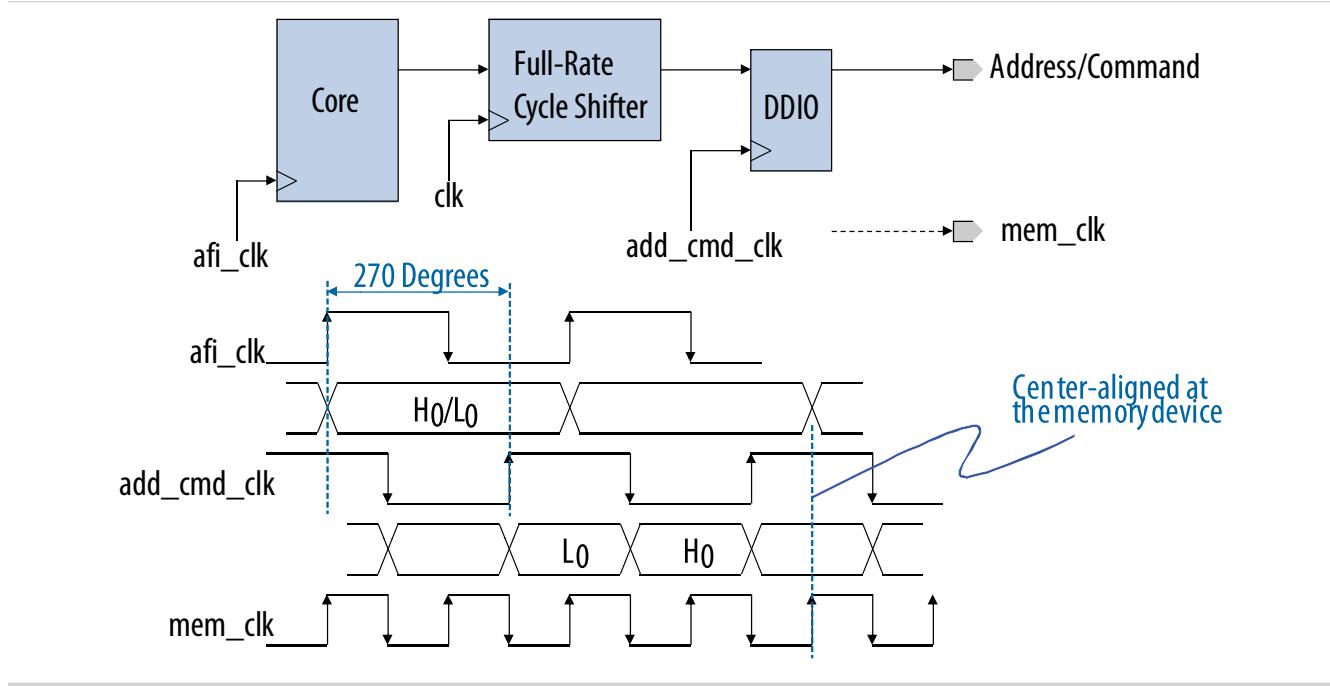
The memory controller controls the read and write addresses and commands to meet the memory specifications.

The PHY is indifferent to address or command—that is, it performs no decoding or other operations—and the circuitry is the same for both. In full-rate and half-rate interfaces, address and command is full rate, while in quarter-rate interfaces, address and command is half rate.

Address and command signals are generated in the Altera PHY interface (AFI) clock domain and sent to the memory device in the address and command clock domain. The double-data rate input/output (DDIO) stage converts the half-rate signals into full-rate signals, when the AFI clock runs at half-rate. For quarter-rate interfaces, additional DDIO stages exist to convert the address and command signals in the quarter-rate AFI clock domain to half-rate.

The address and command clock is offset with respect to the memory clock to balance the nominal setup and hold margins at the memory device (center-alignment). In the example below, this offset is 270 degrees. The Fitter can further optimize margins based on the actual delays and clock skews. In half-rate and quarter-rate designs, the full-rate cycle shifter blocks can perform a shift measured in full-rate cycles to implement the correct write latency; without this logic, the controller would only be able to implement even write latencies as it operates at half the speed. The full-rate cycle shifter is clocked by either the AFI clock or the address and command clock, depending on the PHY configuration, to maximize timing margins on the path from the AFI clock to the address and command clock.

**Figure 1-2: Address and Command Datapath (Half-rate example shown)**

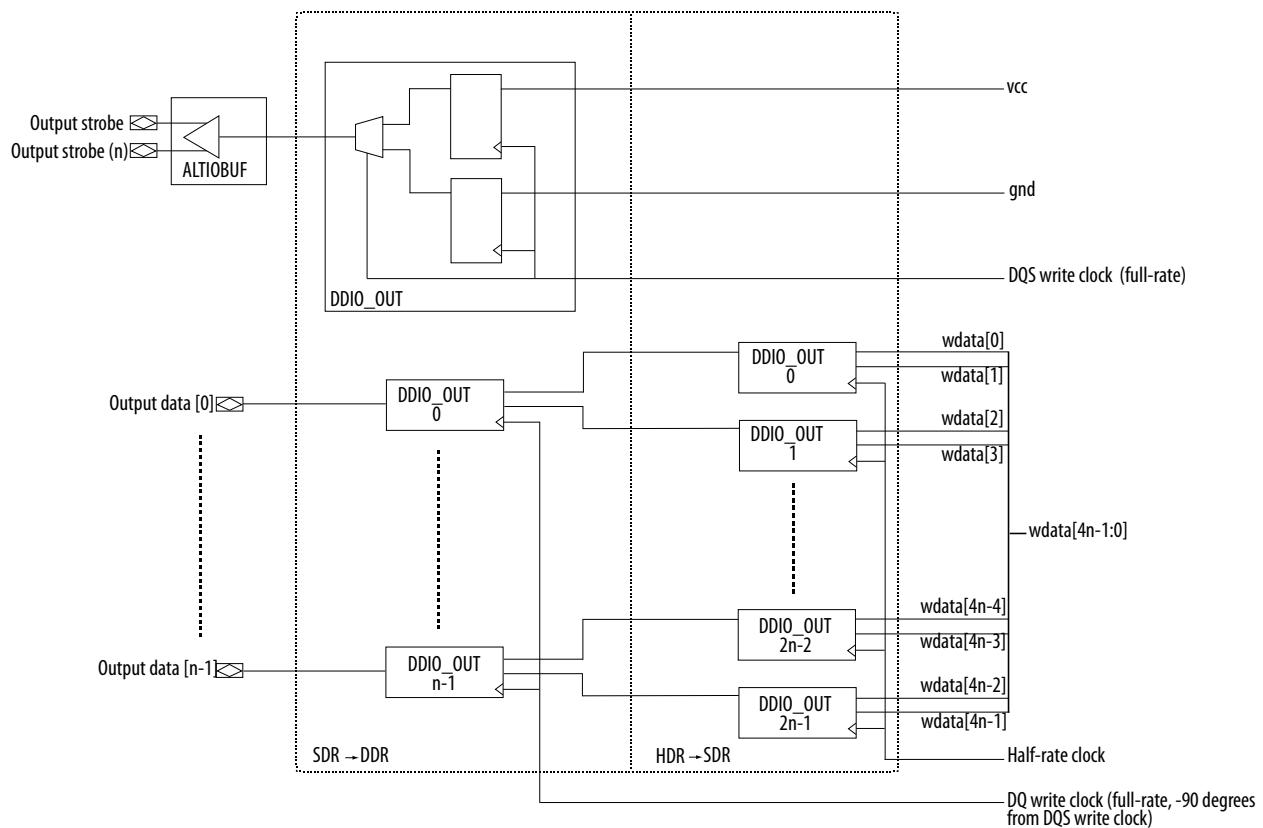


## Write Datapath

The write datapath passes write data from the memory controller to the I/O. The write data valid signal from the memory controller generates the output enable signal to control the output buffer. For memory protocols with a bidirectional data bus, it also generates the dynamic termination control signal, which selects between series (output mode) and parallel (input mode) termination.

The figure below illustrates a simplified write datapath of a typical half-rate interface. The full-rate DQS write clock is sent to a DDIO\_OUT cell. The output of DDIO\_OUT feeds an output buffer which creates a pair of pseudo differential clocks that connects to the memory. In full-rate mode, only the SDR-DDR portion of the path is used; in half-rate mode, the HDR-SDR circuitry is also required. The use of DDIO\_OUT in both the output strobe and output data generation path ensures that their timing characteristics are as similar as possible. The `<variation_name>.pin_assignments.tcl` script automatically specifies the logic option that associates all data pins to the output strobe pin. The Fitter treats the pins as a DQS/DQ pin group.

**Figure 1-3: Write Datapath**



## Leveling Circuitry

Leveling circuitry is dedicated I/O circuitry to provide calibration support for fly-by address and command networks. For DDR3, leveling is always invoked, whether the interface targets a DIMM or a single component. For DDR3 implementations at higher frequencies, a fly-by topology is recommended

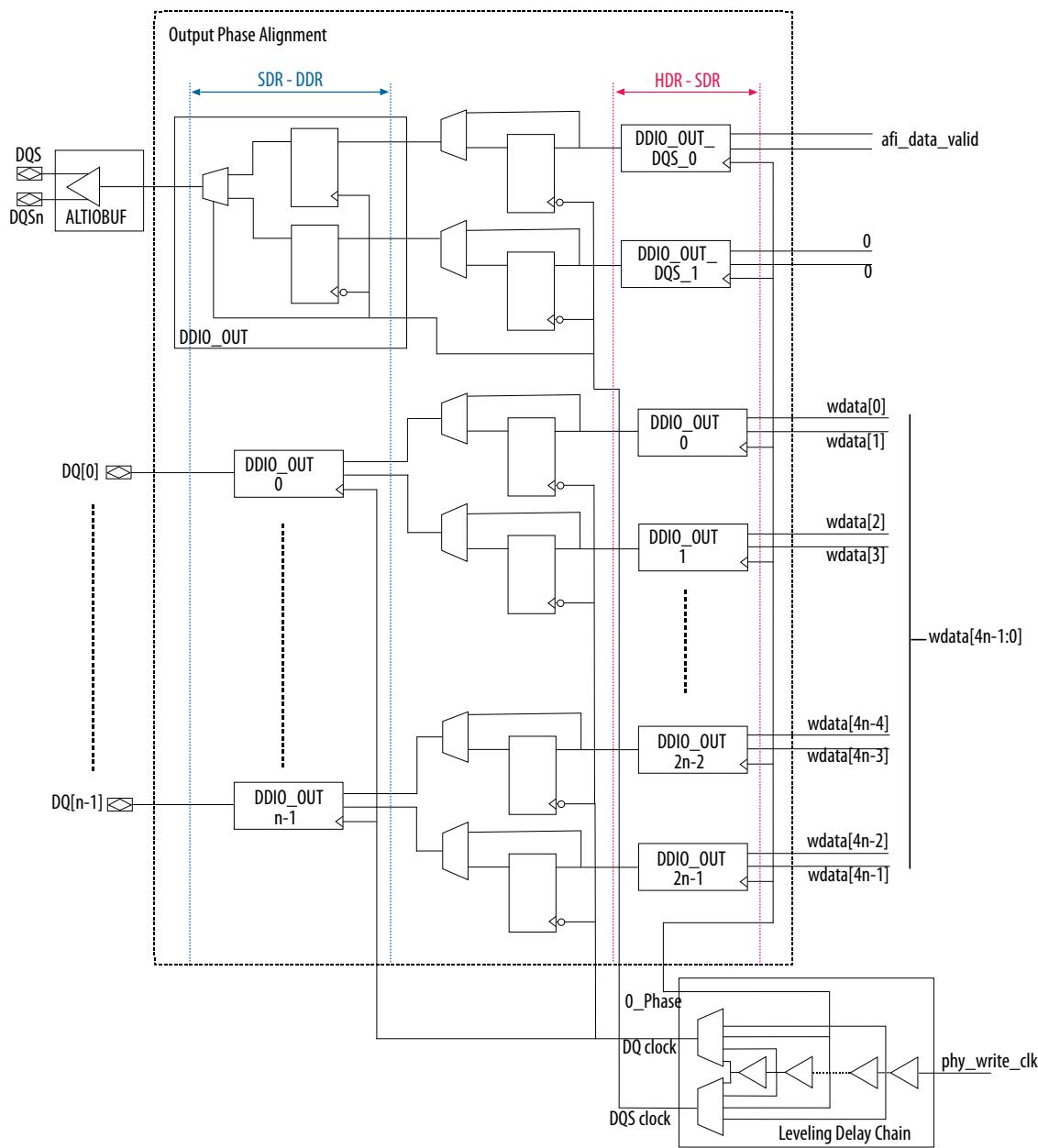
for optimal performance. For DDR2, leveling circuitry is invoked automatically for frequencies above 240 MHz; no leveling is used for frequencies below 240 MHz.

For DDR2 at frequencies below 240 MHz, you should use a tree-style layout. For frequencies above 240 MHz, you can choose either a leveled or balanced-T or Y topology, as the leveled PHY calibrates to either implementation. Regardless of protocol, for devices without a levelling block—such as Arria II GZ, Arria V, and Cyclone V—a balanced-T PCB topology for address/command/clock must be used because fly-by topology is not supported.

For details about leveling delay chains, consult the memory interfaces hardware section of the device handbook for your FPGA.

The following figure shows the write datapath for a leveling interface. The full-rate PLL output clock `phy_write_clk` goes to a leveling delay chain block which generates all other periphery clocks that are needed. The data signals that generate DQ and DQS signals pass to an output phase alignment block. The output phase alignment block feeds an output buffer which creates a pair of pseudo differential clocks that connect to the memory. In full-rate designs, only the SDR-DDR portion of the path is used; in half-rate mode, the HDR-SDR circuitry is also required. The use of `DDIO_OUT` in both the output strobe and output data generation paths ensures that their timing characteristics are as similar as possible. The `<variation_name>.pin_assignments.tcl` script automatically specifies the logic option that associates all data pins to the output strobe pin. The Quartus Prime Fitter treats the pins as a DQS/DQ pin group.

Figure 1-4: Write Datapath for a Leveling Interface



## Read Datapath

The read datapath passes read data from memory to the PHY. The following figure shows the blocks and flow in the read datapath.

For all protocols, the DQS logic block delays the strobe by 90 degrees to center-align the rising strobe edge within the data window. For DDR2, DDR3, and LPDDR2 protocols, the logic block also performs strobe gating, holding the DQS enable signal high for the entire period that data is received. One DQS logic block exists for each data group.

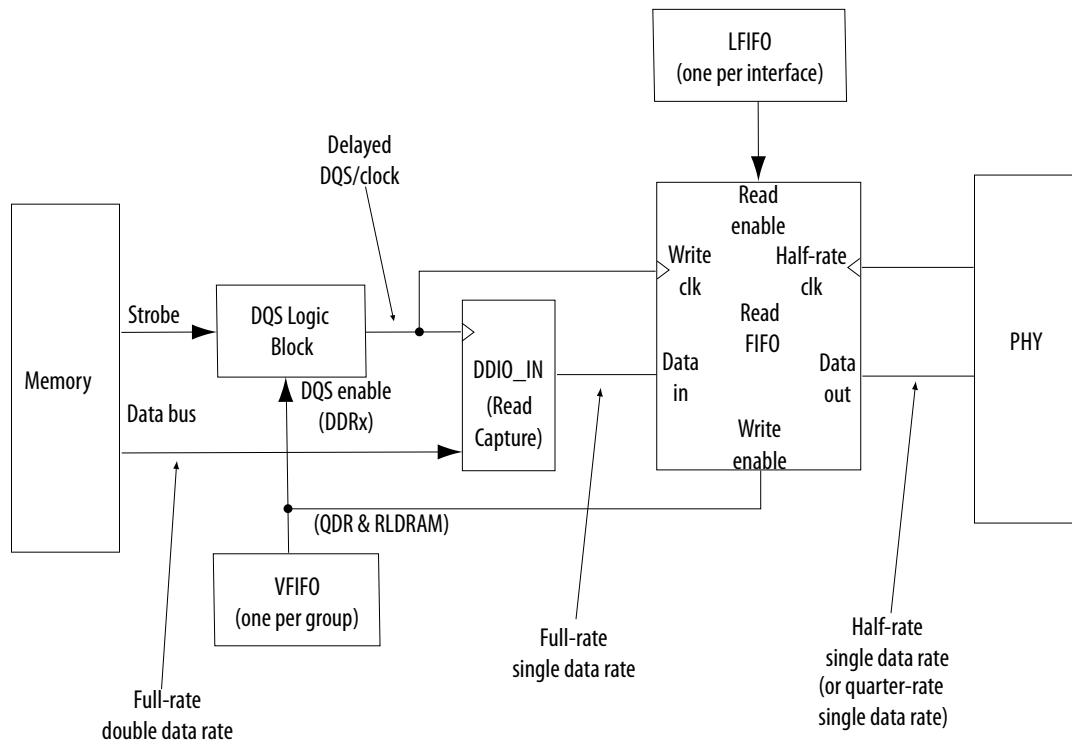
One VFIFO buffer exists for each data group. For DDR2, DDR3, and LPDDR2 protocols, the VFIFO buffer generates the DQS enable signal, which is delayed (by an amount determined during calibration) to align with the incoming DQS signal. For QDR and RLDRAM protocols, the output of the VFIFO buffer serves as the write enable signal for the Read FIFO buffer, signaling when to begin capturing data.

DDIO\_IN receives data from memory at double-data rate and passes data on to the Read FIFO buffer at single-data rate.

The Read FIFO buffer temporarily holds data read from memory; one Read FIFO buffer exists for each data group. For half-rate interfaces, the Read FIFO buffer converts the full-rate, single data-rate input to a half-rate, single data-rate output which is then passed to the PHY core logic. In the case of a quarter-rate interface, soft logic in the PHY performs an additional conversion from half-rate single data rate to quarter-rate single data rate.

One LFIFO buffer exists for each memory interface; the LFIFO buffer generates the read enable signal for all Read FIFO blocks in an interface. The read enable signal is asserted when the Read FIFO blocks have buffered sufficient data from the memory to be read. The timing of the read enable signal is determined during calibration.

**Figure 1-5: Read Datapath**



## Sequencer

Depending on the combination of protocol and IP architecture in your external memory interface, you may have either an RTL-based sequencer or a Nios® II-based sequencer.

RTL-based sequencer implementations and Nios II-based sequencer implementations can have different pin requirements. You may not be able to migrate from an RTL-based sequencer to a Nios II-based sequencer and maintain the same pinout.

For information on sequencer support for different protocol-architecture combinations, refer to *Introduction to Altera Memory Solution* in Volume 1 of this handbook. For information on pin planning, refer to *Planning Pin and FPGA Resources* in Volume 2 of this handbook.

#### Related Information

- [Protocol Support Matrix](#)
- [Planning Pin and FPGA Resources](#)

## Nios II-Based Sequencer

The DDR2, DDR3, and LPDDR2 controllers with UniPHY employ a Nios II-based sequencer that is parameterizable and is dynamically generated at run time. The Nios II-based sequencer is also available with the QDR II and RLDRAM II controllers.

### NIOS II-based Sequencer Function

The sequencer enables high-frequency memory interface operation by calibrating the interface to compensate for variations in setup and hold requirements caused by transmission delays.

UniPHY converts the double-data rate interface of high-speed memory devices to a full-rate or half-rate interface for use within an FPGA. To compensate for slight variations in data transmission to and from the memory device, double-data rate is usually center-aligned with its strobe signal; nonetheless, at high speeds, slight variations in delay can result in setup or hold time violations. The sequencer implements a calibration algorithm to determine the combination of delay and phase settings necessary to maintain center-alignment of data and clock signals, even in the presence of significant delay variations. Programmable delay chains in the FPGA I/Os then implement the calculated delays to ensure that data remains centered. Calibration also applies settings to the FIFO buffers within the PHY to minimize latency and ensures that the read valid signal is generated at the appropriate time.

When calibration is completed, the sequencer returns control to the memory controller.

For more information about calibration, refer to *UniPHY Calibration Stages*, in this chapter.

#### Related Information

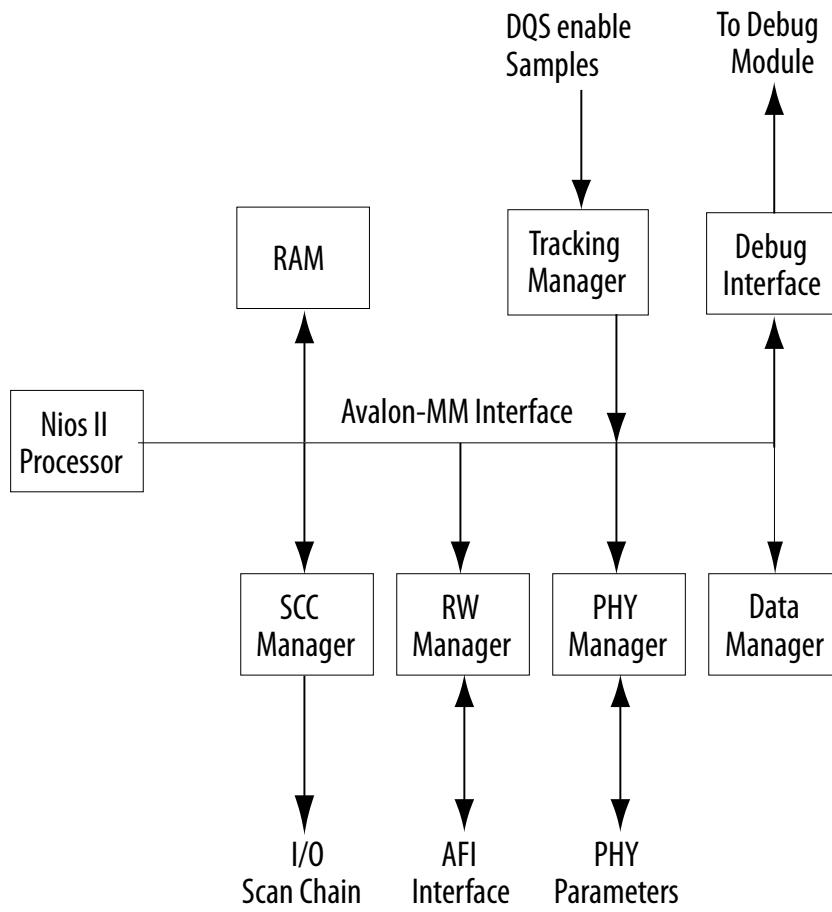
- [UniPHY Calibration Stages](#) on page 1-51

## NIOS II-based Sequencer Architecture

The sequencer is composed of a Nios II processor and a series of hardware-based component managers, connected together by an Avalon bus. The Nios II processor performs the high-level algorithmic operations of calibration, while the component managers handle the lower-level timing, memory protocol, and bit-manipulation operations.

The high-level calibration algorithms are specified in C code, which is compiled into Nios II code that resides in the FPGA RAM blocks. The debug interface provides a mechanism for interacting with the various managers and for tracking the progress of the calibration algorithm, and can be useful for debugging problems that arise within the PHY. The various managers are specified in RTL and implement operations that would be slow or inefficient if implemented in software.

Figure 1-6: NIOS II-based Sequencer Block Diagram



The C code that defines the calibration routines is available for your reference in the `\<name>_s0_software` subdirectory. Altera recommends that you do not modify this C code.

### NIOS II-based Sequencer SCC Manager

The scan chain control (SCC) manager allows the sequencer to set various delays and phases on the I/Os that make up the memory interface. The latest Altera device families provide dynamic delay chains on input, output, and output enable paths which can be reconfigured at runtime. The SCC manager provides the calibration routines access to these chains to add delay on incoming and outgoing signals. A master on the Avalon-MM interface may require the maximum allowed delay setting on input and output paths, and may set a particular delay value in this range to apply to the paths.

The SCC manager implements the Avalon-MM interface and the storage mechanism for all input, output, and phase settings. It contains circuitry that configures a DQ- or DQS-configuration block. The Nios II processor may set delay, phases, or register settings; the sequencer scans the settings serially to the appropriate DQ or DQS configuration block.

### NIOS II-based Sequencer RW Manager

The read write (RW) manager encapsulates the protocol to read and write to the memory device through the Altera PHY Interface (AFI). It provides a buffer that stores the data to be sent to and read from memory, and provides the following commands:

- Write configuration—configures the memory for use. Sets up burst lengths, read and write latencies, and other device specific parameters.
- Refresh—initiates a refresh operation at the DRAM. The command does not exist on SRAM devices. The sequencer also provides a register that determines whether the RW manager automatically generates refresh signals.
- Enable or disable multi-purpose register (MPR)—for memory devices with a special register that contains calibration specific patterns that you can read, this command enables or disables access to the register.
- Activate row—for memory devices that have both rows and columns, this command activates a specific row. Subsequent reads and writes operate on this specific row.
- Precharge—closes a row before you can access a new row.
- Write or read burst—writes or reads a burst length of data.
- Write guaranteed—writes with a special mode where the memory holds address and data lines constant. Altera guarantees this type of write to work in the presence of skew, but constrains to write the same data across the entire burst length.
- Write and read back-to-back—performs back-to-back writes or reads to adjacent banks. Most memory devices have strict timing constraints on subsequent accesses to the same bank, thus back-to-back writes and reads have to reference different banks.
- Protocol-specific initialization—a protocol-specific command required by the initialization sequence.

## NIOS II-based Sequencer PHY Manager

The PHY Manager provides access to the PHY for calibration, and passes relevant calibration results to the PHY. For example, the PHY Manager sets the VFIFO and LFIFO buffer parameters resulting from calibration, signals the PHY when the memory initialization sequence finishes, and reports the pass/fail status of calibration.

## NIOS II-based Sequencer Data Manager

The Data Manager stores parameterization-specific data in RAM, for the software to query.

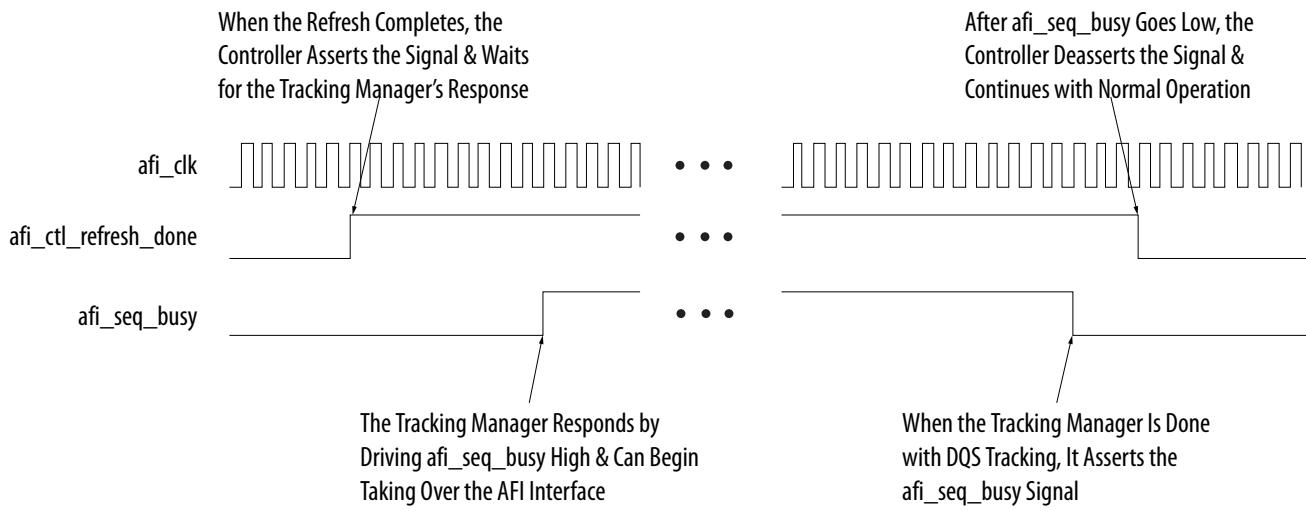
## NIOS II-based Sequencer Tracking Manager

The Tracking Manager detects the effects of voltage and temperature variations that can occur on the memory device over time resulting in reduced margins, and adjusts the DQS enable delay as necessary to maintain adequate operating margins.

The Tracking Manager briefly assumes control of the AFI interface after each memory refresh cycle, issuing a read routine to the RW Manager, and then sampling the DQS tracking. Ideally, the falling edge of the DQS enable signal would align to the last rising edge of the raw DQS signal from the memory device. The Tracking Manager determines whether the DQS enable signal is leading or trailing the raw DQS signal.

Each time a refresh occurs, the Tracking Manager takes a sample of the raw DQS signal; any adjustments of the DQS enable signal occur only after sufficient samples of raw DQS have been taken. When the Tracking Manager determines that the DQS enable signal is either leading or lagging the raw DQS signal, it adjusts the DQS enable appropriately.

The following figure shows the Tracking manager signals.

**Figure 1-7: Tracking Manager Signals**

Some notes on Tracking Manager operation:

- The time taken by the Tracking Manager is arbitrary; if the period taken exceeds the refresh period, the Tracking Manager handles memory refresh.
- afi\_seq\_busy should go high fewer than 10 clock cycles after afi\_ctl\_refresh\_done or afi\_ctl\_long\_idle is asserted.
- afi\_refresh\_done should deassert fewer than 10 clock cycles after afi\_seq\_busy deasserts.
- afi\_ctl\_long\_idle causes the Tracking Manager to execute an algorithm different than periodic refresh; use afi\_ctl\_long\_idle when a long session has elapsed without a periodic refresh.
- The Tracking Manager is instantiated into the sequencer system when DQS Tracking is turned on.

**Table 1-1: Configurations Supporting DQS Tracking**

Device Family	Protocol	Memory Clock Frequency
Arria V (GX/GT/SX/ST), Cyclone V	LPDDR2 (single rank)	All frequencies.
Arria V (GX/GT/SX/ST)	DDR3 (single rank)	450 MHz or higher for speed grade 5, or higher than 534 MHz.
Arria V GZ, Stratix V (E/GS/GT/GX)		750 MHz or higher.

- If you do not want to use DQS tracking, you can disable it (at your own risk), by opening the Verilog file `<variant_name>.if0_c0.v` in an editor, and changing the value of the `USE_DQS_TRACKING` parameter from 1 to 0.

## NIOS II-based Sequencer Processor

The Nios II processor manages the calibration algorithm; the Nios II processor is unavailable after calibration is completed.

The same calibration algorithm supports all device families, with some differences. The following sections describe the calibration algorithm for DDR3 SDRAM on Stratix III devices. Calibration algorithms for other protocols and families are a subset and significant differences are pointed out when necessary. As

the algorithm is fully contained in the software of the sequencer (in the C code) enabling and disabling specific steps involves turning flags on and off.

Calibration consists of the following stages:

- Initialize memory.
- Calibrate read datapath.
- Calibrate write datapath.
- Run diagnostics.

## NIOS II-based Sequencer Calibration and Diagnostics

Calibration must initialize all memory devices before they can operate properly. The sequencer performs this memory initialization stage when it takes control of the PHY at startup.

Calibrating the read datapath comprises the following steps:

- Calibrate DQS enable cycle and phase.
- Perform read per-bit deskew to center the strobe signal within data valid window.
- Reduce LFIFO latency.

Calibrating the write datapath involves the following steps:

- Center align DQS with respect to DQ.
- Align DQS with `mem_clk`.

The sequencer estimates the read and write margins under noisy conditions, by sweeping input and output DQ and DQS delays to determine the size of the data valid windows on the input and output sides. The sequencer stores this diagnostic information in the local memory and you can access it through the debugging interface.

When the diagnostic test finishes, control of the PHY interface passes back to the controller and the sequencer issues a pass or fail signal.

### Related Information

[External Memory Interface Debug Toolkit](#) on page 15-1

## RTL-based Sequencer

The RTL-based sequencer is available for QDR II and RLDRAM II interfaces, on supported device families other than Arria V. The RTL sequencer is a state machine that processes the calibration algorithm.

The sequencer assumes control of the interface at reset (whether at initial startup or when the IP is reset) and maintains control throughout the calibration process. The sequencer relinquishes control to the memory controller only after successful calibration. The following tables list the major states in the RTL-based sequencer.

**Table 1-2: Sequencer States**

RTL-based Sequencer State	Description
RESET	Remain in this state until reset is released.
LOAD_INIT	Load any initialization values for simulation purposes.

RTL-based Sequencer State	Description
STABLE	Wait until the memory device is stable.
WRITE_ZERO	Issue write command to address 0.
WAIT_WRITE_ZERO	Write all 0xAs to address 0.
WRITE_ONE	Issue write command to address 1.
WAIT_WRITE_ONE	Write all 0x5s to address 1.

**Valid Calibration States**

V_READ_ZERO	Issue read command to address 0 (expected data is all 0xAs).
V_READ_NOP	This state represents the minimum number of cycles required between 2 back-to-back read commands. The number of NOP states depends on the burst length.
V_READ_ONE	Issue read command to address 1 (expected data is all 0x5s).
V_WAIT_READ	Wait for read valid signal.
V_COMPARE_READ_ZERO_READ_ONE	Parameterizable number of cycles to wait before making the read data comparisons.
V_CHECK_READ_FAIL	When a read fails, the write pointer (in the AFI clock domain) of the valid FIFO buffer is incremented. The read pointer of the valid FIFO buffer is in the DQS clock domain. The gap between the read and write pointers is effectively the latency between the time when the PHY receives the read command and the time valid data is returned to the PHY.
V_ADD_FULL_RATE	Advance the read valid FIFO buffer write pointer by an extra full rate cycle.
V_ADD_HALF_RATE	Advance the read valid FIFO buffer write pointer by an extra half rate cycle. In full-rate designs, equivalent to V_ADD_FULL_RATE.
V_READ_FIFO_RESET	Reset the read and write pointers of the read data synchronization FIFO buffer.
V_CALIB_DONE	Valid calibration is successful.

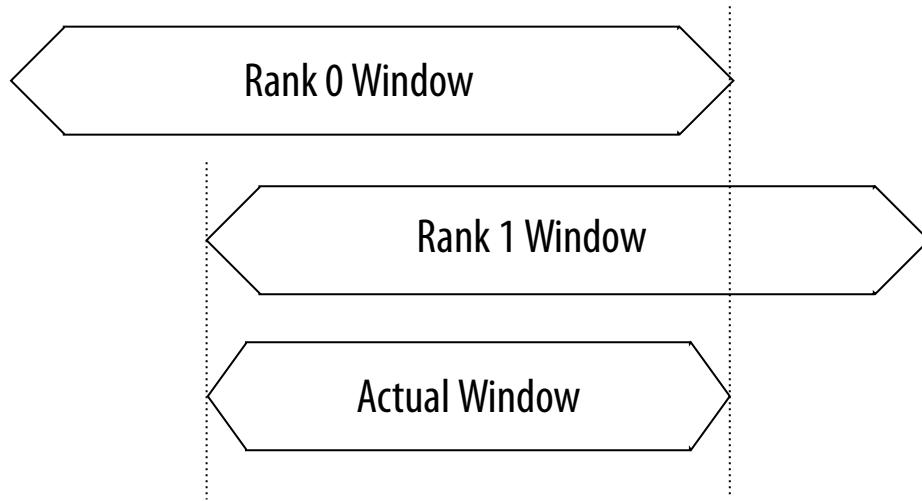
RTL-based Sequencer State	Description
<b>Latency Calibration States</b>	
L_READ_ONE	Issue read command to address 1 (expected data is all 0x5s).
L_WAIT_READ	Wait for read valid signal from read datapath. Initial read latency is set to a predefined maximum value.
L_COMPARE_READ_ONE	Check returned read data against expected data. If data is correct, go to L_REDUCE_LATENCY; otherwise go to L_ADD_MARGIN.
L_REDUCE_LATENCY	Reduce the latency counter by 1.
L_READ_FLUSH	Read from address 0, to flush the contents of the read data resynchronization FIFO buffer.
L_WAIT_READ_FLUSH	Wait until the whole FIFO buffer is flushed, then go back to L_READ and try again.
L_ADD_MARGIN	Increment latency counter by 3 (1 cycle to get the correct data, 2 more cycles of margin for run time variations). If latency counter value is smaller than predefined ideal condition minimum, then go to CALIB_FAIL.
CALIB_DONE	Calibration is successful.
CALIB_FAIL	Calibration is not successful.

## Shadow Registers

Shadow registers are a hardware feature of Arria V GZ and Stratix V devices that enables high-speed multi-rank calibration for DDR3 quarter-rate and half-rate memory interfaces, up to 800MHz for dual-rank interfaces and 667MHz for quad-rank interfaces.

Prior to the introduction of shadow registers, the data valid window of a multi-rank interface was calibrated to the overlapping portion of the data valid windows of the individual ranks. The resulting data valid window for the interface would be smaller than the individual data valid windows, limiting overall performance.

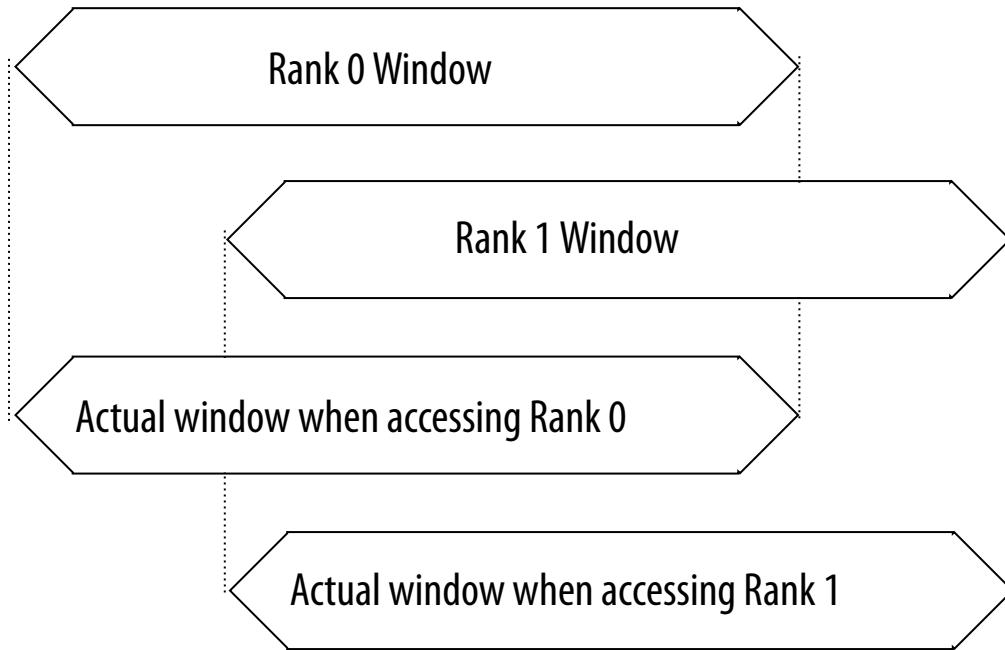
Figure 1-8: Calibration of Overlapping Data Valid Windows, without Shadow Registers



Shadow registers allow the sequencer to calibrate each rank separately and fully, and then to save the calibrated settings for each rank in its own set of shadow registers, which are part of the IP scan chains. During a rank-to-rank switch, the rank-specific set of calibration settings is restored just-in-time to optimize the data valid window for each rank.

The following figure illustrates how the use of rank-specific calibration settings results in a data valid window appropriate for the current rank.

Figure 1-9: Rank-specific Calibration Settings, with Shadow Registers



The shadow registers and their associated rank-switching circuitry are part of the device I/O periphery hardware.

## Shadow Registers Operation

The sequencer calibrates each rank individually and stores the resulting configuration in shadow registers, which are part of the IP scan chains. UniPHY then selects the appropriate configuration for the rank in use, switching between configurations as necessary. Calibration results for deskew delay chains are stored in the shadow registers. For DQS enable/disable, delay chain configurations come directly from the FPGA core.

### Signals

The afi\_wrrank signal indicates the rank to which the controller is writing, so that the PHY can switch to the appropriate setting. Signal timing is identical to afi\_dqs\_burst; that is, afi\_wrrank must be asserted at the same time as afi\_dqs\_burst, and must be of the same duration.

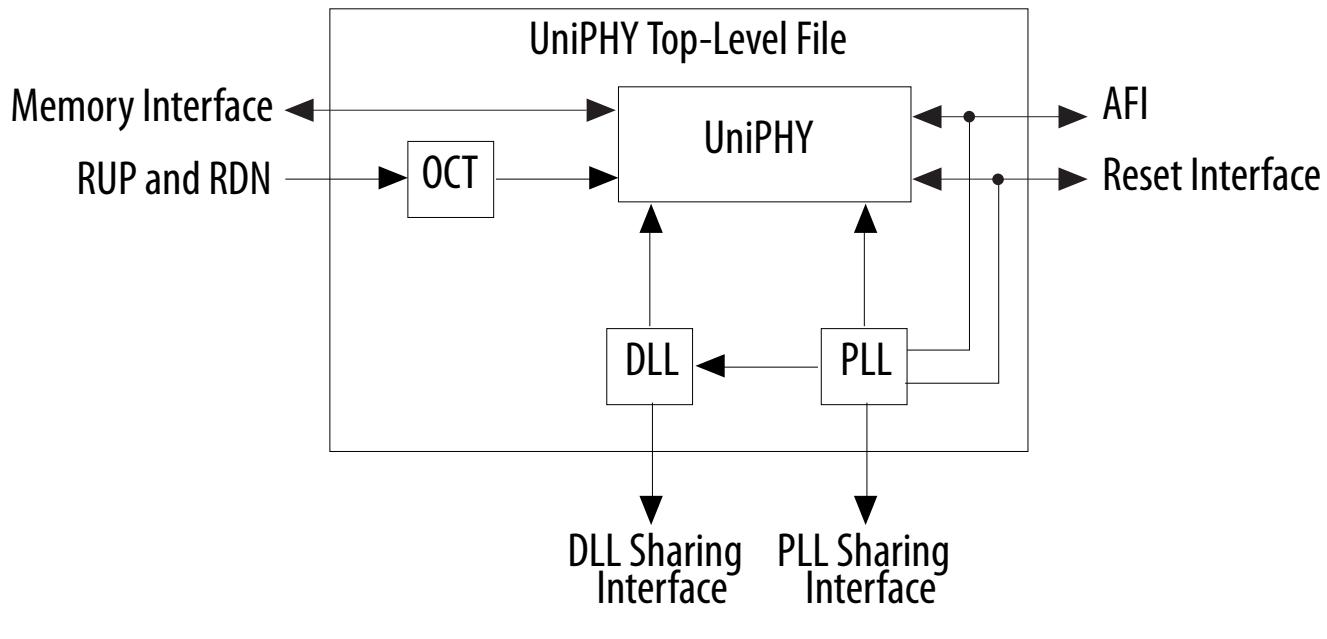
The afi\_rrank signal indicates the rank from which the controller is reading, so that the PHY can switch to the appropriate setting. This signal must be asserted at the same time as afi\_rdata\_en when issuing a read command, and once asserted, must remain unchanged until the controller issues a new read command to another rank.

## UniPHY Interfaces

The following figure shows the major blocks of the UniPHY and how it interfaces with the external memory device and the controller.

**Note:** Instantiating the delay-locked loop (DLL) and the phase-locked loop (PLL) on the same level as the UniPHY eases DLL and PLL sharing.

Figure 1-10: UniPHY Interfaces with the Controller and the External Memory



The following interfaces are on the UniPHY top-level file:

- AFI
- Memory interface
- DLL sharing interface
- PLL sharing interface
- OCT interface

### AFI

The UniPHY datapath uses the Altera PHY interface (AFI). The AFI is in a simple connection between the PHY and controller. The AFI is based on the DDR PHY interface (DFI) specification, with some calibration-related signals not used and some additional Altera-specific sideband signals added.

For more information about the AFI, refer to *AFI 4.0 Specification*, in this chapter.

### The Memory Interface

For information on the memory interface, refer to *UniPHY Signals*, in this chapter.

#### Related Information

- [AFI 4.0 Specification](#) on page 2-52
- [UniPHY Signals](#) on page 1-21

## The DLL and PLL Sharing Interface

You can generate the UniPHY memory interface and configure it to share its PLL, DLL, or both interfaces.

By default, a UniPHY memory interface variant contains a PLL and DLL; the PLL produces a variety of required clock signals derived from the reference clock, and the DLL produces a delay codeword. In this case the PLL sharing mode is "No sharing". A UniPHY variant can be configured as a PLL Master and/or DLL Master, in which case the corresponding interfaces are exported to the UniPHY top-level and can be connected to an identically configured UniPHY variant PLL Slave and/or DLL Slave. The UniPHY slave variant is instantiated without a PLL and/or DLL, which saves device resources.

**Note:** For Arria II GX, Arria II GZ, Stratix III, and Stratix IV devices, the PLL and DLL must both be shared at the same time—their sharing modes must match. This restriction does not apply to Arria V, Arria V GZ, Cyclone V, or Stratix V devices.

**Note:** For devices with hard memory interface components onboard, you cannot share PLL or DLL resources between soft and hard interfaces.

### Sharing PLLs or DLLs

To share PLLs or DLLs, follow these steps:

1. To create a PLL or DLL master, create a UniPHY memory interface IP core. To make the PLL and/or DLL interface appear at the top-level in the core, on the **PHY Settings** tab in the parameter editor, set the **PLL Sharing Mode** and/or **DLL Sharing Mode** to **Master**.
2. To create a PLL or DLL slave, create a second UniPHY memory interface IP core. To make the PLL and/or DLL interface appear at the top-level in the core, on the **PHY Settings** tab set the **PLL Sharing Mode** and/or **DLL Sharing Mode** to **Slave**.
3. Connect the PLL and/or DLL sharing interfaces by following the appropriate step, below:

- **For cores generated with IP Catalog**: connect the PLL and/or DLL interface ports between the master and slave cores in your wrapper RTL. When using PLL sharing, connect the `afi_clk`, `afi_half_clk`, and `afi_reset_export_n` outputs from the UniPHY PLL master to the `afi_clk`, `afi_half_clk`, and `afi_reset_in` inputs on the UniPHY PLL slave.
- **For cores generated with Qsys**, connect the PLL and/or DLL interface in the Qsys GUI. When using PLL sharing, connect the `afi_clk`, `afi_half_clk`, and `afi_reset_export_n` outputs from the UniPHY PLL master to the `afi_clk`, `afi_half_clk`, and `afi_reset_in` inputs on the UniPHY PLL slave.

Qsys supports only one-to-one conduit connections in the patch panel. To share a PLL from a UniPHY PLL master with multiple slaves, you should replicate the number of PLL sharing conduit interfaces in the Qsys patch panel by choosing **Number of PLL sharing interfaces** in the parameter editor.

**Note:** You may connect a slave UniPHY instance to the clocks from a user-defined PLL instead of from a UniPHY master. The general procedure for doing so is as follows:

1. Make a template, by generating your IP with **PLL Sharing Mode** set to `No Sharing`, and then compiling the example project to determine the frequency and phases of the clock outputs from the PLL.
2. Generate an external PLL using the IP Catalog flow, with the equivalent output clocks.
3. Generate your IP with **PLL Sharing Mode** set to `Slave`, and connect the external PLL to the PLL sharing interface.

You must be very careful when connecting clock signals to the slave. Connecting to clocks with frequency or phase different than what the core expects may result in hardware failure.

**Note:** The signal `dll_pll_locked` is an internal signal from the PLL to the DLL which ensures that the DLL remains in reset mode until the PLL becomes locked. This signal is not available for use by customer logic.

## About PLL Simulation

PLL frequencies may differ between the synthesis and simulation file sets. In either case the achieved PLL frequencies and phases are calculated and reported in real time in the parameter editor.

For the simulation file set, clocks are specified in the RTL, not in units of frequency but by the period in picoseconds, thus avoiding clock drift due to picosecond rounding error.

For the synthesis file set, there are two mechanisms by which clock frequencies are specified in the RTL, based on the target device family:

- For Arria V, Arria V GZ, Cyclone V, and Stratix V, clock frequencies are specified in MHz.
- For Arria II GX, Arria II GZ, Stratix III, and Stratix IV, clock frequencies are specified by integer multipliers and divisors. For these families, the real simulation model—as opposed to the default abstract simulation model—also uses clock frequencies specified by integer ratios.

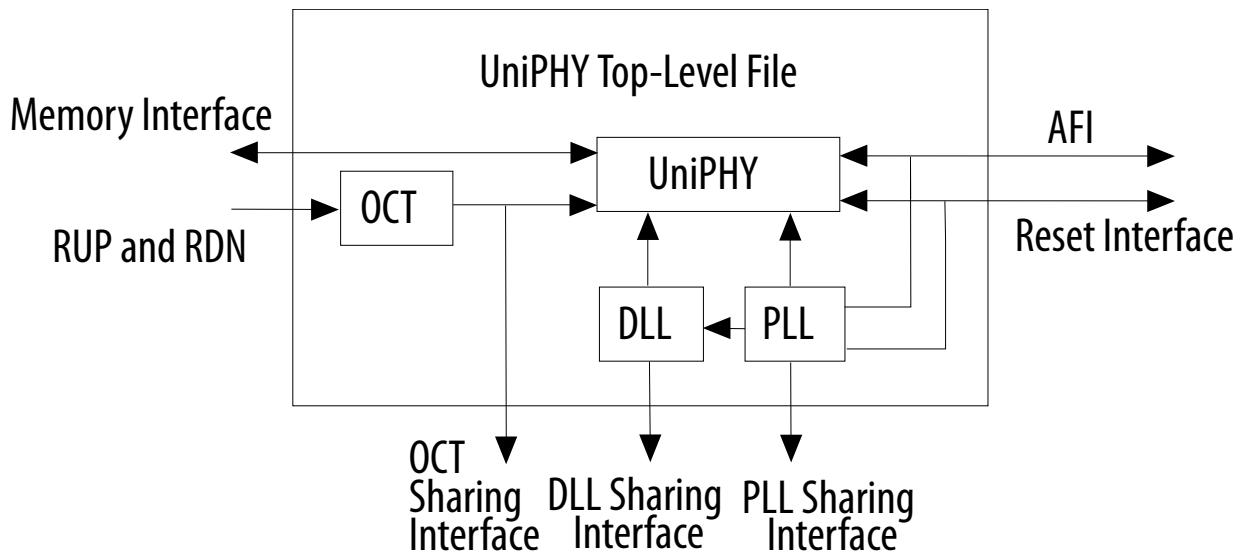
## The OCT Sharing Interface

By default, the UniPHY IP generates the required OCT control block at the top-level RTL file for the PHY.

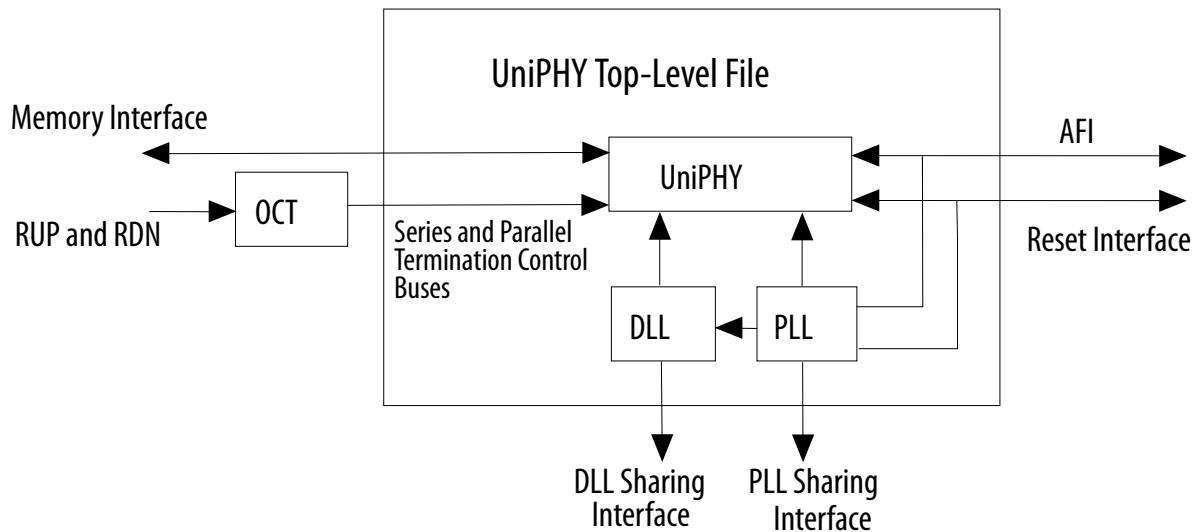
If you want, you can instantiate this block elsewhere in your code and feed the required termination control signals into the IP core by turning off **Master for OCT Control Block** on the **PHY Settings** tab. If you turn off **Master for OCT Control Block**, you must instantiate the OCT control block or use another UniPHY instance as a master, and ensure that the parallel and series termination control bus signals are connected to the PHY.

The following figures show the PHY architecture with and without Master for OCT Control Block.

**Figure 1-11: PHY Architecture with Master for OCT Control Block**



**Figure 1-12: PHY Architecture without Master for OCT Control Block**



### Modifying the Pin Assignment Script for QDR II and RLDRAM II

If you generate a QDR II or RLDRAM II slave IP core, you must modify the pin assignment script to allow the fitter to correctly resolve the OCT termination block name in the OCT master core.

To modify the pin assignment script for QDR II or RLDRAM II slaves, follow these steps:

1. In a text editor, open your system's Tcl pin assignments script file, as follows:

- For systems generated with the IP Catalog: Open the <IP core name>/<slave core name>\_p0\_pin\_assignments.tcl file.
  - For systems generated with Qsys: Open the <HDL Path>/<submodules>/<slave core name>\_p0\_pin\_assignments.tcl file.
2. Search for the following line:

```
set ::master_corename "_MASTER_CORE_"
```

3. Replace \_MASTER\_CORE\_ with the instance name of the UniPHY master to which the slave is connected. The instance name is determined from the pin assignments file name, as follows:

- For systems generated with Qsys, the instance name is the <master core name> component of the pins assignments file name: <HDL path>/<submodules>/<master core name>\_p0\_pin\_assignments.tcl.
- For systems generated with the IP Catalog, the instance name is the <master core name> component of the pins assignments file name: <IP core name>/<master core name>\_p0\_pin\_assignments.tcl .

## UniPHY Signals

The following tables list the UniPHY signals.

**Table 1-3: Clock and Reset Signals**

Name	Direction	Width	Description
pll_ref_clk	Input	1	PLL reference clock input.
global_reset_n	Input	1	Active low global reset for PLL and all logic in the PHY, which causes a complete reset of the whole system. Minimum recommended pulse width is 100ns.
soft_reset_n	Input	1	Holding soft_reset_n low holds the PHY in a reset state. However it does not reset the PLL, which keeps running. It also holds the afi_reset_n output low.

**Table 1-4: DDR2 and DDR3 SDRAM Interface Signals**

Name	Direction	Width	Description
mem_ck, mem_ck_n	Output	MEM_CK_WIDTH	Memory clock.
mem_cke	Output	MEM_CLK_EN_WIDTH	Clock enable.
mem_cs_n	Output	MEM_CHIP_SELECT_WIDTH	Chip select..

Name	Direction	Width	Description
mem_cas_n	Output	MEM_CONTROL_WIDTH	Column address strobe.
mem_ras_n	Output	MEM_CONTROL_WIDTH	Row address strobe.
mem_we_n	Output	MEM_CONTROL_WIDTH	Write enable.
mem_a	Output	MEM_ADDRESS_WIDTH	Address.
mem_ba	Output	MEM_BANK_ADDRESS_WIDTH	Bank address.
mem_dqs, mem_dqs_n	Bidirectional	MEM_DQS_WIDTH	Data strobe.
mem_dq	Bidirectional	MEM_DQ_WIDTH	Data.
mem_dm	Output	MEM_DM_WIDTH	Data mask.
mem_odt	Output	MEM_ODT_WIDTH	On-die termination.
mem_reset_n (DDR3 only)	Output	1	Reset
mem_ac_parity (DDR3 only, RDIMM/LRDIMM only)	Output	MEM_CONTROL_WIDTH	Address/command parity bit. (Even parity, per the RDIMM spec, JESD82-29A.)
mem_err_out_n (DDR3 only, RDIMM/LRDIMM only)	Input	MEM_CONTROL_WIDTH	Address/command parity error.

**Table 1-5: UniPHY Parameters**

Parameter Name	Description
AFI_RATIO	AFI_RATIO is 1 in full-rate designs. AFI_RATIO is 2 for half-rate designs. AFI_RATIO is 4 for quarter-rate designs.
MEM_IF_DQS_WIDTH	The number of DQS pins in the interface.

Parameter Name	Description
MEM_ADDRESS_WIDTH	The address width of the specified memory device.
MEM_BANK_WIDTH	The bank width of the specified memory device.
MEM_CHIP_SELECT_WIDTH	The chip select width of the specified memory device.
MEM_CONTROL_WIDTH	The control width of the specified memory device.
MEM_DM_WIDTH	The DM width of the specified memory device.
MEM_DQ_WIDTH	The DQ width of the specified memory device.
MEM_READ_DQS_WIDTH	The READ DQS width of the specified memory device.
MEM_WRITE_DQS_WIDTH	The WRITE DQS width of the specified memory device.
OCT_SERIES_TERM_CONTROL_WIDTH	—
OCT_PARALLEL_TERM_CONTROL_WIDTH	—
AFI_ADDRESS_WIDTH	The AFI address width, derived from the corresponding memory interface width.
AFI_BANK_WIDTH	The AFI bank width, derived from the corresponding memory interface width.
AFI_CHIP_SELECT_WIDTH	The AFI chip select width, derived from the corresponding memory interface width.
AFI_DATA_MASK_WIDTH	The AFI data mask width.
AFI_CONTROL_WIDTH	The AFI control width, derived from the corresponding memory interface width.
AFI_DATA_WIDTH	The AFI data width.
AFI_DQS_WIDTH	The AFI DQS width.

Parameter Name	Description
DLL_DELAY_CTRL_WIDTH	The DLL delay output control width.
NUM_SUBGROUP_PER_READ_DQS	A read datapath parameter for timing purposes.
QVLD_EXTRA_FLOP_STAGES	A read datapath parameter for timing purposes.
READ_VALID_TIMEOUT_WIDTH	A read datapath parameter; calibration fails when the timeout counter expires.
READ_VALID_FIFO_WRITE_ADDR_WIDTH	A read datapath parameter; the write address width for half-rate clocks.
READ_VALID_FIFO_READ_ADDR_WIDTH	A read datapath parameter; the read address width for full-rate clocks.
MAX_LATENCY_COUNT_WIDTH	A latency calibration parameter; the maximum latency count width.
MAX_READ_LATENCY	A latency calibration parameter; the maximum read latency.
READ_FIFO_READ_ADDR_WIDTH	—
READ_FIFO_WRITE_ADDR_WIDTH	—
MAX_WRITE_LATENCY_COUNT_WIDTH	A write datapath parameter; the maximum write latency count width.
INIT_COUNT_WIDTH	An initailization sequence.
MRSC_COUNT_WIDTH	A memory-specific initialization parameter.
INIT_NOP_COUNT_WIDTH	A memory-specific initialization parameter.
MRS_CONFIGURATION	A memory-specific initialization parameter.
MRS_BURST_LENGTH	A memory-specific initialization parameter.
MRS_ADDRESS_MODE	A memory-specific initialization parameter.
MRS_DLL_RESET	A memory-specific initialization parameter.

Parameter Name	Description
MRS_IMP_MATCHING	A memory-specific initialization parameter.
MRS_ODT_EN	A memory-specific initialization parameter.
MRS_BURST_LENGTH	A memory-specific initialization parameter.
MEM_T_WL	A memory-specific initialization parameter.
MEM_T_RL	A memory-specific initialization parameter.
SEQ_BURST_COUNT_WIDTH	The burst count width for the sequencer.
VCALIB_COUNT_WIDTH	The width of a counter that the sequencer uses.
DOUBLE_MEM_DQ_WIDTH	—
HALF_AFI_DATA_WIDTH	—
CALIB_REG_WIDTH	The width of the calibration status register.
NUM_AFI_RESET	The number of AFI resets to generate.

**Note:** For information about the AFI signals, refer to *AFI 4.0 Specification* in this chapter.

#### Related Information

[AFI 4.0 Specification](#) on page 2-52

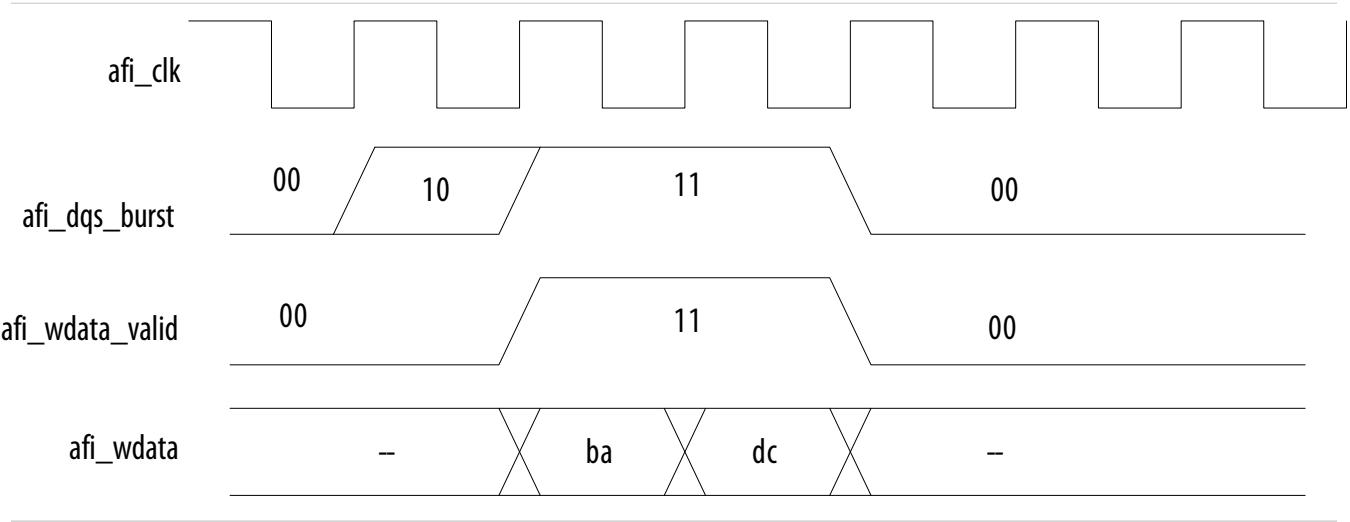
## PHY-to-Controller Interfaces

Various modules connect to UniPHY through specific ports.

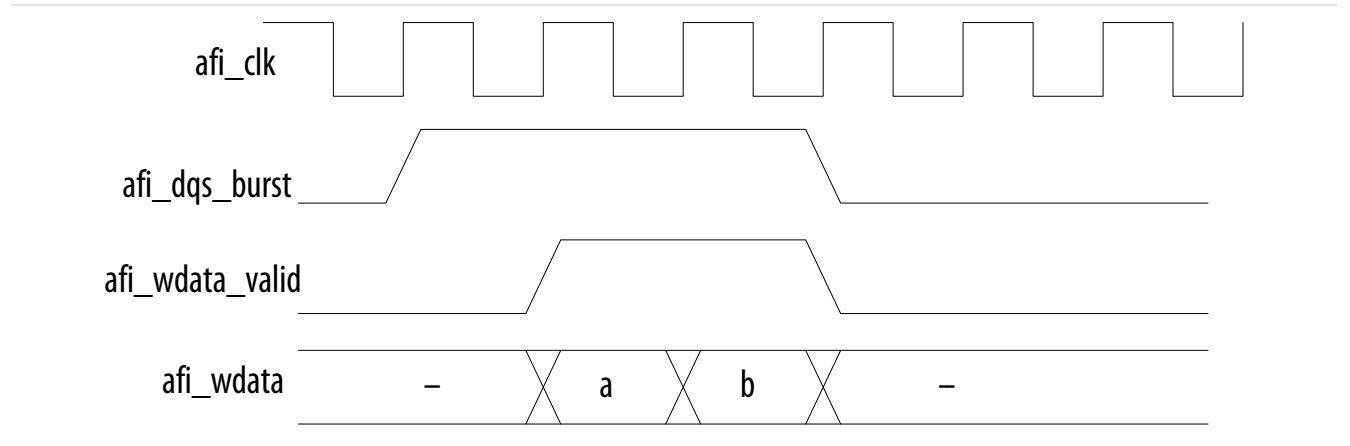
The AFI standardizes and simplifies the interface between controller and PHY for all Altera memory designs, thus allowing you to easily interchange your own controller code with Altera's high-performance controllers. The AFI PHY interface includes an administration block that configures the memory for calibration and performs necessary accesses to mode registers that configure the memory as required.

For half-rate designs, the address and command signals in the UniPHY are asserted for one `mem_clk` cycle (1T addressing), such that there are two input bits per address and command pin in half-rate designs. If you require a more conservative 2T addressing (where signals are asserted for two `mem_clk` cycles), drive both input bits (of the address and command signal) identically in half-rate designs.

The following figure shows the half-rate write operation.

**Figure 1-13: Half-Rate Write with Word-Aligned Data**

The following figure shows a full-rate write.

**Figure 1-14: Full-Rate Write**

For quarter-rate designs, the address and command signals in the UniPHY are asserted for one mem\_clk cycle (1T addressing), such that there are four input bits per address and command pin in quarter-rate designs. If you require a more conservative 2T addressing (where signals are asserted for two mem\_clk cycles), drive either the two lower input bits or the two upper input bits (of the address and command signal) identically.

After calibration is completed, the sequencer sends the write latency in number of clock cycles to the controller.

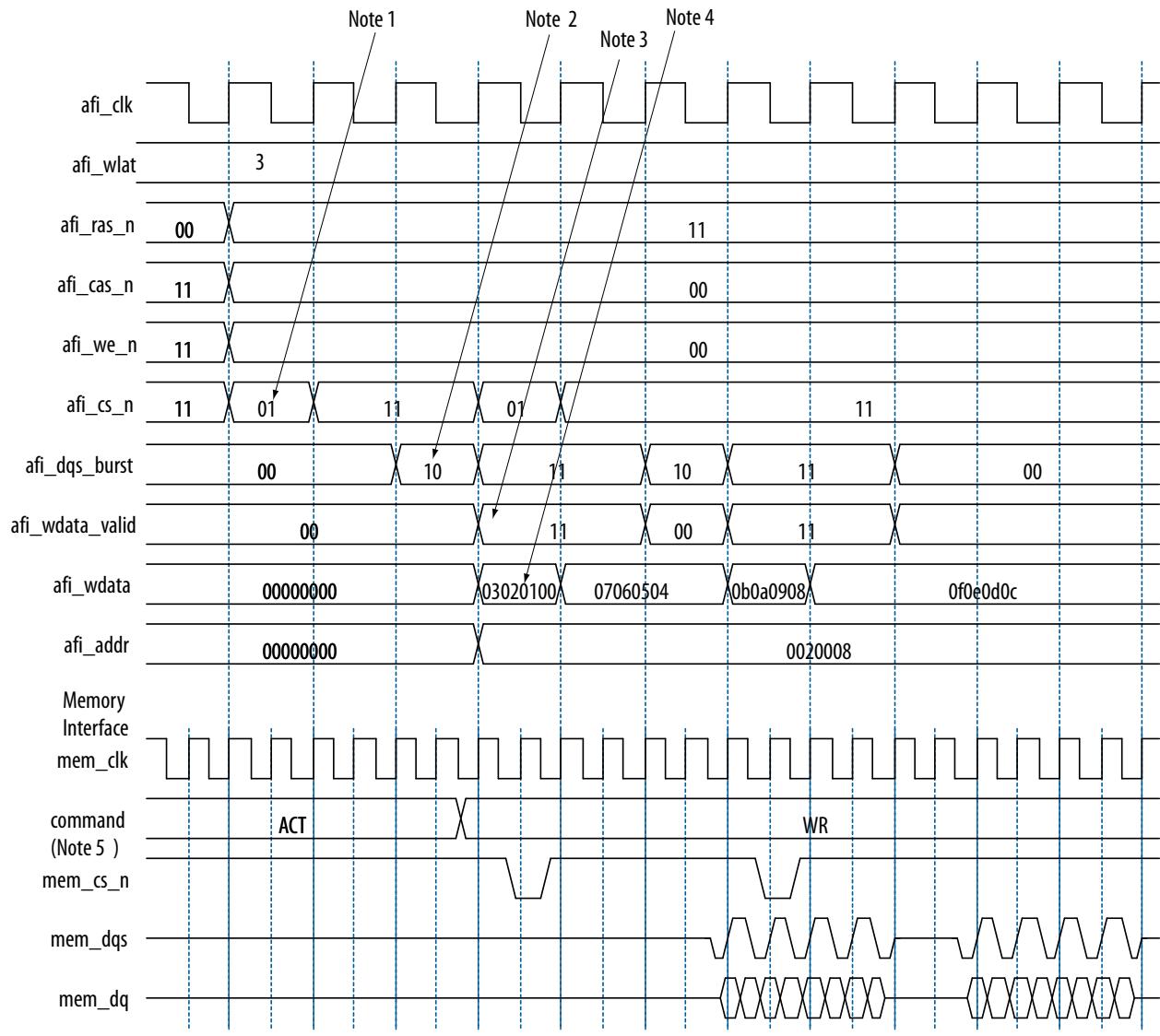
The AFI has the following conventions:

- With the AFI, high and low signals are combined in one signal, so for a single chip select (`afi_cs_n`) interface, `afi_cs_n[1:0]`, location 0 appears on the memory bus on one `mem_clk` cycle and location 1 on the next `mem_clk` cycle.
- Note:** This convention is maintained for all signals so for an 8 bit memory interface, the write data (`afi_wdata`) signal is `afi_wdata[31:0]`, where the first data on the DQ pins is `afi_wdata[7:0]`, then `afi_wdata[15:8]`, then `afi_wdata[23:16]`, then `afi_wdata[31:24]`.
- Spaced reads and writes have the following definitions:
    - Spaced writes—write commands separated by a gap of one controller clock (`afi_clk`) cycle.
    - Spaced reads—read commands separated by a gap of one controller clock (`afi_clk`) cycle.

The following figures show writes and reads, where the IP core writes data to and reads from the same address. In each example, `afi_rdata` and `afi_wdata` are aligned with controller clock (`afi_clk`) cycles. All the data in the bit vector is valid at once. These figures assume the following general points:

- The burst length is four.
- An 8-bit interface with one chip select.
- The data for one controller clock (`afi_clk`) cycle represents data for two memory clock (`mem_clk`) cycles (half-rate interface).

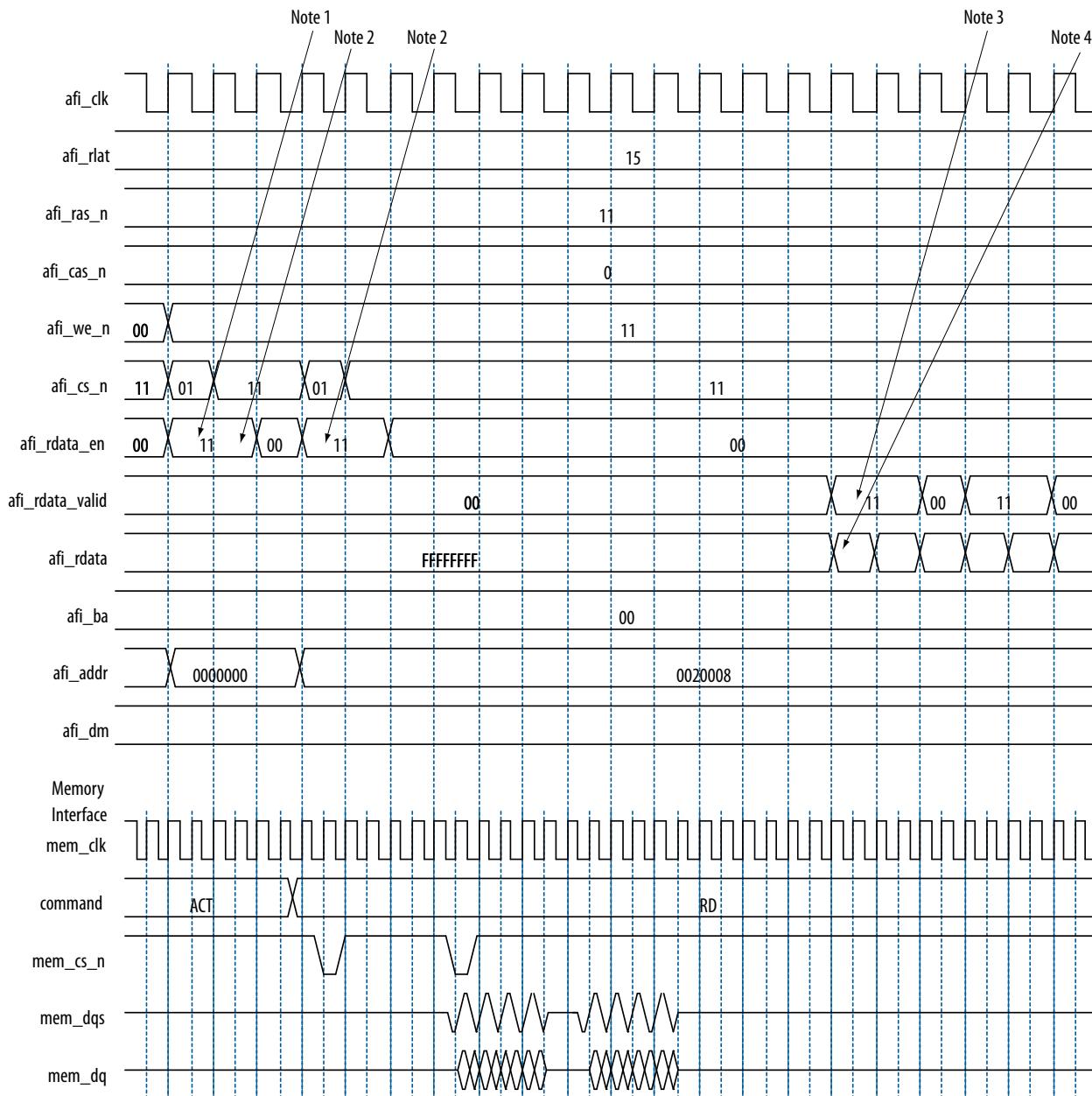
Figure 1-15: Word-Aligned Writes



must drive `afi_cs_n` and then wait `afi_wlat` (two in this example) `afi_clks` before driving `afi_wdata_valid`.

4. Observe the ordering of write data (`afi_wdata`). Compare this to data on the `mem_dq` signal.
5. In all waveforms a command record is added that combines the memory pins `ras_n`, `cas_n` and `we_n` into the current command that is issued. This command is registered by the memory when chip select (`mem_cs_n`) is low. The important commands in the presented waveforms are WR= write, ACT = activate.

**Figure 1-16: Word-Aligned Reads**



Notes to Figure:

1. For AFI, `afi_rdata_en` is required to be asserted one memory clock cycle before chip select (`afi_cs_n`) is asserted. In the half-rate `afi_clk` domain, this requirement manifests as the controller driving 11 (as opposed to the 01) on `afi_rdata_en`.
2. AFI requires that `afi_rdata_en` is driven for the duration of the read. In this example, it is driven to 11 for two half-rate `afi_clks`, which equates to driving to 1, for the four memory clock cycles of this four-beat burst.
3. The `afi_rdata_valid` returns 15 (`afi_rlat`) controller clock (`afi_clk`) cycles after `afi_rdata_en` is asserted. Returned is when the `afi_rdata_valid` signal is observed at the output of a register within the controller. A controller can use the `afi_rlat` value to determine when to register to returned data, but this is unnecessary as the `afi_rdata_valid` is provided for the controller to use as an enable when registering read data.
4. Observe the alignment of returned read data with respect to data on the bus.

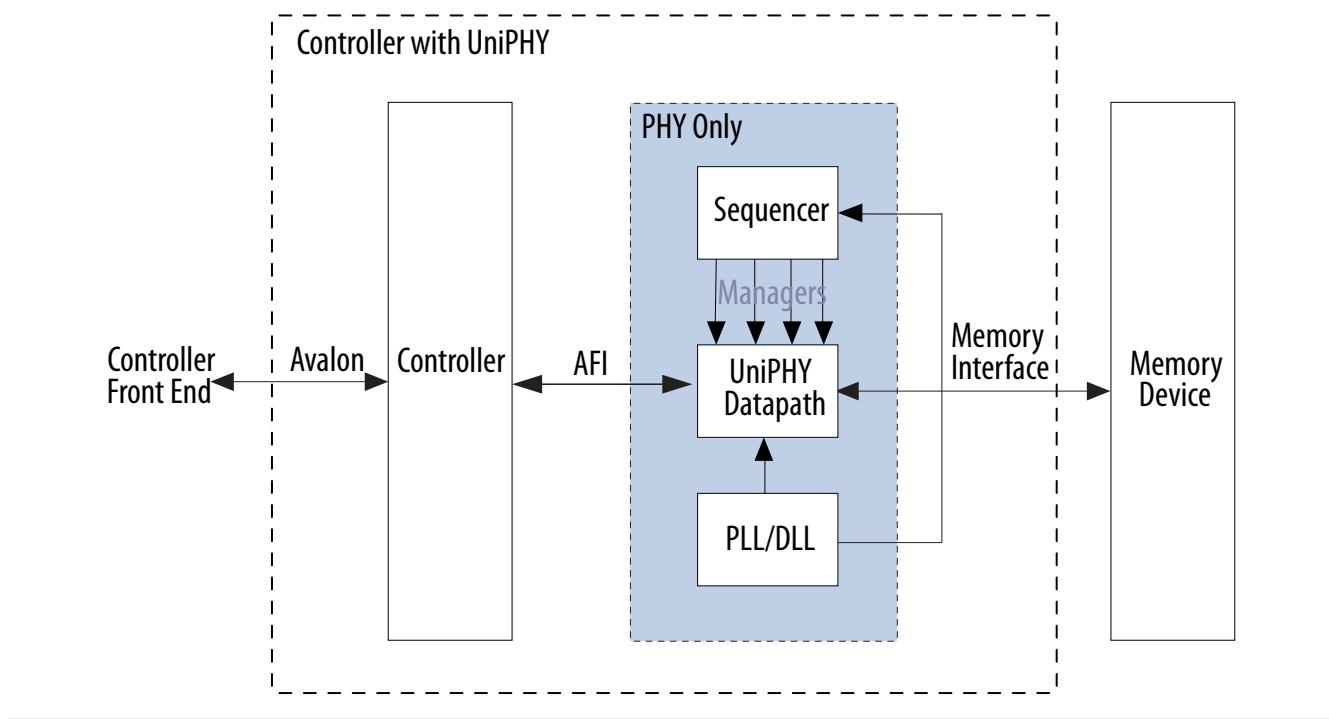
#### Related Information

- [AFI 4.0 Specification](#) on page 2-52
- [Timing Diagrams for UniPHY IP](#) on page 14-1

## Using a Custom Controller

By default, the UniPHY-based external memory interface IP cores are delivered with both the PHY and the memory controller integrated, as depicted in the following figure.

If you want to use your own custom controller with the UniPHY PHY, check the **Generate PHY only** box on the **PHY Settings** tab of the parameter editor and generate the IP. The resulting top-level IP consists of only the sequencer, UniPHY datapath, and PLL/DLL — the shaded area in the figure below.

**Figure 1-17: Memory Controller with UniPHY**

The AFI interface is exposed at the top-level of the generated IP core; you can connect the AFI interface to your custom controller.

When you enable **Generate PHY only**, the generated example designs include the memory controller appropriately instantiated to mediate read/write commands from the traffic generator to the PHY-only IP.

For information on the AFI protocol, refer to the *AFI 4.0 Specification*, in this chapter. For information on the example designs, refer to Chapter 9, *Example Designs*, in this volume.

#### Related Information

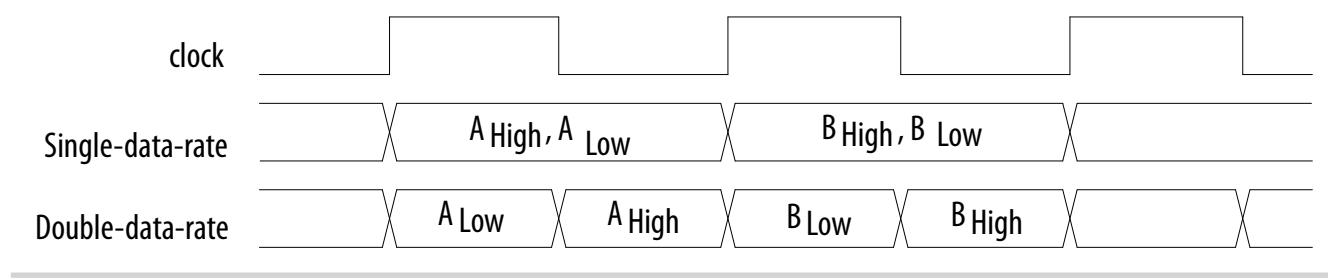
- [AFI 4.0 Specification](#) on page 2-52
- [Traffic Generator and BIST Engine](#) on page 11-21

## AFI 3.0 Specification

The Altera PHY interface (AFI) 3.0 defines communication between the controller and physical layer (PHY) in the external memory interface IP.

The AFI is a single-data-rate interface, meaning that data is transferred on the rising edge of each clock cycle. Most memory interfaces, however, operate at double-data-rate, transferring data on both the rising and falling edges of the clock signal. If the AFI interface is to directly control a double-data-rate signal, two single-data-rate bits must be transmitted on each clock cycle; the PHY then sends out one bit on the rising edge of the clock and one bit on the falling edge.

The AFI convention is to send the low part of the data first and the high part second, as shown in the following figure.

**Figure 1-18: Single Versus Double Data Rate Transfer**

## Bus Width and AFI Ratio

In cases where the AFI clock frequency is one-half or one-quarter of the memory clock frequency, the AFI data must be twice or four times as wide, respectively, as the corresponding memory data. The ratio between AFI clock and memory clock frequencies is referred to as the AFI ratio. (A half-rate AFI interface has an AFI ratio of 2, while a quarter-rate interface has an AFI ratio of 4.)

In general, the width of the AFI signal depends on the following three factors:

- The size of the equivalent signal on the memory interface. For example, if `a[15:0]` is a DDR3 address input and the AFI clock runs at the same speed as the memory interface, the equivalent `afi_addr` bus will be 16-bits wide.
- The data rate of the equivalent signal on the memory interface. For example, if `d[7:0]` is a double-data-rate QDR II input data bus and the AFI clock runs at the same speed as the memory interface, the equivalent `afi_write_data` bus will be 16-bits wide.
- The AFI ratio. For example, if `cs_n` is a single-bit DDR3 chip select input and the AFI clock runs at half the speed of the memory interface, the equivalent `afi_cs_n` bus will be 2-bits wide.

The following formula summarizes the three factors described above:

$$\text{AFI\_width} = \text{memory\_width} * \text{signal\_rate} * \text{AFI\_RATE\_RATIO}$$

**Note:** The above formula is a general rule, but not all signals obey it. For definite signal-size information, refer to the specific table.

## AFI Parameters

The following tables list Altera PHY interface (AFI) parameters for AFI 3.0.

The parameters described in the following tables affect the width of AFI signal buses. Parameters prefixed by `MEM_IF_` refer to the signal size at the interface between the PHY and memory device.

**Table 1-6: Ratio Parameters**

Parameter Name	Description
<code>AFI_RATE_RATIO</code>	The ratio between the AFI clock frequency and the memory clock frequency. For full-rate interfaces this value is 1, for half-rate interfaces the value is 2, and for quarter-rate interfaces the value is 4.
<code>DATA_RATE_RATIO</code>	The number of data bits transmitted per clock cycle. For single-data rate protocols this value is 1, and for double-data rate protocols this value is 2.

Parameter Name	Description
ADDR_RATE_RATIO	The number of address bits transmitted per clock cycle. For single-data rate address protocols this value is 1, and for double-data rate address protocols this value is 2.

**Table 1-7: Memory Interface Parameters**

Parameter Name	Description
MEM_IF_ADDR_WIDTH	The width of the address bus on the memory device(s).
MEM_IF_BANKADDR_WIDTH	The width of the bank address bus on the interface to the memory device(s). Typically, the $\log_2$ of the number of banks.
MEM_IF_CS_WIDTH	The number of chip selects on the interface to the memory device(s).
MEM_IF_WRITE_DQS_WIDTH	The number of DQS (or write clock) signals on the write interface. For example, the number of DQS groups.
MEM_IF_CLK_PAIR_COUNT	The number of CK/CK# pairs.
MEM_IF_DQ_WIDTH	The number of DQ signals on the interface to the memory device(s). For single-ended interfaces such as QDR II, this value is the number of D or Q signals.
MEM_IF_DM_WIDTH	The number of data mask pins on the interface to the memory device(s).
MEM_IF_READ_DQS_WIDTH	The number of DQS signals on the read interface. For example, the number of DQS groups.

**Table 1-8: Derived AFI Parameters**

Parameter Name	Derivation Equation
AFI_ADDR_WIDTH	MEM_IF_ADDR_WIDTH * AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_BANKADDR_WIDTH	MEM_IF_BANKADDR_WIDTH * AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_CONTROL_WIDTH	AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_CS_WIDTH	MEM_IF_CS_WIDTH * AFI_RATE_RATIO
AFI_DM_WIDTH	MEM_IF_DM_WIDTH * AFI_RATE_RATIO * DATA_RATE_RATIO
AFI_DQ_WIDTH	MEM_IF_DQ_WIDTH * AFI_RATE_RATIO * DATA_RATE_RATIO

Parameter Name	Derivation Equation
AFI_WRITE_DQS_WIDTH	MEM_IF_WRITE_DQS_WIDTH * AFI_RATE_RATIO
AFI_LAT_WIDTH	6
AFI_RLAT_WIDTH	AFI_LAT_WIDTH
AFI_WLAT_WIDTH	AFI_LAT_WIDTH * MEM_IF_WRITE_DQS_WIDTH
AFI_CLK_PAIR_COUNT	MEM_IF_CLK_PAIR_COUNT
AFI_WRANK_WIDTH	Number of ranks * MEM_IF_WRITE_DQS_WIDTH *AFI_RATE_RATIO
AFI_RRANK_WIDTH	Number of ranks * MEM_IF_READ_DQS_WIDTH *AFI_RATE_RATIO

## AFI Signals

The following tables list Altera PHY interface (AFI) signals grouped according to their functions.

In each table, the **Direction** column denotes the direction of the signal relative to the PHY. For example, a signal defined as an output passes out of the PHY to the controller. The AFI specification does not include any bidirectional signals.

Not all signals are used for all protocols.

### AFI Clock and Reset Signals

The AFI interface provides up to two clock signals and an asynchronous reset signal.

**Table 1-9: Clock and Reset Signals**

Signal Name	Direction	Width	Description
afi_clk	Output	1	Clock with which all data exchanged on the AFI bus is synchronized. In general, this clock is referred to as full-rate, half-rate, or quarter-rate, depending on the ratio between the frequency of this clock and the frequency of the memory device clock.
afi_half_clk	Output	1	Clock signal that runs at half the speed of the afi_clk. The controller uses this signal when the half-rate bridge feature is in use. This signal is optional.
afi_reset_n	Output	1	Asynchronous reset output signal. You must synchronize this signal to the clock domain in which you use it.

## AFI Address and Command Signals

The address and command signals for AFI 3.0 encode read/write/configuration commands to send to the memory device. The address and command signals are single-data rate signals.

**Table 1-10: Address and Command Signals**

Signal Name	Direction	Width	Description
afi_ba	Input	AFI_BANKADDR_WIDTH	Bank address. (Not applicable for LPDDR3.)
afi_cke	Input	AFI_CLK_EN_WIDTH	Clock enable.
afi_cs_n	Input	AFI_CS_WIDTH	Chip select signal. (The number of chip selects may not match the number of ranks; for example, RDIMMs and LRDIMMs require a minimum of 2 chip select signals for both single-rank and dual-rank configurations. Consult your memory device data sheet for information about chip select signal width.) (Matches the number of ranks for LPDDR3.)
afi_ras_n	Input	AFI_CONTROL_WIDTH	RAS# (for DDR2 and DDR3 memory devices.)
afi_we_n	Input	AFI_CONTROL_WIDTH	WE# (for DDR2, DDR3, and RLDRAM II memory devices.)
afi_cas_n	Input	AFI_CONTROL_WIDTH	CAS# (for DDR2 and DDR3 memory devices.)
afi_ref_n	Input	AFI_CONTROL_WIDTH	REF# (for RLDRAM II memory devices.)
afi_RST_n	Input	AFI_CONTROL_WIDTH	RESET# (for DDR3 and DDR4 memory devices.)
afi_odt	Input	AFI_CLK_EN_WIDTH	On-die termination signal for DDR2, DDR3, and LPDDR3 memory devices. (Do not confuse this memory device signal with the FPGA's internal on-chip termination signal.)
afi_mem_clk_disable	Input	AFI_CLK_PAIR_COUNT	When this signal is asserted, mem_clk and mem_clk_n are disabled. This signal is used in low-power mode.

Signal Name	Direction	Width	Description
afi_wps_n	Output	AFI_CS_WIDTH	WPS (for QDR II/II+ memory devices.)
afi_rps_n	Output	AFI_CS_WIDTH	RPS (for QDR II/II+ memory devices.)

## AFI Write Data Signals

Write Data Signals for AFI 3.0 control the data, data mask, and strobe signals passed to the memory device during write operations.

**Table 1-11: Write Data Signals**

Signal Name	Direction	Width	Description
afi_dqs_burst	Input	AFI_WRITE_DQS_WIDTH	Controls the enable on the strobe (DQS) pins for DDR2, DDR3, and LPDDR2 memory devices. When this signal is asserted, mem_dqs and mem_dqsn are driven.  This signal must be asserted before afi_wdata_valid to implement the write preamble, and must be driven for the correct duration to generate a correctly timed mem_dqs signal.
afi_wdata_valid	Input	AFI_WRITE_DQS_WIDTH	Write data valid signal. This signal controls the output enable on the data and data mask pins.
afi_wdata	Input	AFI_DQ_WIDTH	Write data signal to send to the memory device at double-data rate. This signal controls the PHY's mem_dq output.
afi_dm	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_dm signal for DDR2, DDR3, LPDDR2 and RLDRAM II memory devices.)
afi_bws_n	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_bws_n signal for QDR II/II+ memory devices.

Signal Name	Direction	Width	Description
afi_wrrank	Input	AFI_WRANK_WIDTH	Shadow register signal. Signal indicating the rank to which the controller is writing, so that the PHY can switch to the appropriate setting. Signal timing is identical to afi_dqs_burst; that is, afi_wrrank must be asserted at the same time as afi_dqs_burst, and must be of the same duration.

## AFI Read Data Signals

Read Data Signals for AFI 3.0 control the data sent from the memory device during read operations.

**Table 1-12: Read Data Signals**

Signal Name	Direction	Width	Description
afi_rdata_en	Input	AFI_RATE_RATIO	Read data enable. Indicates that the memory controller is currently performing a read operation. This signal is held high only for cycles of relevant data (read data masking). If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rdata_en_full	Input	AFI_RATE_RATIO	Read data enable full. Indicates that the memory controller is currently performing a read operation. This signal is held high for the entire read burst. If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rdata	Output	AFI_DQ_WIDTH	Read data from the memory device. This data is considered valid only when afi_rdata_valid is asserted by the PHY.
afi_rdata_valid	Output	AFI_RATE_RATIO	Read data valid. When asserted, this signal indicates that the afi_rdata bus is valid. If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).

Signal Name	Direction	Width	Description
afi_rrank	Input	AFI_RRANK_WIDTH	Shadow register signal. Signal indicating the rank from which the controller is reading, so that the PHY can switch to the appropriate setting. Must be asserted at the same time as afi_rdata_en when issuing a read command, and once asserted, must remain unchanged until the controller issues a new read command to another rank.

## AFI Calibration Status Signals

The PHY instantiates a sequencer which calibrates the memory interface with the memory device and some internal components such as read FIFOs and valid FIFOs. The sequencer reports the results of the calibration process to the controller through the Calibration Status Signals in the AFI interface.

**Table 1-13: Calibration Status Signals**

Signal Name	Direction	Width	Description
afi_cal_success	Output	1	Asserted to indicate that calibration has completed successfully.
afi_cal_fail	Output	1	Asserted to indicate that calibration has failed.
afi_cal_req	Input	1	Effectively a synchronous reset for the sequencer. When this signal is asserted, the sequencer returns to the reset state; when this signal is released, a new calibration sequence begins.
afi_wlat	Output	AFI_WLAT_WIDTH	The required write latency in afi_clk cycles, between address/command and write data being issued at the PHY/controller interface. The afi_wlat value can be different for different groups; each group's write latency can range from 0 to 63. If write latency is the same for all groups, only the lowest 6 bits are required.
afi_rlat (1)	Output	AFI_RLAT_WIDTH	The required read latency in afi_clk cycles between address/command and read data being returned to the PHY/controller interface. Values can range from 0 to 63.

Note to Table:

1. The afi\_rlat signal is not supported for PHY-only designs. Instead, you can sample the afi\_rdata\_valid signal to determine when valid read data is available.

## AFI Tracking Management Signals

When tracking management is enabled, the sequencer can take control over the AFI 3.0 interface at given intervals, and issue commands to the memory device to track the internal DQS Enable signal alignment to the DQS signal returning from the memory device. The tracking management portion of the AFI 3.0 interface provides a means for the sequencer and the controller to exchange handshake signals.

**Table 1-14: Tracking Management Signals**

Signal Name	Direction	Width	Description
afi_ctl_refresh_done	Input	MEM_IF_CS_WIDTH	Handshaking signal from controller to tracking manager, indicating that a refresh has occurred and waiting for a response.
afi_seq_busy	Output	MEM_IF_CS_WIDTH	Handshaking signal from sequencer to controller, indicating when DQS tracking is in progress.
afi_ctl_long_idle	Input	MEM_IF_CS_WIDTH	Handshaking signal from controller to tracking manager, indicating that it has exited low power state without a periodic refresh, and waiting for response.

## Register Maps

The following table lists the overall register mapping for the DDR2, DDR3, and LPDDR2 SDRAM Controllers with UniPHY.

**Note:** Addresses shown in the table are 32-bit word addresses. If a byte-addressed master such as a Nios II processor accesses the CSR, it is necessary to multiply the addresses by four.

**Table 1-15: Register Map**

Address	Description
<b>UniPHY Register Map</b>	
0x001	Reserved.
0x004	UniPHY status register 0.
0x005	UniPHY status register 1.
0x006	UniPHY status register 2.
0x007	UniPHY memory initialization parameters register 0.

### Controller Register Map

Address	Description
0x100	Reserved.
0x110	Controller status and configuration register.
0x120	Memory address size register 0.
0x121	Memory address size register 1.
0x122	Memory address size register 2.
0x123	Memory timing parameters register 0.
0x124	Memory timing parameters register 1.
0x125	Memory timing parameters register 2.
0x126	Memory timing parameters register 3.
0x130	ECC control register.
0x131	ECC status register.
0x132	ECC error address register.

## UniPHY Register Map

The UniPHY register map allows you to control the memory components' mode register settings. The following table lists the register map for UniPHY.

**Note:** Addresses shown in the table are 32-bit word addresses. If a byte-addressed master such as a Nios II processor accesses the CSR, it is necessary to multiply the addresses by four.

**Table 1-16: UniPHY Register Map**

Address	Bit	Name	Default	Access	Description
0x001	15:0	Reserved.	0	—	Reserved for future use.
	31:16	Reserved.	0	—	Reserved for future use.
0x002	15:0	Reserved.	0	—	Reserved for future use.
	31:16	Reserved.	0	—	Reserved for future use.

Address	Bit	Name	Default	Access	Description
0x004	0	SOFT_RESET	—	Write only	Initiate a soft reset of the interface. This bit is automatically deasserted after reset.
	23:1	Reserved.	0	—	Reserved for future use.
	24	AFI_CAL_SUCCESS	—	Read only	Reports the value of the UniPHY afi_cal_success. Writing to this bit has no effect.
	25	AFI_CAL_FAIL	—	Read only	Reports the value of the UniPHY afi_cal_fail. Writing to this bit has no effect.
	26	PLL_LOCKED	—	Read only	Reports the PLL lock status.
	31:27	Reserved.	0	—	Reserved for future use.
0x005	7:0	Reserved.	0	—	Reserved for future use.
	15:8	Reserved.	0	—	Reserved for future use.
	23:16	Reserved.	0	—	Reserved for future use.
	31:24	Reserved.	0	—	Reserved for future use.
0x006	7:0	INIT FAILING STAGE	—	Read only	<p>Initial failing error stage of calibration. Only applicable if AFI_CAL_FAIL=1.</p> <p>0: None</p> <p>1: Read Calibration - VFIFO</p> <p>2: Write Calibration - Write Leveling</p> <p>3: Read Calibration - LFIFO Calibration</p> <p>4: Write Calibration - Write Deskew</p> <p>5: Unused</p> <p>6: Refresh</p> <p>7: Calibration Skipped</p>

Address	Bit	Name	Default	Access	Description
					8: Calibration Aborted  9: Read Calibration - VFIFO After Writes
15:8		INIT_FAILING_SUBSTAGE	—	Read only	<p>Initial failing error substage of calibration. Only applicable if AFI_CAL_FAIL=1.</p> <p>If INIT_FAILING_STAGE = 1 or 9:</p> <ul style="list-style-type: none"> <li>1: Read Calibration - Guaranteed read failure</li> <li>2: Read Calibration - No working DQSen phase found</li> <li>3: Read Calibration - Per-bit read deskew failure</li> </ul> <p>If INIT_FAILING_STAGE = 2:</p> <ul style="list-style-type: none"> <li>1: Write Calibration - No first working write leveling phase found</li> <li>2: Write Calibration - No last working write leveling phase found</li> <li>3: Write Calibration - Write leveling copy failure</li> </ul> <p>If INIT_FAILING_STAGE = other, substages do not apply.</p>
23:16		INIT_FAILING_GROUP	—	Read only	<p>Initial failing error group of calibration. Only applicable if AFI_CAL_FAIL=1.</p> <p>Returns failing DQ pin instead of failing group, if:</p> <p>INIT_FAILING_STAGE=1 and INIT_FAILING_SUBSTAGE=3.</p> <p>Or</p> <p>INIT_FAILING_STAGE=4 and INIT_FAILING_SUBSTAGE=1.</p>

Address	Bit	Name	Default	Access	Description
	31:24	Reserved.	0	—	Reserved for future use.
0x007	31:0	DQS_DETECT	—	Read only	Identifies if DQS edges have been identified for each of the groups. Each bit corresponds to one DQS group.
0x008(DDR 2)	1:0	RTT_NOM	—	Read only	Rtt (nominal) setting of the DDR2 Extended Mode Register used during memory initialization.
	31:2	Reserved.	0	—	Reserved for future use.
0x008(DDR 3)	2:0	RTT_NOM	—		Rtt (nominal) setting of the DDR3 MR1 mode register used during memory initialization.
	4:3	Reserved.	0		Reserved for future use.
	6:5	ODS	—		Output driver impedance control setting of the DDR3 MR1 mode register used during memory initialization.
	8:7	Reserved.	0		Reserved for future use.
	10:9	RTT_WR	—		Rtt (writes) setting of the DDR3 MR2 mode register used during memory initialization.
	31:11	Reserved.	0		Reserved for future use.
	3:0	DS			Driver impedance control for MR3 during initialization.
	31:4	Reserved.			Reserved for future use.

## Controller Register Map

The controller register map allows you to control the memory controller settings.

**Note:** Dynamic reconfiguration is not currently supported.

For information on the controller register map, refer to *Controller Register Map*, in the *Functional Description—HPC II* chapter.

**Related Information**

- [Soft Controller Register Map](#) on page 6-35
- [Hard Controller Register Map](#) on page 6-42

## Ping Pong PHY

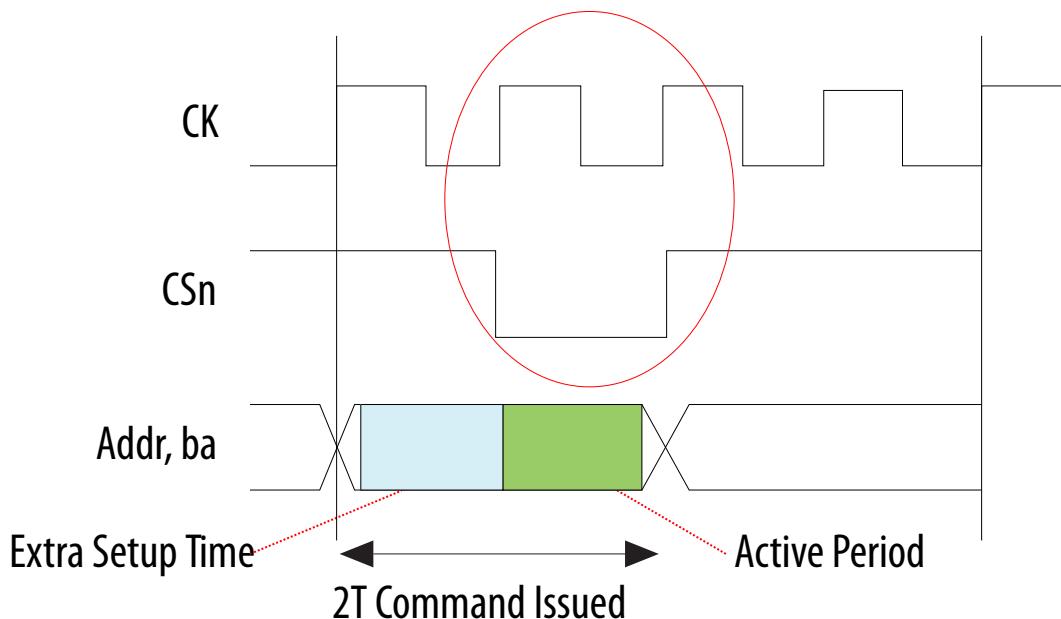
Ping Pong PHY is an implementation of UniPHY that allows two memory interfaces to share address and command buses through time multiplexing. Compared to having two independent interfaces, Ping Pong PHY uses fewer pins and less logic, while maintaining equivalent throughput.

The Ping Pong PHY supports only quarter-rate configurations of the DDR3 protocol on Arria V GZ and Stratix V devices.

### Ping Pong PHY Feature Description

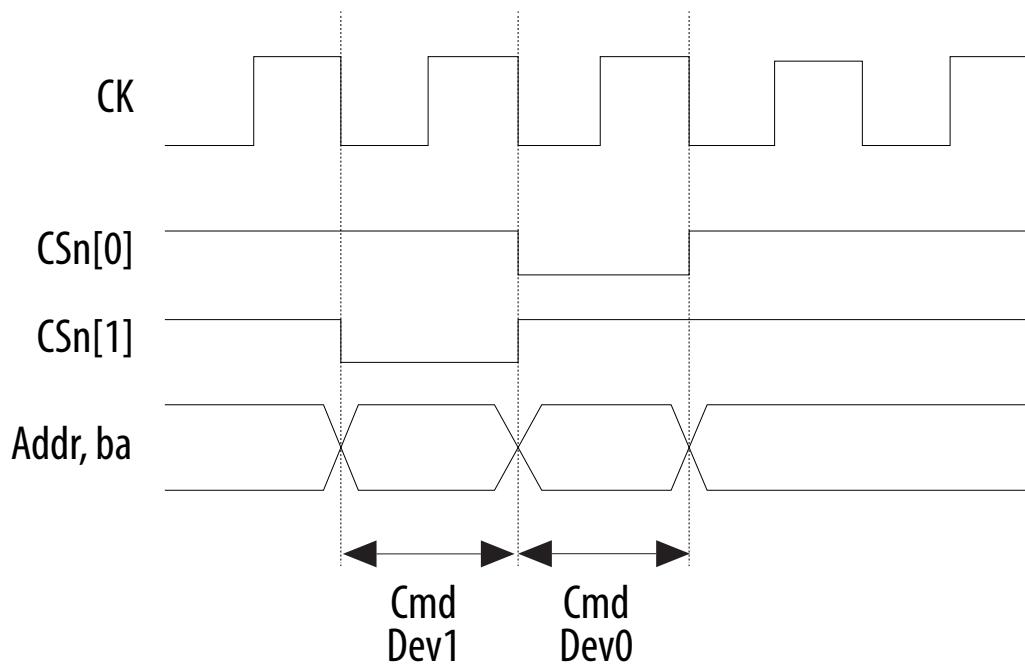
In conventional UniPHY, the address and command buses of a DDR3 quarter-rate interface use 2T time—meaning that they are issued for two full-rate clock cycles, as illustrated below.

**Figure 1-19: 2T Command Timing**



With the Ping Pong PHY, address and command signals from two independent controllers are multiplexed onto shared buses by delaying one of the controller outputs by one full-rate clock cycle. The result is 1T timing, with a new command being issued on each full-rate clock cycle. The following figure shows address and command timing for the Ping Pong PHY.

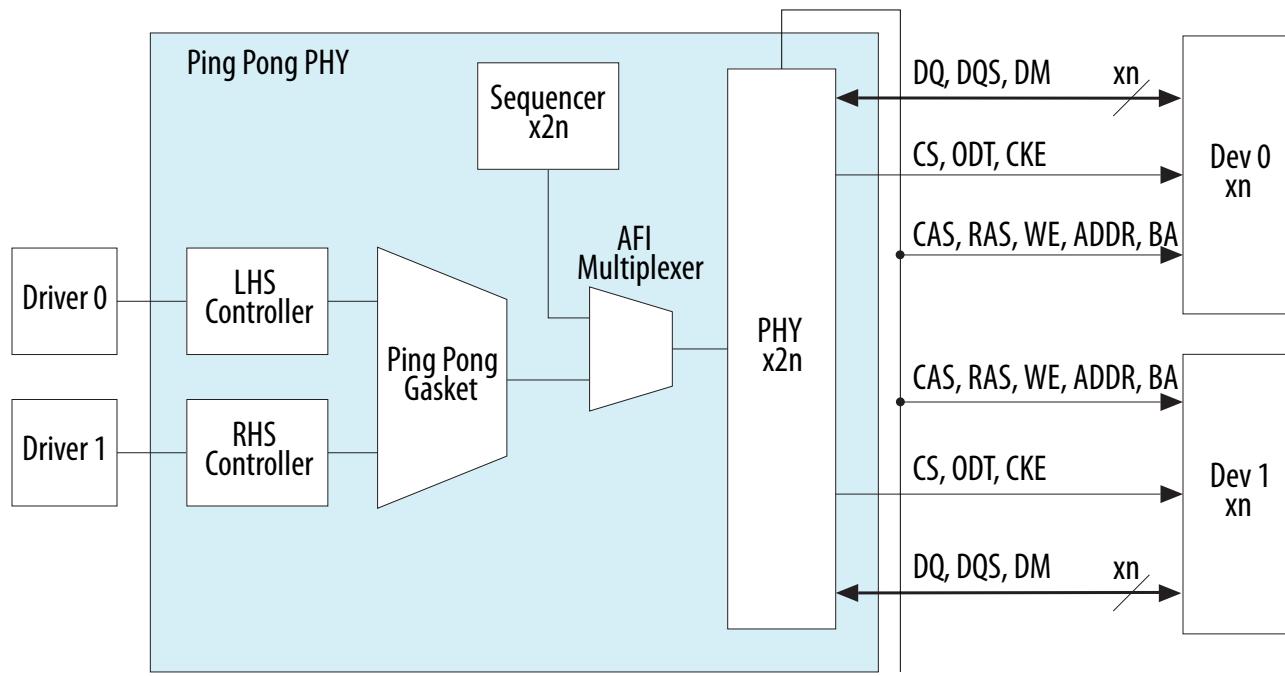
Figure 1-20: 1T Command Timing Use by Ping Pong PHY



## Ping Pong PHY Architecture

The following figure shows a top-level block diagram of the Ping Pong PHY. Functionally, the IP looks like two independent memory interfaces. The two controller blocks are referred to as right-hand side (RHS) and left-hand side (LHS), respectively. A gasket block located between the controllers and the PHY merges the AFI signals. The PHY is double data width and supports both memory devices. The sequencer is the same as with regular UniPHY, and calibrates the entire double-width PHY.

Figure 1-21: Ping Pong PHY Architecture

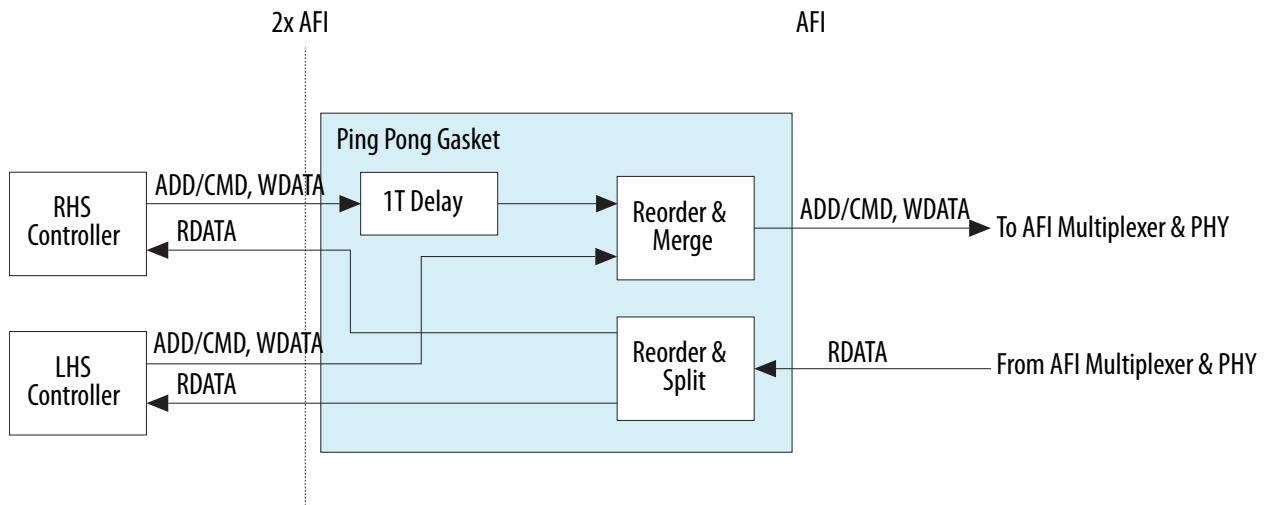


### Ping Pong Gasket

The gasket delays and remaps quarter-rate signals so that they are correctly time-multiplexed at the full-rate PHY output. The gasket also merges address and command buses ahead of the PHY.

AFI interfaces at the input and output of the gasket provide compatibility with the PHY and with memory controllers.

Figure 1-22: Ping Pong PHY Gasket Architecture



The following table shows how the gasket processes key AFI signals.

**Table 1-17: Key AFI Signals Processed by Ping Pong PHY Gasket**

Signal	Direction(Width multiplier)	Description	Gasket Conversions
cas, ras, we, addr, ba	Controller (1x) to PHY (1x)	Address and command buses shared between devices.	Delay RHS by 1T; merge.
cs, odt, cke	Controller (1x) to PHY (2x)	Chip select, on-die termination, and clock enable, one per device.	Delay RHS by 1T; reorder, merge.
wdata, wdata_valid, dqs_burst, dm	Controller (1x) to PHY (2x)	Write datapath signals, one per device.	Delay RHS by 1T; reorder, merge.
rdata_en_rd, rdata_en_rd_full	Controller (1x) to PHY (2x)	Read datapath enable signals indicating controller performing a read operation, one per device.	Delay RHS by 1T.
rdata_rdata_valid	PHY (2x) to Controller (1x)	Read data, one per device.	Reorder; split.
cal_fail, cal_success, seq_busy, wlat, rlat	PHY (1x) to Controller (1x)	Calibration result, one per device.	Pass through.
rst_n, mem_clk_disable, ctl_refresh_done, ctl_long_idle	Controller (1x) to PHY (1x)	Reset and DQS tracking signals, one per PHY.	AND (&)
cal_req, init_req	Controller (1x) to PHY (1x)	Controller to sequencer requests.	OR ( )
wrank, rrank	Controller (1x) to PHY (2x)	Shadow register support.	Delay RHS by 1T; reorder; merge.

## Ping Pong PHY Calibration

The sequencer treats the Ping Pong PHY as a regular interface of double the width. For example, in the case of two x16 devices, the sequencer calibrates both devices together as a x32 interface. The sequencer chip select signal fans out to both devices so that they are treated as a single interface. The VFIFO calibration process is unchanged. For LFIFO calibration, the LFIFO buffer is duplicated for each interface and the worst-case read datapath delay of both interfaces is used.

## Ping Pong PHY Operation

To use the Ping Pong PHY, proceed as described below.

1. Configure a single memory interface according to your requirements.
2. Select the **Enable Ping Pong PHY** option in the **Advanced PHY Options** section of the **PHY Settings** tab in the DDR3 parameter editor.

The Quartus Prime software then replicates the interface, resulting in two memory controllers and a shared PHY, with the gasket block inserted between the controllers and PHY. The system makes the necessary modifications to top-level component connections, as well as the PHY read and write datapaths, and the AFI mux, without further input from you.

## Efficiency Monitor and Protocol Checker

The Efficiency Monitor and Protocol Checker allows measurement of traffic efficiency on the Avalon-MM bus between the controller and user logic, measures read latencies, and checks the legality of Avalon commands passed from the master. The Efficiency Monitor and Protocol Checker is available with the DDR2, DDR3, and LPDDR2 SDRAM controllers with UniPHY and the RLDARAM II Controller with UniPHY. The Efficiency Monitor and Protocol Checker is not available for QDR II and QDR II+ SRAM, or for the MAX 10 device family, or for Arria V or Cyclone V designs using the Hard Memory Controller.

### Efficiency Monitor

The Efficiency Monitor reports read and write throughput on the controller input, by counting command transfers and wait times, and making that information available to the External Memory Interface Toolkit via an Avalon slave port. This information may be useful to you when experimenting with advanced controller settings, such as command look ahead depth and burst merging.

### Protocol Checker

The Protocol Checker checks the legality of commands on the controller's input interface against Altera's Avalon interface specification, and sets a flag in a register on an Avalon slave port if an illegal command is detected.

### Read Latency Counter

The Read Latency Counter measures the minimum and maximum wait times for read commands to be serviced on the Avalon bus. Each read command is time-stamped and placed into a FIFO buffer upon arrival, and latency is determined by comparing that timestamp to the current time when the first beat of the returned read data is provided back to the master.

## Using the Efficiency Monitor and Protocol Checker

To include the Efficiency Monitor and Protocol Checker when you generate your IP core, proceed as described below.

1. On the **Diagnostics** tab in the parameter editor, turn on **Enable the Efficiency Monitor and Protocol Checker on the Controller Avalon Interface**.
2. To see the results of the data compiled by the Efficiency Monitor and Protocol Checker, use the External Memory Interface Toolkit.

For information on the External Memory Interface Toolkit, refer to *External Memory Interface Debug Toolkit*, in section 2 of this volume. For information about the Avalon interface, refer to *Avalon Interface Specifications*.

#### Related Information

- [Avalon Interface Specifications](#)
- [External Memory Interface Debug Toolkit](#) on page 15-1

## Avalon CSR Slave and JTAG Memory Map

The following table lists the memory map of registers inside the Efficiency Monitor and Protocol Checker. This information is only of interest if you want to communicate directly with the Efficiency Monitor and Protocol Checker without using the External Memory Interface Toolkit. This CSR map is not part of the UniPHY CSR map.

Prior to reading the data in the CSR, you must issue a read command to address 0x01 to take a snapshot of the current data.

**Table 1-18: Avalon CSR Slave and JTAG Memory Map**

Address	Bit	Name	Default	Access	Description
0x01	31:0	Reserved	0	Read Only	Used internally by EMIF Toolkit to identify Efficiency Monitor type. This address must be read prior to reading the other CSR contents.
0x02	31:0	Reserved	0	—	Used internally by EMIF Toolkit to identify Efficiency Monitor version.

Address	Bit	Name	Default	Access	Description
0x08	0	Efficiency Monitor reset	—	Write only	Write a 0 to reset.
	7:1	Reserved	—	—	Reserved for future use.
	8	Protocol Checker reset	—	Write only	Write a 0 to reset.
	15:9	Reserved	—	—	Reserved for future use.
	16	Start/stop Efficiency Monitor	—	Read/ Write	Starting and stopping statistics gathering.
	23:17	Reserved	—	—	Reserved for future use.
	31:24	Efficiency Monitor status	—	Read Only	bit 0: Efficiency Monitor stopped bit 1: Waiting for start of pattern bit 2: Running bit 3: Counter saturation
0x10	15:0	Efficiency Monitor address width	—	Read Only	Address width of the Efficiency Monitor.
	31:16	Efficiency Monitor data width	—	Read Only	Data Width of the Efficiency Monitor.
0x11	15:0	Efficiency Monitor byte enable	—	Read Only	Byte enable width of the Efficiency Monitor.
	31:16	Efficiency Monitor burst count width	—	Read Only	Burst count width of the Efficiency Monitor.
0x14	31:0	Cycle counter	—	Read Only	Clock cycle counter for the Efficiency Monitor. Lists the number of clock cycles elapsed before the Efficiency Monitor stopped.
0x18	31:0	Transfer counter	—	Read Only	Counts any read or write data transfer cycle.
0x1C	31:0	Write counter	—	Read Only	Counts write requests, including those during bursts.
0x20	31:0	Read counter	—	Read Only	Counts read requests.

Address	Bit	Name	Default	Access	Description
0x24	31:0	Readtotal counter	—	Read Only	Counts read requests (total burst requests).
0x28	31:0	NTC waitrequest counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to slave wait request high.
0x2C	31:0	NTC noreaddatavalid counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to slave not having read data.
0x30	31:0	NTC master write idle counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to master not issuing command, or pause in write burst.
0x34	31:0	NTC master idle counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to master not issuing command anytime.
0x40	31:0	Read latency min	—	Read Only	The lowest read latency value.
0x44	31:0	Read latency max	—	Read Only	The highest read latency value.
0x48	31:0	Read latency total [31:0]	—	Read Only	The lower 32 bits of the total read latency.
0x49	31:0	Read latency total [63:32]	—	Read Only	The upper 32 bits of the total read latency.
0x50	7:0	Illegal command	—	Read Only	Bits used to indicate which illegal command has occurred. Each bit represents a unique error.
	31:8	Reserved	—	—	Reserved for future use.

## UniPHY Calibration Stages

The DDR2, DDR3, and LPDDR2 SDRAM, QDR II and QDR II+ SRAM, and RLDRAM II Controllers with UniPHY, and the RLDRAM 3 PHY-only IP, go through several stages of calibration. Calibration information is useful in debugging calibration failures.

The section includes an overview of calibration, explanation of the calibration stages, and a list of generated calibration signals. The information in this section applies only to the Nios II-based sequencer used in the DDR2, DDR3, and LPDDR2 SDRAM Controllers with UniPHY versions 10.0 and later, and,

optionally, in the QDR II and QDR II+ SRAM and RLDRAM II Controllers with UniPHY version 11.0 and later, and the RLDRAM 3 PHY-only IP. The information in this section applies to the Arria II GZ, Arria V, Arria V GZ, Cyclone V, Stratix III, Stratix IV, and Stratix V device families.

**Note:** For QDR II and QDR II+ SRAM and RLDRAM II Controllers with UniPHY version 11.0 and later, you have the option to select either the RTL-based sequencer or the Nios II-based sequencer. Generally, choose the RTL-based sequencer when area is the major consideration, and choose the Nios II-based sequencer when performance is the major consideration.

**Note:** For RLDRAM 3, write leveling is not performed. The sequencer does not attempt to optimize margin for the  $t_{CKDK}$  timing requirement.

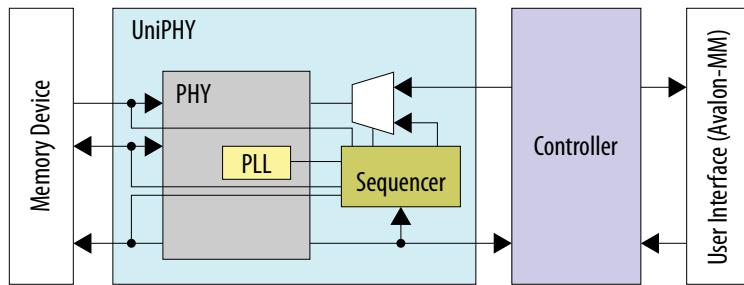
## Calibration Overview

Calibration configures the memory interface (PHY and I/Os) so that data can pass reliably to and from memory.

The sequencer illustrated in the figure below calibrates the PHY and the I/Os. To correctly transmit data between a memory device and the FPGA at high speed, the data must be center-aligned with the data clock.

Calibration also determines the delay settings needed to center-align the various data signals with respect to their clocks. I/O delay chains implement the required delays in accordance with the computed alignments. The Nios II-based sequencer performs two major tasks: FIFO buffer calibration and I/O calibration. FIFO buffer calibration adjusts FIFO lengths and I/O calibration adjusts any delay chain and phase settings to center-align data signals with respect to clock signals for both reads and writes. When the calibration process completes, the sequencer shuts off and passes control to the memory controller.

**Figure 1-23: Sequencer in Memory Interface Logic**



## Calibration Stages

The calibration process begins when the PHY reset signal deasserts and the PLL and DLL lock.

The following stages of calibration take place:

1. Read calibration part one—DQS enable calibration (only for DDR2 and DDR3 SDRAM Controllers with UniPHY) and DQ/DQS centering
2. Write calibration part one—Leveling
3. Write calibration part two—DQ/DQS centering
4. Read calibration part two—Read latency minimization

**Note:** For multirank calibration, the sequencer transmits every read and write command to each rank in sequence. Each read and write test is successful only if all ranks pass the test. The sequencer calibrates to the intersection of all ranks.

The calibration process assumes the following conditions; if either of these conditions is not true, calibration likely fails in its early stages:

- The address and command paths must be functional; calibration does not tune the address and command paths. (The Quartus Prime software fully analyzes the timing for the address and command paths, and the slack report is accurate, assuming the correct board timing parameters.)
- At least one bit per group must work before running per-bit-deskew calibration. (This assumption requires that DQ-to-DQS skews be within the recommended 20 ps.)

## Memory Initialization

The memory is powered up according to protocol initialization specifications. All ranks power up simultaneously. Once powered, the device is ready to receive mode register load commands. This part of initialization occurs separately for each rank. The sequencer issues mode register set commands on a per-chip-select basis and initializes the memory to the user-specified settings.

## Stage 1: Read Calibration Part One—DQS Enable Calibration and DQ/DQS Centering

Read calibration occurs in two parts. Part one is DQS enable calibration with DQ/DQS centering, which happens during stage 1 of the overall calibration process; part two is read latency minimization, which happens during stage 4 of the overall calibration process.

The objectives of DQS enable calibration and DQ/DQS centering are as follows:

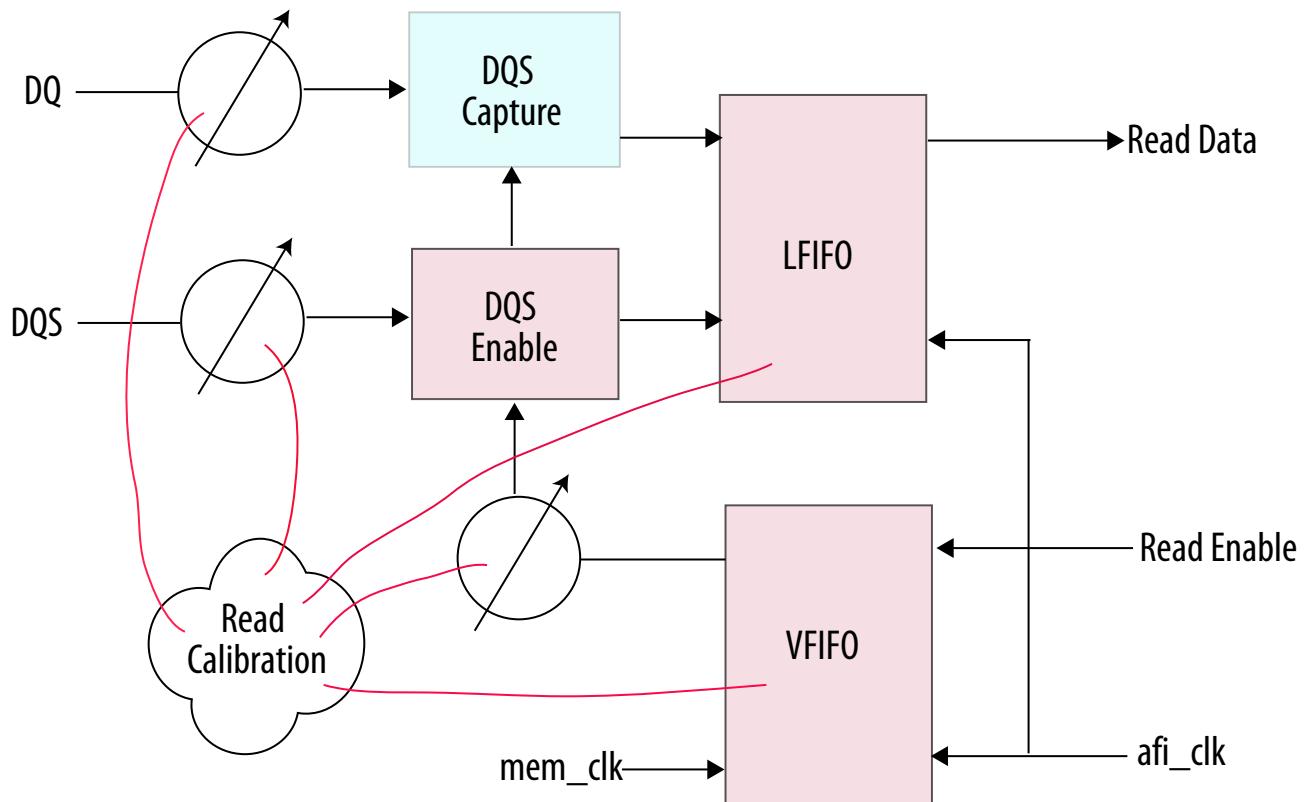
- To calculate when the read data is received after a read command is issued to setup the Data Valid Prediction FIFO (VFIFO) cycle
- To align the input data (DQ) with respect to the clock (DQS) to maximize the read margins (DDR2 and DDR3 only)

DQS enable calibration and DQ/DQS centering consists of the following actions:

- Guaranteed Write
- DQS Enable Calibration
- DQ/DQS Centering

The following figure illustrates the components in the read data path that the sequencer calibrates in this stage. (The round knobs in the figure represent configurable hardware over which the sequencer has control.)

Figure 1-24: Read Data Path Calibration Model

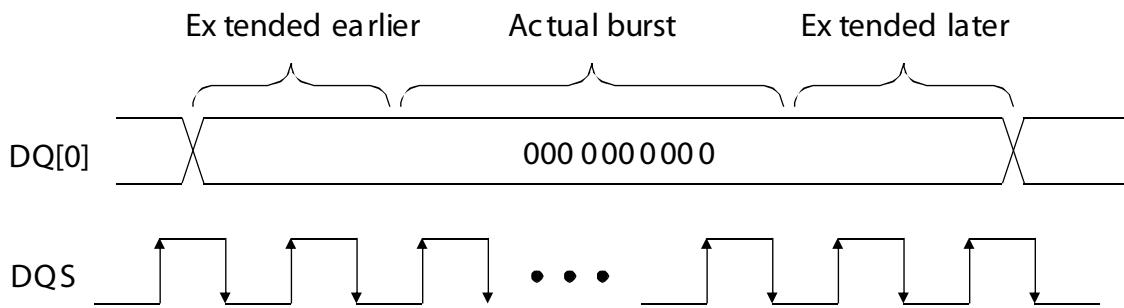


### Guaranteed Write

Because initially no communication can be reliably performed with the memory device, the sequencer uses a guaranteed write mechanism to write data into the memory device. (For the QDR II protocol, guaranteed write is not necessary, a simple write mechanism is sufficient.)

The guaranteed write is a write command issued with all data pins, all address and bank pins, and all command pins (except chip select) held constant. The sequencer begins toggling DQS well before the expected latch time at memory and continues to toggle DQS well after the expected latch time at memory. DQ-to-DQS relationship is not a factor at this stage because DQ is held constant.

Figure 1-25: Guaranteed Write of Zeros



The guaranteed write consists of a series of back-to-back writes to alternating columns and banks. For example, for DQ[0] for the DDR3 protocol, the guaranteed write performs the following operations:

- Writes a full burst of zeros to bank 0, column 0
- Writes a full burst of zeros to bank 0, column 1
- Writes a full burst of ones to bank 3, column 0
- Writes a full burst of ones to bank 3, column 1

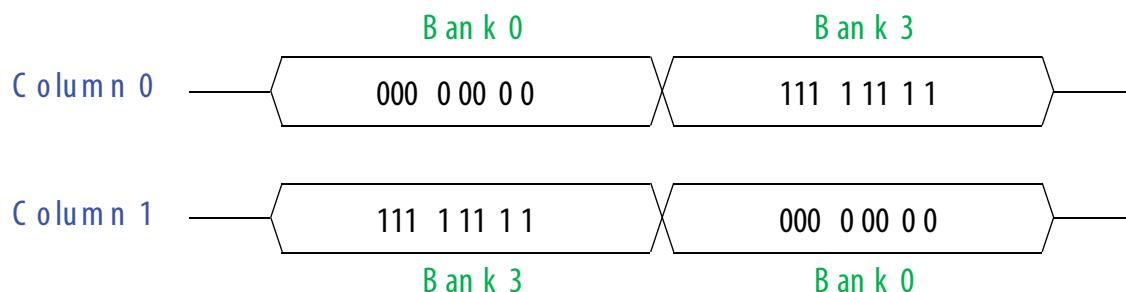
(Different protocols may use different combinations of banks and columns.)

The guaranteed write is followed by back-to-back read operations at alternating banks, effectively producing a stream of zeros followed by a stream of ones, or vice versa. The sequencer uses the zero-to-one and one-to-zero transitions in between the two bursts to identify a correct read operation, as shown in the figure below.

Although the approach described above for pin DQ[0] would work by writing the same pattern to all DQ pins, it is more effective and robust to write (and read) alternating ones and zeros to alternating DQ bits. The value of the DQ bit is still constant across the burst, and the back-to-back read mechanism works exactly as described above, except that odd DQ bits have ones instead of zeros, or vice versa.

The guaranteed write does not ensure a correct DQS-to-memory clock alignment at the memory device—DQS-to-memory clock alignment is performed later, in stage 2 of the calibration process. However, the process of guaranteed write followed by read calibration is repeated several times for different DQS-to-memory clock alignments, to ensure at least one correct alignment is found.

**Figure 1-26: Back to Back Reads on Pin DQ[0]**



## DQS Enable Calibration

DQS enable calibration ensures reliable capture of the DQ signal without glitches on the DQS line. At this point LFIFO is set to its maximum value to guarantee a reliable read from read capture registers to the core. Read latency is minimized later.

**Note:** The full DQS enable calibration is applicable only for DDR2 and DDR3 protocols; QDR II and RLDRAM protocols use only the VFIFO-based cycle-level calibration, described below.

**Note:** Delay and phase values used in this section are examples, for illustrative purposes. Your exact values may vary depending on device and configuration.

DQS enable calibration controls the timing of the enable signal using 3 independent controls: a cycle-based control (the VFIFO), a phase control, and a delay control. The VFIFO selects the cycle by shifting the controller-generated read data enable signal, `rdata_en`, by a number of full-rate clock cycles. The phase is controlled using the DLL, while the delays are adjusted using a sequence of individual delay taps. The resolution of the phase and delay controls varies with family and configuration, but is approximately 45° for the phase, and between 10 and 50 picoseconds for the delays.

The sequencer finds the two edges of the DQS enable window by searching the space of cycles, phases, and delays (an exhaustive search can usually be avoided by initially assuming the window is at least one phase wide). During the search, to test the current settings, the sequencer issues back-to-back reads from column 0 of bank 0 and bank 3, and column 1 of bank 0 and bank 3, as shown in the preceding figure. Two full bursts are read and compared with the reference data for each phase and delay setting.

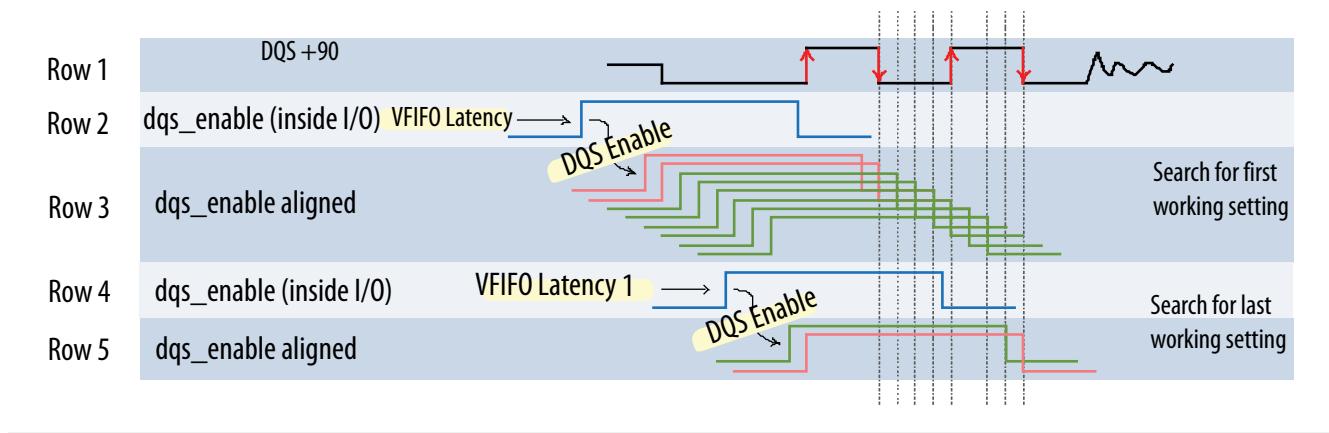
Once the sequencer identifies the two edges of the window, it center-aligns the falling edge of the DQS enable signal within the window. At this point, per-bit deskew has not yet been performed, therefore not all bits are expected to pass the read test; however, for read calibration to succeed, at least one bit per group must pass the read test.

The following figure shows the DQS and DQS enable signal relationship. The goal of DQS enable calibration is to find settings that satisfy the following conditions:

- The DQS enable signal rises before the first rising edge of DQS.
- The DQS enable signal is at one after the second-last falling edge of DQS.
- The DQS enable signal falls before the last falling edge of DQS.

The ideal position for the falling edge of the DQS enable signal is centered between the second-last and last falling edges of DQS.

**Figure 1-27: DQS and DQS Enable Signal Relationships**



The following points describe each row of the above figure:

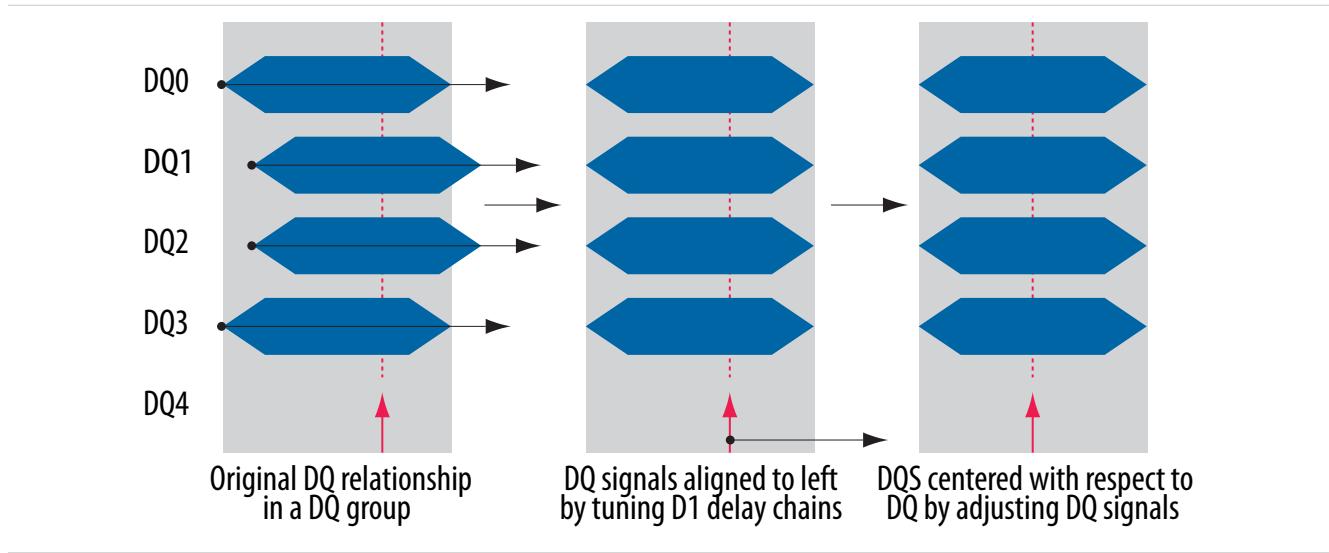
- Row 1 shows the DQS signal shifted by 90° to center-align it to the DQ data.
- Row 2 shows the raw DQS enable signal from the VFIFO.
- Row 3 shows the effect of sweeping DQS enable phases. The first two settings (shown in red) fail to properly gate the DQS signal because the enable signal turns off before the second-last falling edge of DQS. The next six settings (shown in green) gate the DQS signal successfully, with the DQS signal covering DQS from the first rising edge to the second-last falling edge.
- Row 4 shows the raw DQS enable signal from the VFIFO, increased by one clock cycle relative to Row 2.
- Row 5 shows the effect of sweeping DQS enable, beginning from the initial DQS enable of Row 4. The first setting (shown in green) successfully gates DQS, with the signal covering DQS from the first rising edge to the second-last falling edge. The second signal (shown in red), does not gate DQS successfully because the enable signal extends past the last falling edge of DQS. Any further adjustment would show the same failure.

## Centering DQ/DQS

The centering DQ/DQS stage attempts to align DQ and DQS signals on reads within a group. Each DQ signal within a DQS group might be skewed and consequently arrive at the FPGA at a different time. At this point, the sequencer sweeps each DQ signal in a DQ group to align them, by adjusting DQ input delay chains (D1).

The following figure illustrates a four DQ/DQS group per-bit-deskew and centering.

**Figure 1-28: Per-bit Deskew**



To align and center DQ and DQS, the sequencer finds the right edge of DQ signals with respect to DQS by sweeping DQ signals within a DQ group to the right until a failure occurs. In the above figure, DQ0 and DQ3 fail after six taps to the right; DQ1 and DQ2 fail after 5 taps to the right. To align the DQ signals, DQ0 and DQ3 are shifted to the right by 1 tap.

To find the center of DVW, the DQS signal is shifted to the right until a failure occurs. In the above figure, a failure occurs after 3 taps, meaning that there are 5 taps to the right edge and 3 taps to the left edge. To center-align DQ and DQS, the sequencer shifts the aligned DQ signal by 1 more tap to the right.

**Note:** The sequencer does not adjust DQS directly; instead, the sequencer center-aligns DQS with respect to DQ by delaying the DQ signals.

## Stage 2: Write Calibration Part One

The objectives of the write calibration stage are to align DQS to the memory clock at each memory device, and to compensate for address, command, and memory clock skew at each memory device. This stage is important because the address, command, and clock signals for each memory component arrive at different times.

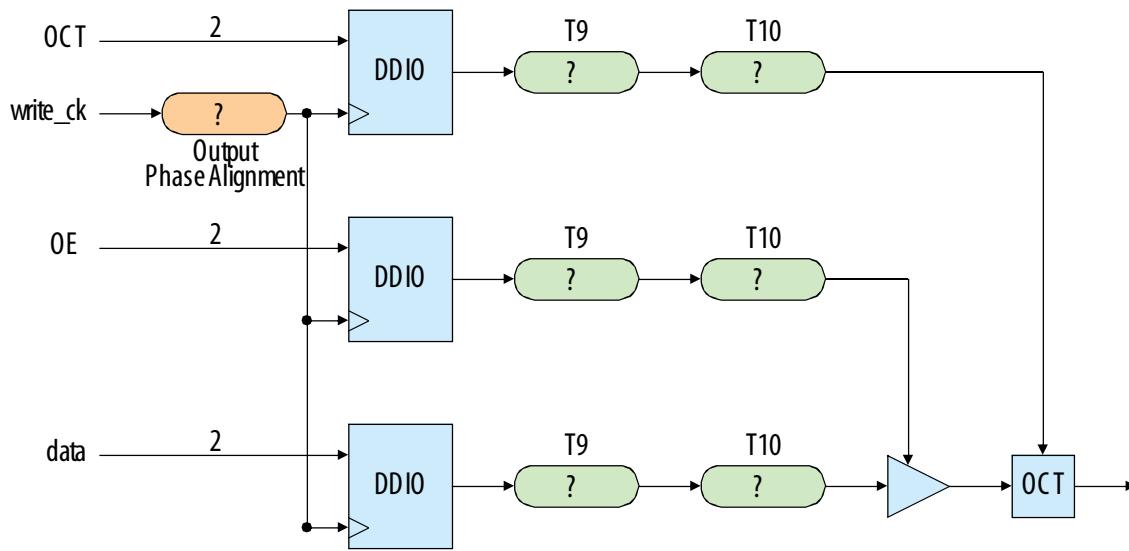
**Note:** This stage applies only to DDR2, DDR3, LPDDR2, and RLDRAM II protocols; it does not apply to the QDR II and QDR II+ protocols.

Memory clock signals and DQ/DM and DQS signals have specific relationships mandated by the memory device. The PHY must ensure that these relationships are met by skewing DQ/DM and DQS signals. The relationships between DQ/DM and DQS and memory clock signals must meet the tDQSS, tDSS, and tDSH timing constraints.

The sequencer calibrates the write data path using a variety of random burst patterns to compensate for the jitter on the output data path. Simple write patterns are insufficient to ensure a reliable write operation because they might cause imprecise DQS-to-CK alignments, depending on the actual capture circuitry on a memory device. The write patterns in the write leveling stage have a burst length of 8, and are generated by a linear feedback shift register in the form of a pseudo-random binary sequence.

The write data path architecture is the same for DQ, DM, and DQS pins. The following figure illustrates the write data path for a DQ signal. The phase coming out of the Output Phase Alignment block can be set to different values to center-align DQS with respect to DQ, and it is the same for data, OE, and OCT of a given output.

**Figure 1-29: Write Data Path**



In write leveling, the sequencer performs write operations with different delay and phase settings, followed by a read. The sequencer can implement any phase shift between 0° and 720° (depending on device and configuration). The sequencer uses the Output Phase Alignment for coarse delays and T9 and T10 for fine delays; T9 has 15 taps of 50 ps each, and T10 has 7 taps of 50 ps each.

The DQS signal phase is held at +90° with respect to DQ signal phase (Stratix IV example).

**Note:** Coarse delays are called *phases*, and fine delays are called *delays*; phases are process, voltage, and temperature (PVT) compensated, delays are not (depending on family).

For 28 nm devices:

- I/O delay chains are not PVT compensated.
- DQS input delay chain is PVT compensated.
- Leveling delay chains are PVT compensated (does not apply to Arria V or Cyclone V devices).
- T11 delay chain for postamble gating has PVT and nonPVT compensated modes, but the PVT compensated mode is not used.

**Note:** Delay and phase values used in this section are examples, for illustrative purposes. Your exact values may vary depending on device and configuration.

The sequencer writes and reads back several burst-length-8 patterns. Because the sequencer has not performed per-bit deskew on the write data path, not all bits are expected to pass the write test. However, for write calibration to succeed, at least one bit per group must pass the write test. The test begins by shifting the DQ/DQS phase until the first write operation completes successfully. The DQ/DQS signals are

then delayed to the left by D5 and D6 to find the left edge for that working phase. Then DQ/DQS phase continues the shift to find the last working phase. For the last working phase, DQ/DQS is delayed in 50 ps steps to find the right edge of the last working phase.

The sequencer sweeps through all possible phase and delay settings for each DQ group where the data read back is correct, to define a window within which the PHY can reliably perform write operations. The sequencer picks the closest value to the center of that window as the phase/delay setting for the write data path.

## Stage 3: Write Calibration Part Two—DQ/DQS Centering

The process of DQ/DQS centering in write calibration is similar to that performed in read calibration, except that write calibration is performed on the output path, using D5 and D6 delay chains.

## Stage 4: Read Calibration Part Two—Read Latency Minimization

At this stage of calibration the sequencer adjusts LFIFO latency to determine the minimum read latency that guarantees correct reads.

### Read Latency Tuning

In general, DQ signals from different DQ groups may arrive at the FPGA in a staggered fashion. In a DIMM or multiple memory device system, the DQ/DQS signals from the first memory device arrive sooner, while the DQ/DQS signals from the last memory device arrive the latest at the FPGA.

LFIFO transfers data from the capture registers in IOE to the core and aligns read data to the AFI clock. Up to this point in the calibration process, the read latency has been a maximum value set initially by LFIFO; now, the sequencer progressively lowers the read latency until the data can no longer be transferred reliably. The sequencer then increases the latency by one cycle to return to a working value and adds an additional cycle of margin to assure reliable reads.

## Calibration Signals

The following table lists signals produced by the calibration process.

Table 1-19: Calibration Signals

Signal	Description
afi_cal_fail	Asserts high if calibration fails.
afi_cal_success	Asserts high if calibration is successful.

## Calibration Time

The time needed for calibration varies, depending on many factors including the interface width, the number of ranks, frequency, board layout, and difficulty of calibration. In general, designs using the Nios II-based sequencer will take longer to calibrate than designs using the RTL-based sequencer.

The following table lists approximate typical and maximum calibration times for various protocols.

**Table 1-20: Approximate Calibration Times**

Protocol	Typical Calibration Time	Maximum Calibration Time
DDR2, DDR3, LPDDR2, RLDRAM 3	50-250 ms	Can take several minutes if the interface is difficult to calibrate, or if calibration initially fails and exhausts multiple retries.
QDR II/II+, RLDRAM II (with Nios II-based sequencer)	50-100 ms	Can take several minutes if the interface is difficult to calibrate, or if calibration initially fails and exhausts multiple retries.
QDR II/II+, RLDRAM II (with RTL-based sequencer)	<5 ms	<5 ms

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"> <li>Added statement that Efficiency Monitor and Protocol checker is not available for QDR II and QDR II+ SRAM, or for the MAX 10 device family, or for Arria V or Cyclone V designs using the Hard Memory Controller, to <i>Efficiency Monitor and Protocol Checker</i> topic.</li> </ul>
November 2015	2015.11.02	<ul style="list-style-type: none"> <li>Replaced the <i>AFI 4.0 Specification</i> with the <i>AFI 3.0 Specification</i>.</li> <li>Replaced instances of <i>Quartus II</i> with <i>Quartus Prime</i>.</li> </ul>
May 2015	2015.05.04	Maintenance release.

Date	Version	Changes
December 2014	2014.12.15	<ul style="list-style-type: none"><li>• Added several parameters to the <i>AFI Specification</i> section:<ul style="list-style-type: none"><li>• MEM_IF_BANKGROUP_WIDTH</li><li>• MEM_IF_C_WIDTH</li><li>• MEM_IF_CKE_WIDTH</li><li>• MEM_IF_ODT_WIDTH</li><li>• AFI_BANKGROUP_WIDTH</li><li>• AFI_C_WIDTH</li><li>• AFI_CKE_WIDTH</li><li>• AFI_ODT_WIDTH</li></ul></li><li>• Added several signals to the <i>AFI Specification</i> section:<ul style="list-style-type: none"><li>• afi_addr</li><li>• afi_bg</li><li>• afi_c_n</li><li>• afi_rw_n</li><li>• afi_act_n</li><li>• afi_par</li><li>• afi_alert_n</li><li>• afi_ainv</li><li>• afi_dinv</li></ul></li><li>• Changed the <i>Width</i> information for several signals in the <i>AFI Specification</i> section:<ul style="list-style-type: none"><li>• afi_dqs_burst</li><li>• afi_wdata_valid</li><li>• afi_stl_refresh_done</li><li>• afi_seq_busy</li><li>• afi_ctl_long_idle</li></ul></li></ul>
August 2014	2014.08.15	<ul style="list-style-type: none"><li>• Added note about 32-bit word addresses to <i>Register Maps</i> and <i>UniPHY Register Map</i> tables.</li><li>• Changed register map information for address 0x004, bit 26, in <i>UniPHY Register Map</i> table.</li></ul>

Date	Version	Changes
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>Removed references to HardCopy.</li> <li>Removed <i>DLL Offset Control Block</i>.</li> <li>Removed references to SOPC Builder.</li> <li>Increased minimum recommended pulse width for <i>global_reset_n</i> signal to 100ns.</li> <li>Corrected terminology inconsistency.</li> <li>Added information explaining PVT compensation.</li> <li>Added quarter-rate information to <i>PHY-to-Controller Interfaces</i> section.</li> <li>Expanded descriptions of <i>INIT FAILING STAGE</i> and <i>INIT FAILING SUBSTAGE</i> in UniPHY Register map.</li> <li>Added footnote about <i>afi_rlat</i> signal to <i>Calibration Status Signals</i> table.</li> </ul>
November 2012	3.1	<ul style="list-style-type: none"> <li>Moved <i>Controller Register Map</i> to <i>Functional Description—HPC II Controller</i> chapter.</li> <li>Updated Sequencer States information in Table 1–2.</li> <li>Enhanced <i>Using a Custom Controller</i> information.</li> <li>Enhanced <i>Tracking Manager</i> information.</li> <li>Added Ping Pong PHY information.</li> <li>Added RLDRAM 3 support.</li> <li>Added LRDIMM support.</li> <li>Added Arria V GZ support.</li> </ul>
June 2012	3.0	<ul style="list-style-type: none"> <li>Added <i>Shadow Registers</i> section.</li> <li>Added LPDDR2 support.</li> <li>Added new AFI signals.</li> <li>Added <i>Calibration Time</i> section.</li> <li>Added Feedback icon.</li> </ul>
November 2011	2.1	<ul style="list-style-type: none"> <li>Consolidated UniPHY information from <i>11.0 DDR2 and DDR3 SDRAM Controller with UniPHY User Guide</i>, <i>QDR II and QDR II+ SRAM Controller with UniPHY User Guide</i>, and <i>RLDRAM II Controller with UniPHY IP User Guide</i>.</li> <li>Revised <i>Reset and Clock Generation</i> and <i>Dedicated Clock Networks</i> sections.</li> <li>Revised Figure 1–3 and Figure 1–5.</li> <li>Added Tracking Manager to <i>Sequencer</i> section.</li> <li>Revised <i>Interfaces</i> section for DLL, PLL, and OCT sharing interfaces.</li> <li>Revised <i>Using a Custom Controller</i> section.</li> <li>Added <i>UniPHY Calibration Stages</i> section; reordered stages 3 and 4, removed stage 5.</li> </ul>

# Functional Description—Arria 10 EMIF

2

2016.05.02

EMI\_RM



Subscribe



Send Feedback

Arria 10 devices can interface with external memory devices clocking at frequencies of up to 1.3 GHz. The external memory interface IP component for Arria 10 devices provides a single parameter editor for creating external memory interfaces, regardless of memory protocol. Unlike earlier EMIF solutions which used protocol-specific parameter editors to create memory interfaces via a complex RTL generation method, the Arria 10 EMIF solution captures the protocol-specific hardened EMIF logic of the Arria 10 device together with more generic soft logic.

The Arria 10 EMIF solution is designed with the following implementations in mind:

## Hard Memory Controller and Hard PHY

This implementation provides a complete external memory interface, based on the hard memory controller and hard PHY that are part of the Arria 10 silicon. An Avalon-MM interface is available for integration with user logic.

## Soft Memory Controller and Hard PHY

This implementation provides a complete external memory interface, using an Altera-provided soft-logic-based memory controller and the hard PHY that is part of the Arria 10 silicon. An Avalon-MM interface is available for integration with user logic.

## Custom Memory Controller and Hard PHY (PHY only)

This implementation provides access to the Altera AFI interface, to allow use of a custom or third-party memory controller with the hard PHY that is part of the Arria 10 silicon. Because only the PHY component is provided by Altera, this configuration is also known as *PHY only*.

## Related Information

- [Arria 10 EMIF Architecture: Introduction](#) on page 2-3
- [Hardware Resource Sharing Among Multiple EMIFs](#) on page 2-15
- [Arria 10 EMIF IP Component](#) on page 2-18
- [Compiling Arria 10 EMIF IP with the Quartus Prime Software](#) on page 2-39
- [Debugging Arria 10 EMIF IP](#) on page 2-40
- [Arria 10 Core Fabric and General Purpose I/Os Handbook](#)

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

## Supported Memory Protocols

The following table lists the external memory protocols supported by Arria 10 devices.

**Table 2-1: Supported Memory Protocols**

Protocol	Hard Controller and Hard PHY	Soft Controller and Hard PHY	PHY Only
DDR4	Yes	—	Yes
DDR3	Yes	—	Yes
LPDDR3	Yes	—	Yes
RDRAM 3	—	Third party	Yes
QDR II/II+/II+ Xtreme	—	Yes	—
QDR-IV	—	Yes	—

Memory protocols not listed above are not supported by the Arria 10 EMIF IP; however, you can implement a custom memory interface for these protocols using the Altera PHYLite Megafunction.

LPDDR3 is supported for simulation, compilation, and timing. Hardware support for LPDDR3 will be provided in a future release.

**Note:** To achieve maximum top-line spec performance for a DDR4 interface, both read data bus inversion and periodic OCT calibration must be enabled. For maximum top-line spec performance for a QDR-IV interface, read data bus inversion must be enabled.

### Related Information

[Altera PHYLite for Memory Megafunction User Guide](#)

## Key Differences Compared to UniPHY IP and Previous Device Families

The Arria 10 EMIF IP has a new design which bears several notable differences compared to UniPHY-based IP. If you are familiar with the UniPHY-based IP, you should review the following differences, as they affect the way you generate, instantiate, and use the Arria 10 EMIF IP.

- Unlike the UniPHY-based IP, which presents a protocol-specific parameter editor for each supported memory protocol, the Arria 10 EMIF IP uses one parameter editor for all memory protocols.
- With UniPHY-based IP, you must run the `<variation_name>_pin_assignments.tcl` script following synthesis, to apply I/O assignments to the project's `.qsf` file. In Arria 10 EMIF IP, the `<variation_name>_pin_assignments.tcl` script is no longer necessary. All the I/O assignments are included in the generated `.qip` file, which the Quartus Prime software processes during compilation. Assignments that you make in the `.qsf` file override those in the `.qip` file.
- The Arria 10 EMIF IP includes a `<variation_name>readme.txt` file, located in the `/altera_emif_arch_nf_<version>` directory. This file contains important information about the implementation of the IP, including pin location guidelines, information on resource sharing, and signal descriptions.
- To generate the synthesis example design or the simulation example design, you need to run additional scripts after generation.

## Migrating from Previous Device Families

There is no automatic migration mechanism for external memory interface IP generated for previous device families.

To migrate an existing EMIF IP from an earlier device family to Arria 10, you must reparameterize and regenerate the IP targeting Arria 10, using either the IP Catalog or Qsys. If you attempt to recompile an existing IP generated for a previous device family, you will encounter errors in the Quartus Prime software.

UniPHY-based IP continues to be supported for previous device families.

## Arria 10 EMIF Architecture: Introduction

The Arria 10 EMIF architecture contains many new hardware features designed to meet the high-speed requirements of emerging memory protocols, while consuming the smallest amount of core logic area and power.

The following are key hardware features of the Arria 10 EMIF architecture:

### Hard Sequencer

The sequencer employs a hard Nios II processor, and can perform memory calibration for a wide range of protocols. You can share the sequencer among multiple memory interfaces of the same or different protocols.

### Hard PHY

The hard PHY in Arria 10 devices can interface with external memories running at speeds of up to 1.3 GHz. The PHY circuitry is hardened in the silicon, which simplifies the challenges of achieving timing closure and minimal power consumption.

### Hard Memory Controller

The hard memory controller reduces latency and minimizes core logic consumption in the external memory interface. The hard memory controller supports the DDR3, DDR4, and LPDDR3 memory protocols.

### PHY-Only Mode

Protocols that use a hard controller (DDR4, DDR3, and LPDDR3) as well as RLDRAM 3, provide a "PHY-only" option. When selected, this option generates only the PHY and sequencer, but not the controller. This PHY-Only mode provides a mechanism by which to integrate your own custom soft controller.

### High-Speed PHY Clock Tree

Dedicated high speed PHY clock networks clock the I/O buffers in Arria 10 EMIF IP. The PHY clock trees exhibit low jitter and low duty cycle distortion, maximizing the data valid window.

### Automatic Clock Phase Alignment

Automatic clock phase alignment circuitry dynamically adjust the clock phase of core clock networks to match the clock phase of the PHY clock networks. The clock phase alignment circuitry minimizes clock skew that can complicate timing closure in transfers between the FPGA core and the periphery.

### Resource Sharing

The Arria 10 architecture simplifies resource sharing between memory interfaces. Resources such as the OCT calibration block, PLL reference clock pin, and core clock can be shared. The hard Nios processor in the I/O AUX must be shared across all interfaces in a column.

#### Related Information

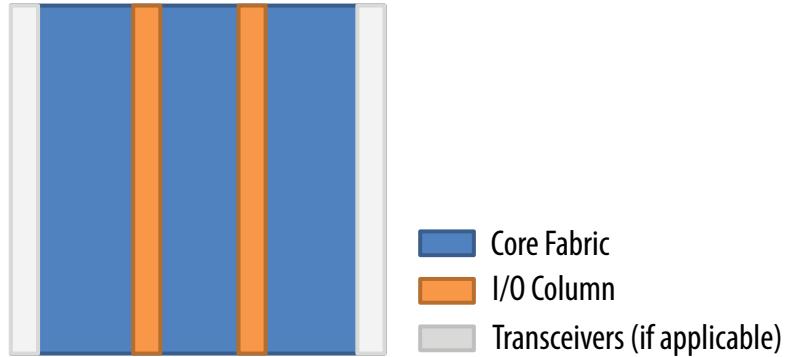
[Arria 10 Core Fabric and General Purpose I/Os Handbook](#)

## Arria 10 EMIF Architecture: I/O Subsystem

The I/O subsystem consists of two columns inside the core of Arria 10 devices.

Each column can be thought of as loosely analogous to an I/O bank.

**Figure 2-1: Arria 10 I/O Subsystem**



The I/O subsystem provides the following features:

- General-purpose I/O registers and I/O buffers
- On-chip termination control (OCT)
- I/O PLLs for external memory interfaces and user logic
- Low-voltage differential signaling (LVDS)
- External memory interface components, as follows:
  - Hard memory controller
  - Hard PHY
  - Hard Nios processor and calibration logic
  - DLL

#### Related Information

[Arria 10 Core Fabric and General Purpose I/Os Handbook](#)

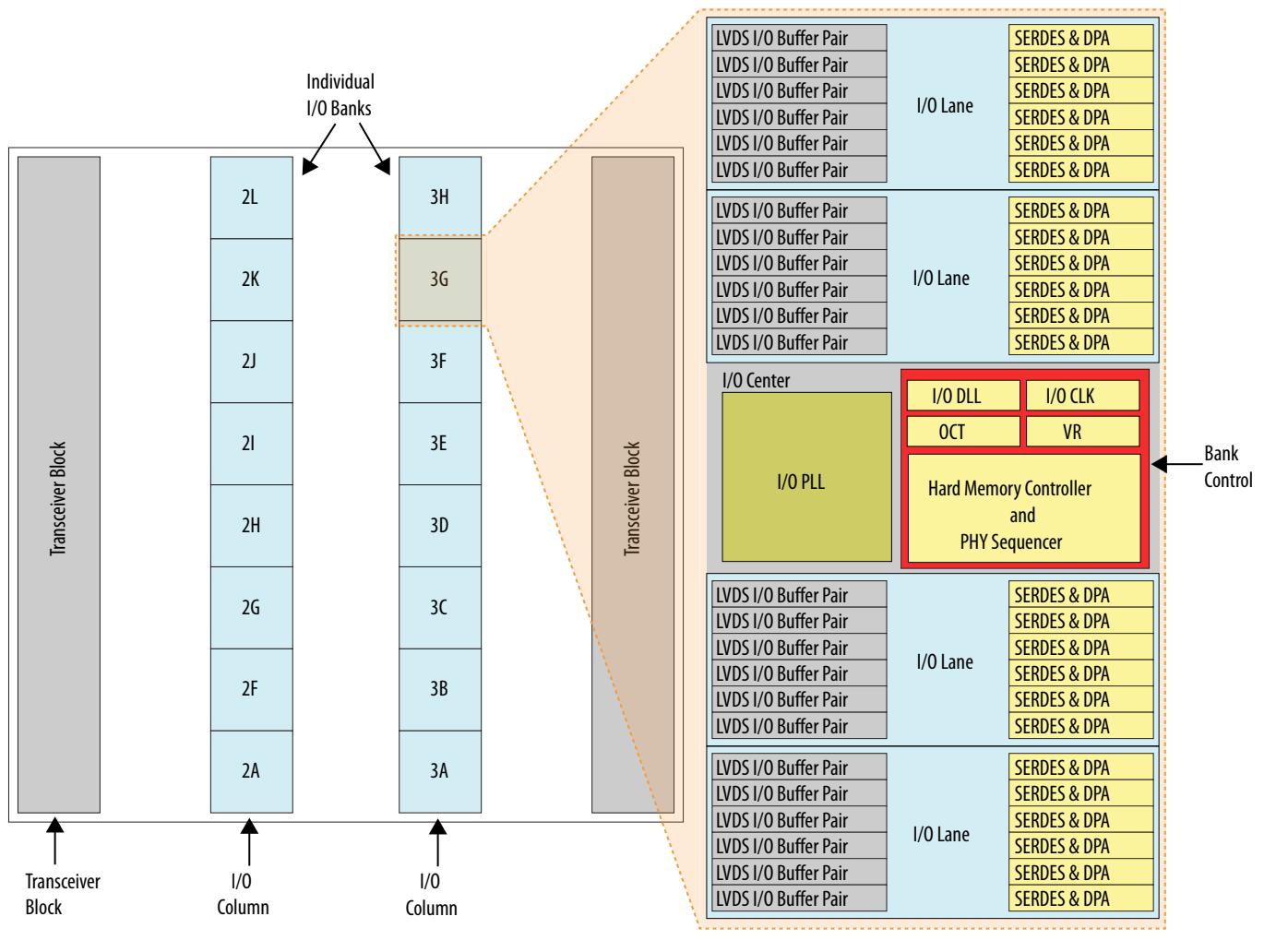
## Arria 10 EMIF Architecture: I/O Column

Arria 10 devices have two I/O columns, which contain the hardware related to external memory interfaces.

Each I/O column contains the following major parts:

- A hardened Nios processor with dedicated memory. This Nios block is referred to as the I/O AUX.
- Up to 13 I/O banks. Each I/O bank contains the hardware necessary for an external memory interface.

Figure 2-2: I/O Column



## Arria 10 EMIF Architecture: I/O AUX

Each column includes one I/O AUX, which contains a hardened Nios II processor with dedicated memory. The I/O AUX is responsible for calibration of all the EMIFs in the column.

The I/O AUX includes dedicated memory which stores both the calibration algorithm and calibration run-time data. The hardened Nios II processor and the dedicated memory can be used only by an external memory interface, and cannot be employed for any other use. The I/O AUX can interface with soft logic, such as the debug toolkit, via an Avalon-MM bus.

The I/O AUX is clocked by an on-die oscillator, and therefore does not consume a PLL.

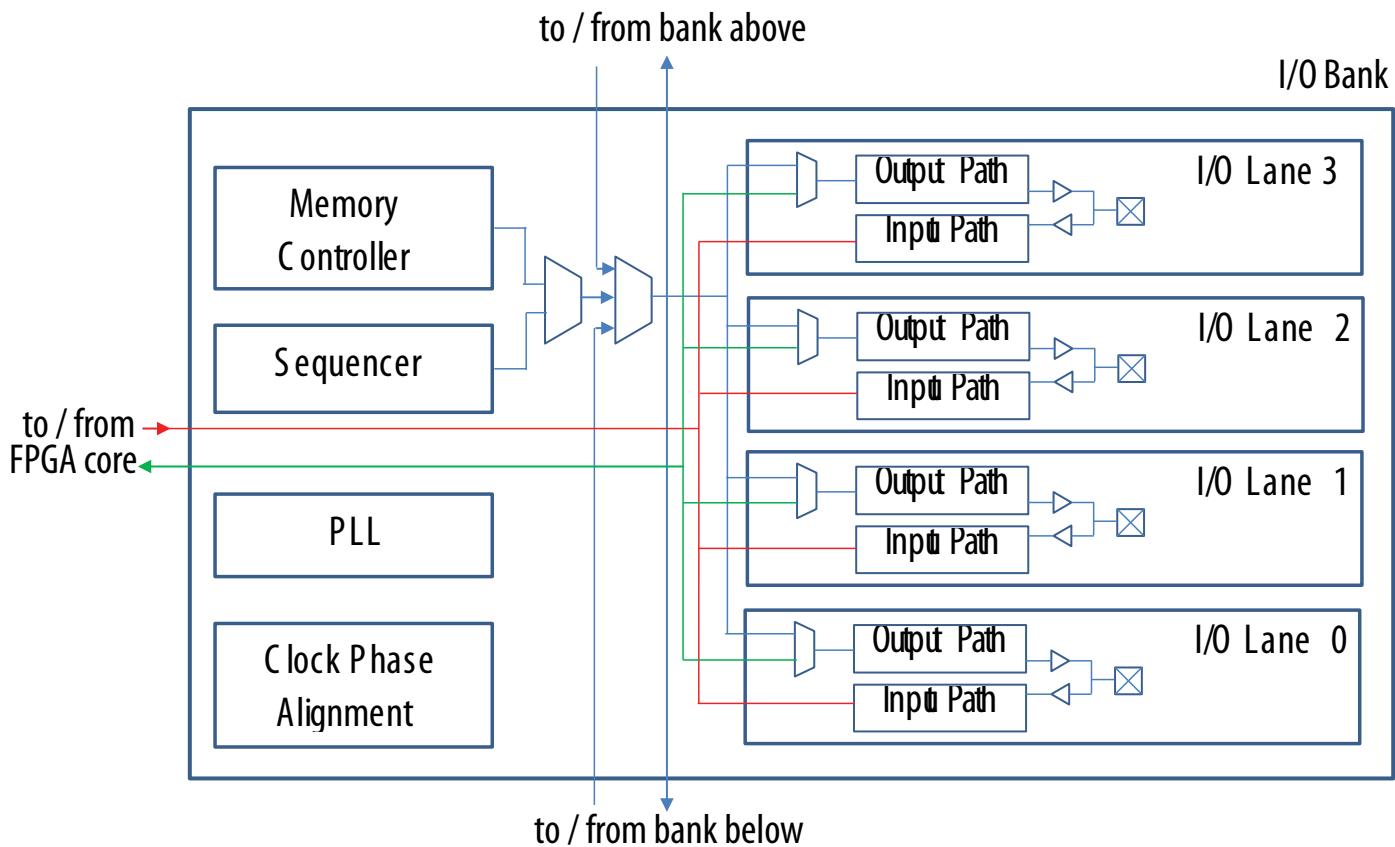
## Arria 10 EMIF Architecture: I/O Bank

A single I/O bank contains all the hardware needed to build an external memory interface. Each I/O column contains up to 13 I/O banks; the exact number of banks depends on device size and pin package. You can make a wider interface by connecting multiple banks together.

Each I/O bank resides in an I/O column, and contains the following components:

- Hard memory controller
- Sequencer components
- PLL and PHY clock trees
- DLL
- Input DQS clock trees
- 48 pins, organized into four I/O lanes of 12 pins each

**Figure 2-3: I/O Bank Architecture in Arria 10 Devices**



### I/O Bank Usage

The pins in an I/O bank can serve as address and command pins, data pins, or clock and strobe pins for an external memory interface. You can implement a narrow interface, such as a DDR3 or DDR4 x8 interface, with only a single I/O bank. A wider interface, such as x72 or x144, can be implemented by configuring multiple adjacent banks in a multi-bank interface. Any pins in a bank which are not used by the external memory interface remain available for use as general purpose I/O pins (of the same voltage standard).

Every I/O bank includes a hard memory controller which you can configure for DDR3 or DDR4. In a multi-bank interface, only the controller of one bank is active; controllers in the remaining banks are turned off to conserve power.

To use a multi-bank Arria 10 EMIF interface, you must observe the following rules:

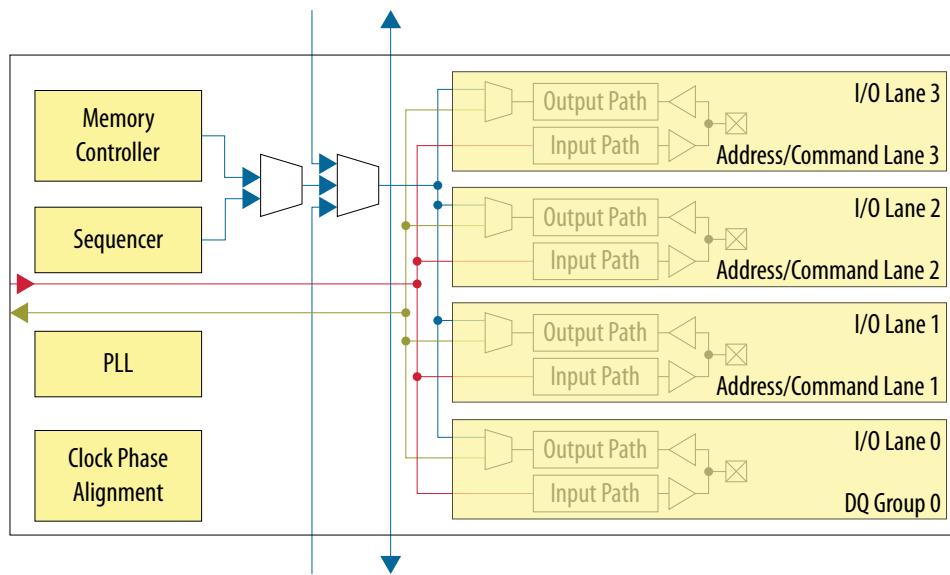
- Designate one bank as the address and command bank.
- The address and command bank must contain all the address and command pins.
- The locations of individual address and command pins within the address and command bank must adhere to the pin map defined in the pin table— regardless of whether you use the hard memory controller or not.
- If you do use the hard memory controller, the address and command bank contains the active hard controller.

All the I/O banks in a column are capable of functioning as the address and command bank. However, for minimal latency, you should select the center-most bank of the interface as the address and command bank.

### Implementing a x8 Interface with Hard Memory Controller

The following diagram illustrates the use of a single I/O bank to implement a DDR3 or DDR4 x8 interface using the hard memory controller.

**Figure 2-4: Single Bank x8 Interface With Hard Controller**



In the above diagram, shaded cells indicate resources that are in use.

**Note:** For information on the I/O lanes and pins in use, consult the pin table for your device or the `<variation_name>/altera_emif_arch_nf_140/<synth|sim>/<variation_name>_altera_emif_arch_nf_140_<unique ID>_readme.txt` file generated with your IP.

#### Related Information

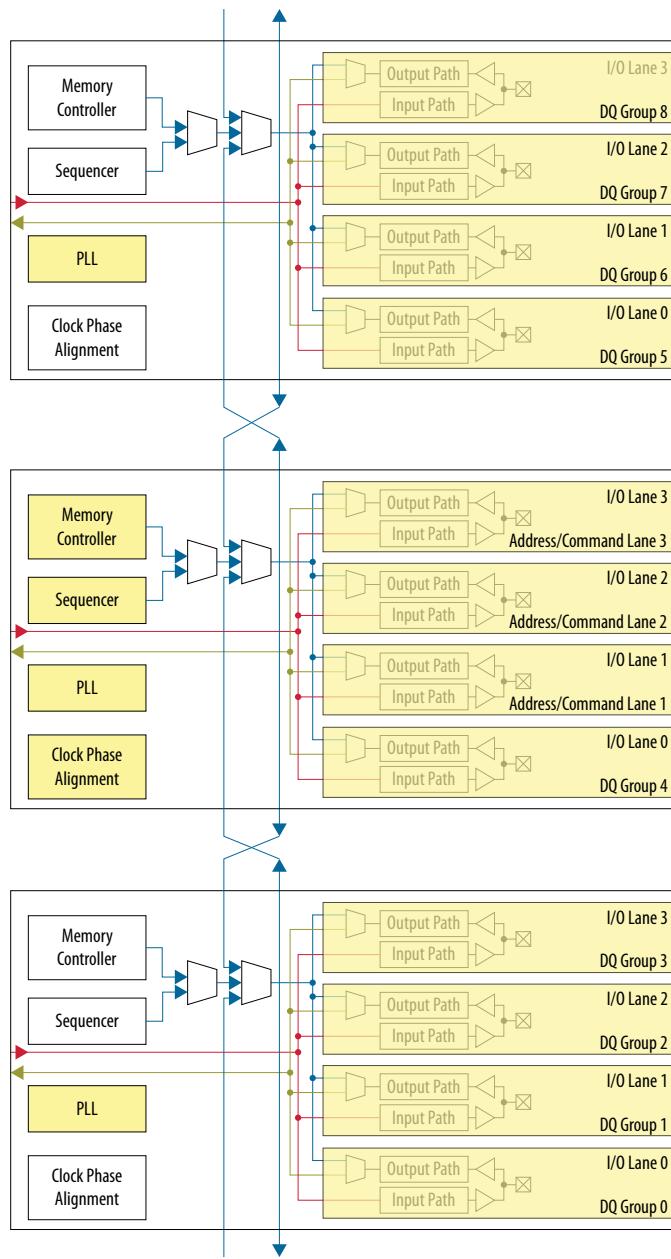
[Arria 10 Core Fabric and General Purpose I/Os Handbook](#)

### Implementing a x72 Interface with Hard Memory Controller

The following diagram illustrates one possible implementation of a DDR3 or DDR4 x72 interface using the hard memory controller.

Note that only the hard memory controller in the address and command bank is used. Similarly, only the clock phase alignment block of the address and command bank is used to generate clock signals for the FPGA core.

Figure 2-5: Multi-Bank x72 Interface With Hard Controller



In the above diagram, shaded cells indicate resources that are in use.

**Note:** For information on the I/O lanes and pins in use, consult the pin table for your device or the [`<variation\_name>/altera\_emif\_arch\_nf\_140/<synth|sim>/<variation\_name>\_altera\_emif\_arch\_nf\_140\_<unique ID>.readme.txt`](#) file generated with your IP.

**Related Information****Arria 10 Core Fabric and General Purpose I/Os Handbook****Arria 10 EMIF Architecture: I/O Lane**

An I/O bank contains 48 I/O pins, organized into four I/O lanes of 12 pins each.

Each I/O lane can implement one x8/x9 read capture group (DQS group), with two pins functioning as the read capture clock/strobe pair (DQS/DQS#), and up to 10 pins functioning as data pins (DQ and DM pins). To implement x18 and x36 groups, you can use multiple lanes within the same bank.

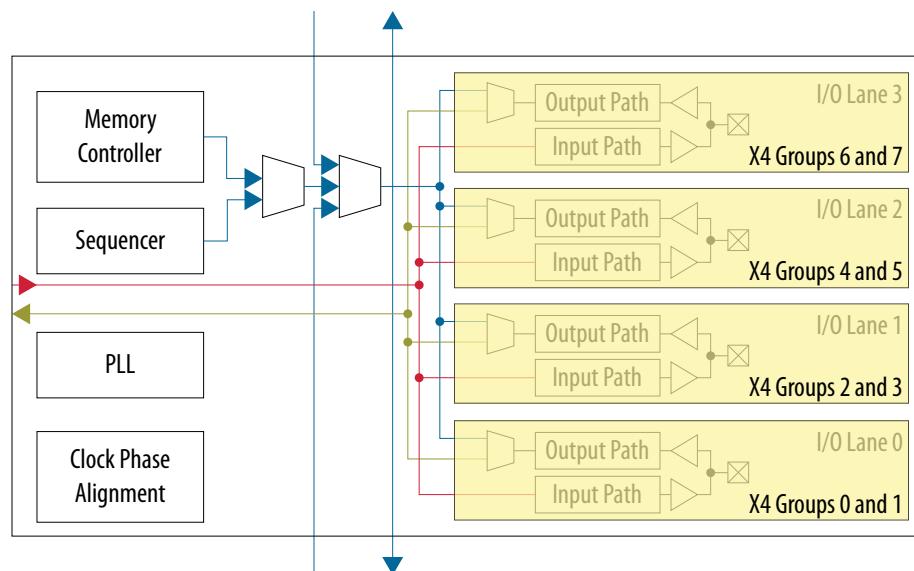
It is also possible to implement a pair of x4 groups in a lane. In this case, four pins function as clock/strobe pair, and 8 pins function as data pins. DM is not available for x4 groups. There must be an even number of x4 groups for each interface.

For x4 groups, DQS0 and DQS1 must be placed in the same I/O lane as a pair. Similarly, DQS2 and DQS3 must be paired. In general, DQS( $x$ ) and DQS( $x+1$ ) must be paired in the same I/O lane.

**Table 2-2: Lanes Used Per Group**

Group Size	Number of Lanes Used	Maximum Number of Data Pins per Group
x8 / x9	1	10
x18	2	22
x36	4	46
pair of x4	1	4 per group, 8 per lane

**Figure 2-6: x4 Group**



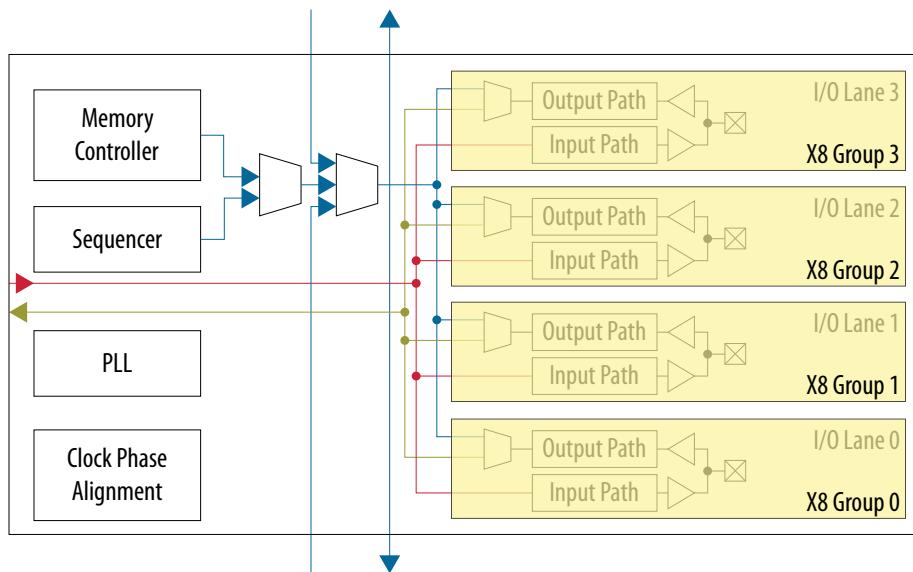
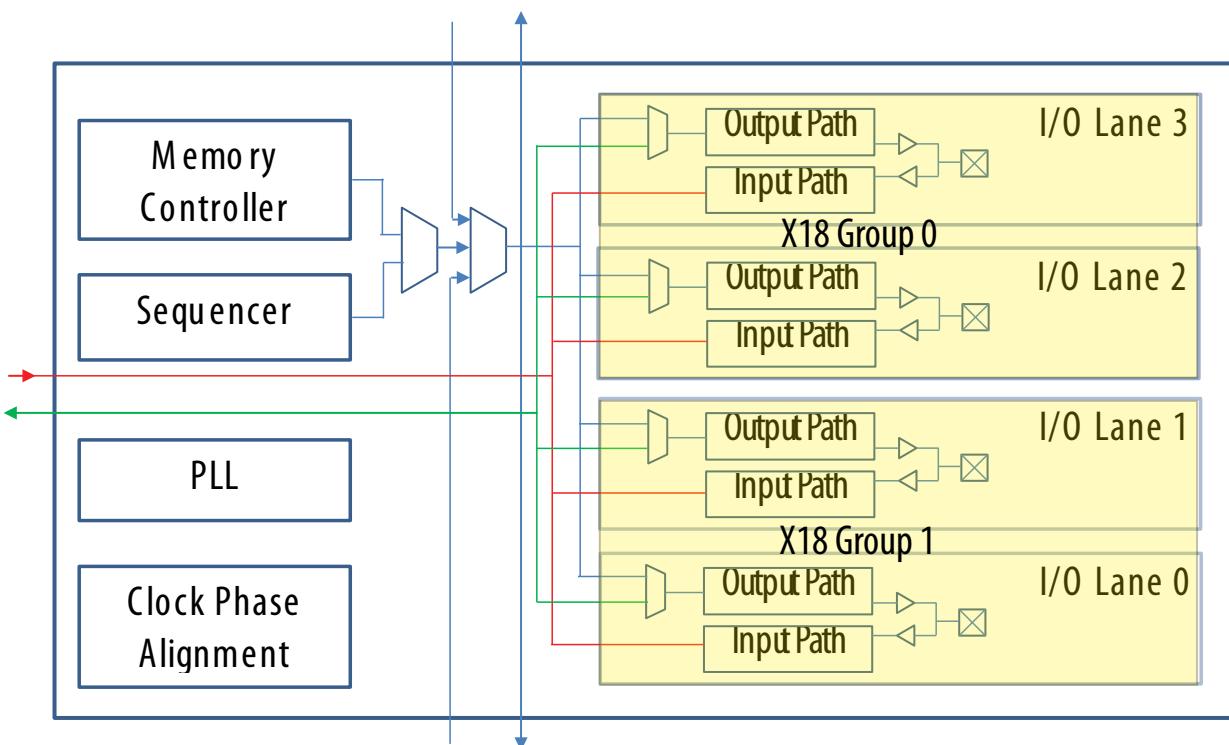
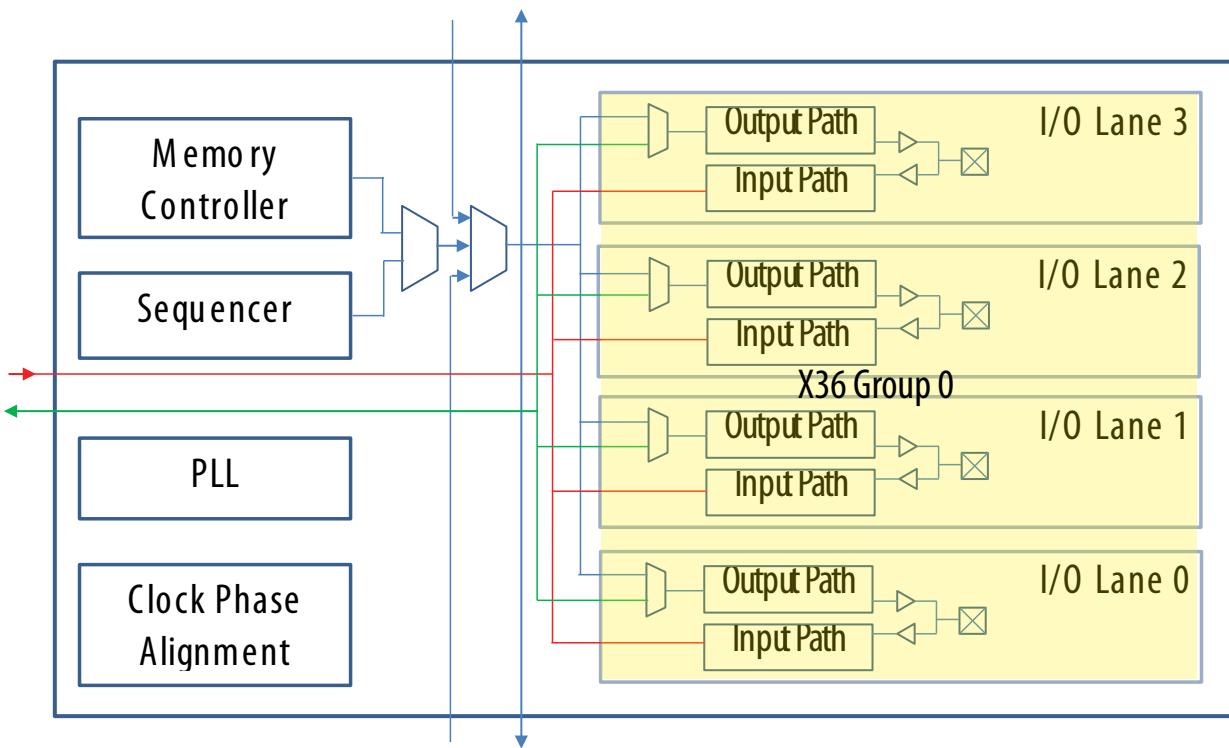
**Figure 2-7: x8 Group****Figure 2-8: x18 Group**

Figure 2-9: x36 Group



## Arria 10 EMIF Architecture: Input DQS Clock Tree

The input DQS clock tree is a balanced clock network that distributes the read capture clock and strobe from the external memory device to the read capture registers inside the I/Os.

You can configure an input DQS clock tree in x4 mode, x8/x9 mode, x18 mode, or x36 mode.

Within every bank, only certain physical pins at specific locations can drive the input DQS clock trees. The pin locations that can drive the input DQS clock trees vary, depending on the size of the group.

Table 2-3: Pins Usable as Read Capture Clock / Strobe Pair

Group Size	Index of Lanes Spanned by Clock Tree	In-Bank Index of Pins Usable as Read Capture Clock / Strobe Pair	
		Positive Leg	Negative Leg
x4	0A	4	5
x4	0B	8	9
x4	1A	16	17
x4	1B	20	21
x4	2A	28	29
x4	2B	32	33
x4	3A	40	41

Group Size	Index of Lanes Spanned by Clock Tree	In-Bank Index of Pins Usable as Read Capture Clock / Strobe Pair	
		Positive Leg	Negative Leg
x4	3B	44	45
x8 / x9	0	4	5
x8 / x9	1	16	17
x8 / x9	2	28	29
x8 / x9	3	40	41
x18	0, 1	12	13
x18	2, 3	36	37
x36	0, 1, 2, 3	20	21

## Arria 10 EMIF Architecture: PHY Clock Tree

Dedicated high-speed clock networks drive I/Os in Arria 10 EMIF. Each PHY clock network spans only one bank.

The relatively short span of the PHY clock trees results in low jitter and low duty-cycle distortion, maximizing the data valid window.

The PHY clock tree in Arria 10 devices can run as fast as 1.3 GHz. All Arria 10 external memory interfaces use the PHY clock trees.

## Arria 10 EMIF Architecture: PLL Reference Clock Networks

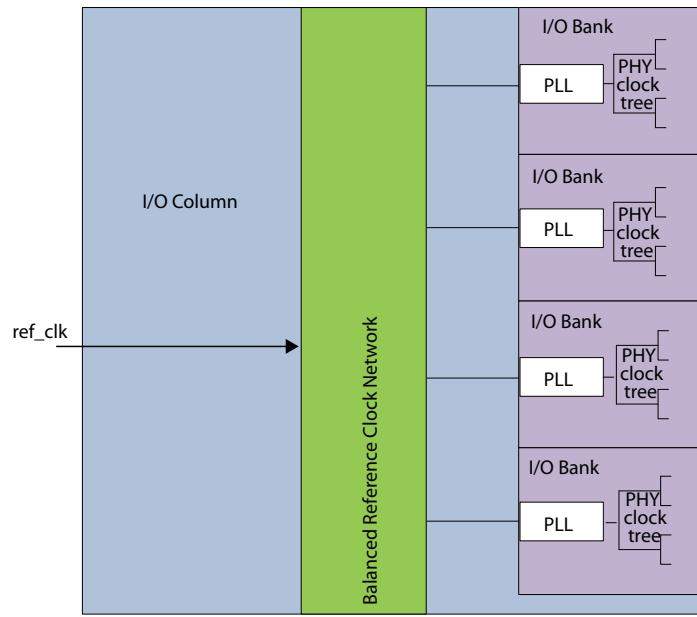
Each I/O bank includes a PLL that can drive the PHY clock trees of that bank, through dedicated connections. In addition to supporting EMIF-specific functions, such PLLs can also serve as general-purpose PLLs for user logic.

Arria 10 external memory interfaces that span multiple banks use the PLL in each bank. (Previous device families relied on a single PLL with clock signals broadcast to all I/Os via a clock network.) The Arria 10 architecture allows for relatively short PHY clock networks, reducing jitter and duty-cycle distortion.

In a multi-bank interface, the clock outputs of individual PLLs must remain in phase; this is achieved by the following mechanisms:

- A single PLL reference clock source feeds all PLLs. The reference clock signal reaches the PLLs by a balanced PLL reference clock tree. The Quartus Prime software automatically configures the PLL reference clock tree so that it spans the correct number of banks.
- The IP sets the PLL M and N values appropriately to maintain synchronization among the clock dividers across the PLLs. This requirement restricts the legal PLL reference clock frequencies for a given memory interface frequency and clock rate. The Arria 10 EMIF IP parameter editor automatically calculates and displays the set of legal PLL reference clock frequencies. If you plan to use an on-board oscillator, you must ensure that its frequency matches the PLL reference clock frequency that you select from the displayed list. The correct M and N values of the PLLs are set automatically based on the PLL reference clock frequency that you select.

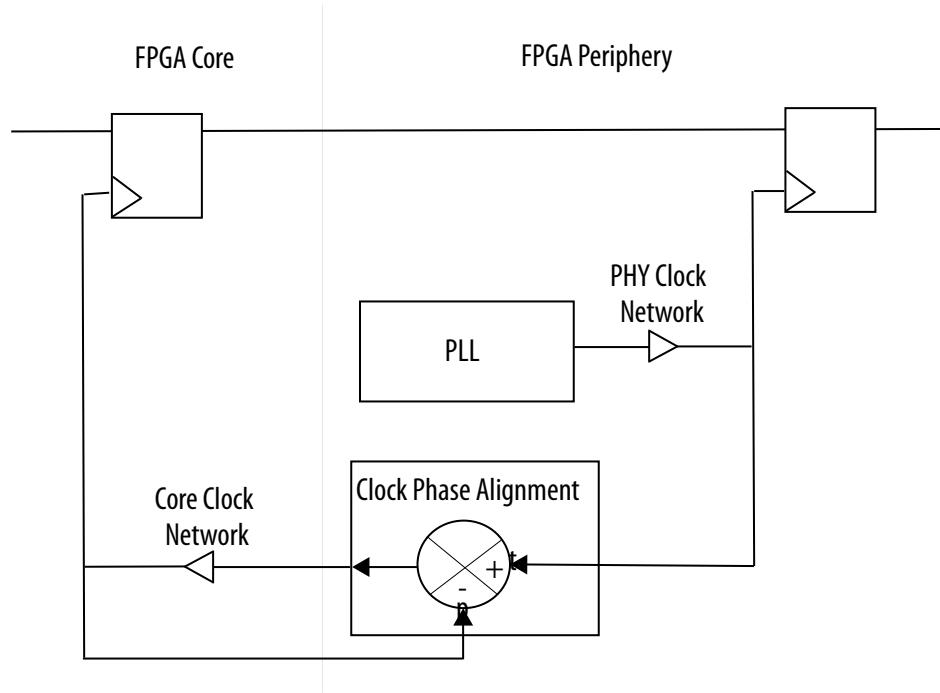
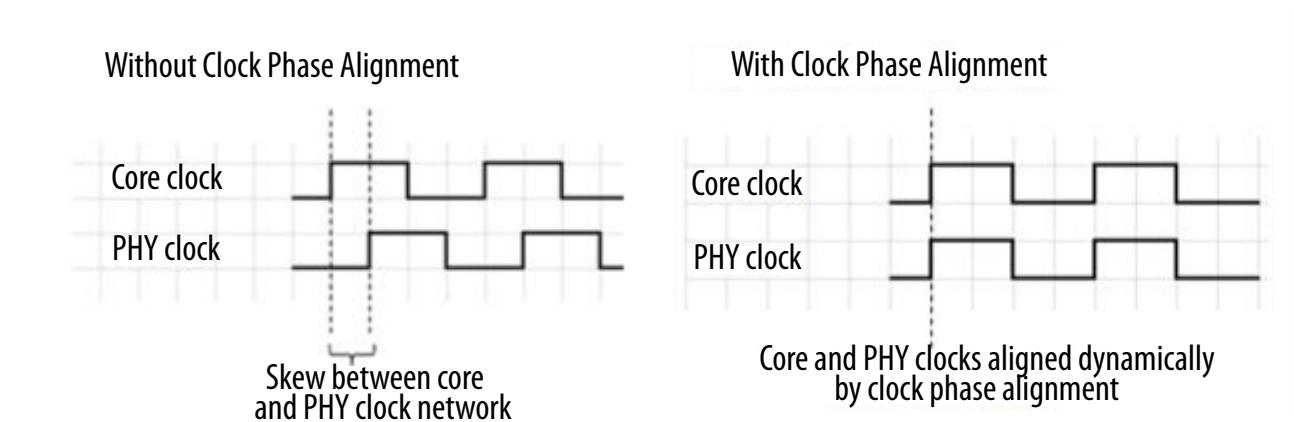
**Note:** The PLL reference clock pin may be placed in the address and command I/O bank or in a data I/O bank, there is no implication on timing.

**Figure 2-10: PLL Balanced Reference Clock Tree**

## Arria 10 EMIF Architecture: Clock Phase Alignment

In Arria 10 external memory interfaces, a global clock network clocks registers inside the FPGA core, and the PHY clock network clocks registers inside the FPGA periphery. Clock phase alignment circuitry employs negative feedback to dynamically adjust the phase of the core clock signal to match the phase of the PHY clock signal.

The clock phase alignment feature effectively eliminates the clock skew effect in all transfers between the core and the periphery, facilitating timing closure. All Arria 10 external memory interfaces employ clock phase alignment circuitry.

**Figure 2-11: Clock Phase Alignment Illustration****Figure 2-12: Effect of Clock Phase Alignment**

## Hardware Resource Sharing Among Multiple EMIFs

Often, it is necessary or desirable to share resources between interfaces.

The following topics explain which hardware resources can be shared, and provide guidance for doing so.

## I/O Aux Sharing

The I/O Aux contains a hard Nios-II processor and dedicated memory storing the calibration software code and data.

When a column contains multiple memory interfaces, the hard Nios-II processor calibrates each interface serially. Interfaces placed within the same I/O column always share the same I/O Aux. The Quartus Prime Fitter handles I/O Aux sharing automatically.

## I/O Bank Sharing

Data lanes from multiple compatible interfaces can share a physical I/O bank to achieve a more compact pin placement. To share an I/O bank, interfaces must use the same memory protocol, rate, frequency, I/O standard, and PLL reference clock signal.

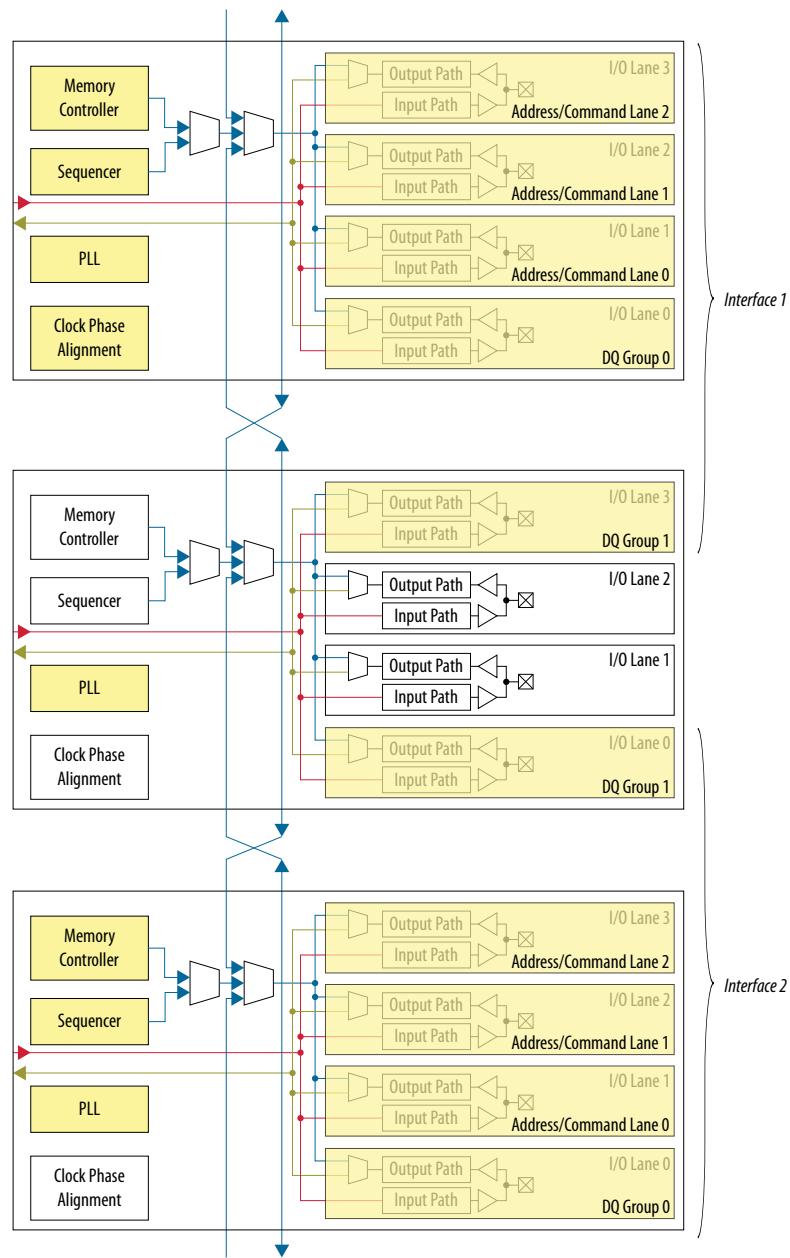
### Rules for Sharing I/O Banks

- A bank cannot serve as the address and command bank for more than one interface. This means that lanes which implement address and command pins for different interfaces cannot be allocated to the same physical bank.

**Note:** An exception to the above rule exists when two interfaces are configured in a Ping-Pong PHY fashion. In such a configuration, two interfaces share the same set of address and command pins, effectively meaning that they share the same address and command tile.

- Pins within a lane cannot be shared by multiple memory interfaces.
- Pins that are not used by EMIF IP can serve as general-purpose I/Os of compatible voltage and termination settings.
- You can configure a bank as LVDS or as EMIF, but not both at the same time.
- Interfaces that share banks must reside at consecutive bank locations.

The following diagram illustrates two x16 interfaces sharing an I/O bank. The two interfaces share the same clock phase alignment block, so that one core clock signal can interact with both interfaces. Without sharing, the two interfaces would occupy a total of four physical banks instead of three.

**Figure 2-13: I/O Bank Sharing**

## PLL Reference Clock Sharing

In Arria 10, every I/O bank contains a PLL, meaning that it is not necessary to share PLLs in the interest of conserving resources. Nonetheless, it is often desirable to share PLLs for other reasons.

You might want to share PLLs between interfaces for the following reasons:

- To conserve pins.
- When combined with the use of the balanced PLL reference clock tree, to allow the clock signals at different interfaces to be synchronous and aligned to each other. For this reason, interfaces that share core clock signals must also share the PLL reference clock signal.

To implement PLL reference clock sharing, open your RTL and connect the PLL reference clock signal at your design's top-level to the PLL reference clock port of multiple interfaces.

To share a PLL reference clock, the following requirements must be met:

- Interfaces must expect a reference clock signal of the same frequency.
- Interfaces must be placed in the same column.
- Interfaces must be placed at adjacent bank locations.

## Core Clock Network Sharing

It is often desirable or necessary for multiple memory interfaces to be accessible using a single clock domain in the FPGA core.

You might want to share core clock networks for the following reasons:

- To minimize the area and latency penalty associated with clock domain crossing.
- To minimize consumption of core clock networks.

Multiple memory interfaces can share the same core clock signals under the following conditions:

- The memory interfaces have the same protocol, rate, frequency, and PLL reference clock source.
- The interfaces reside in the same I/O column.
- The interfaces reside in adjacent bank locations.

For multiple memory interfaces to share core clocks, you must specify one of the interfaces as master and the remaining interfaces as slaves. Use the **Core clocks sharing** setting in the parameter editor to specify the master and slaves.

In your RTL, connect the `clks_sharing_master_out` signal from the master interface to the `clks_sharing_slave_in` signal of all the slave interfaces. Both the master and slave interfaces expose their own output clock ports in the RTL (e.g. `emif_usr_clk`, `afi_clk`), but the signals are equivalent, so it does not matter whether a clock port from a master or a slave is used.

Core clock sharing necessitates PLL reference clock sharing; therefore, only the master interface exposes an input port for the PLL reference clock. All slave interfaces use the same PLL reference clock signal.

## Arria 10 EMIF IP Component

The external memory interface IP component for Arria 10 provides a complete solution for implementing DDR3, DDR4, and QDR-IV external memory interfaces. The EMIF IP also includes a protocol-specific calibration algorithm that automatically determines the optimal delay settings for a robust external memory interface.

The external memory interface IP comprises the following parts:

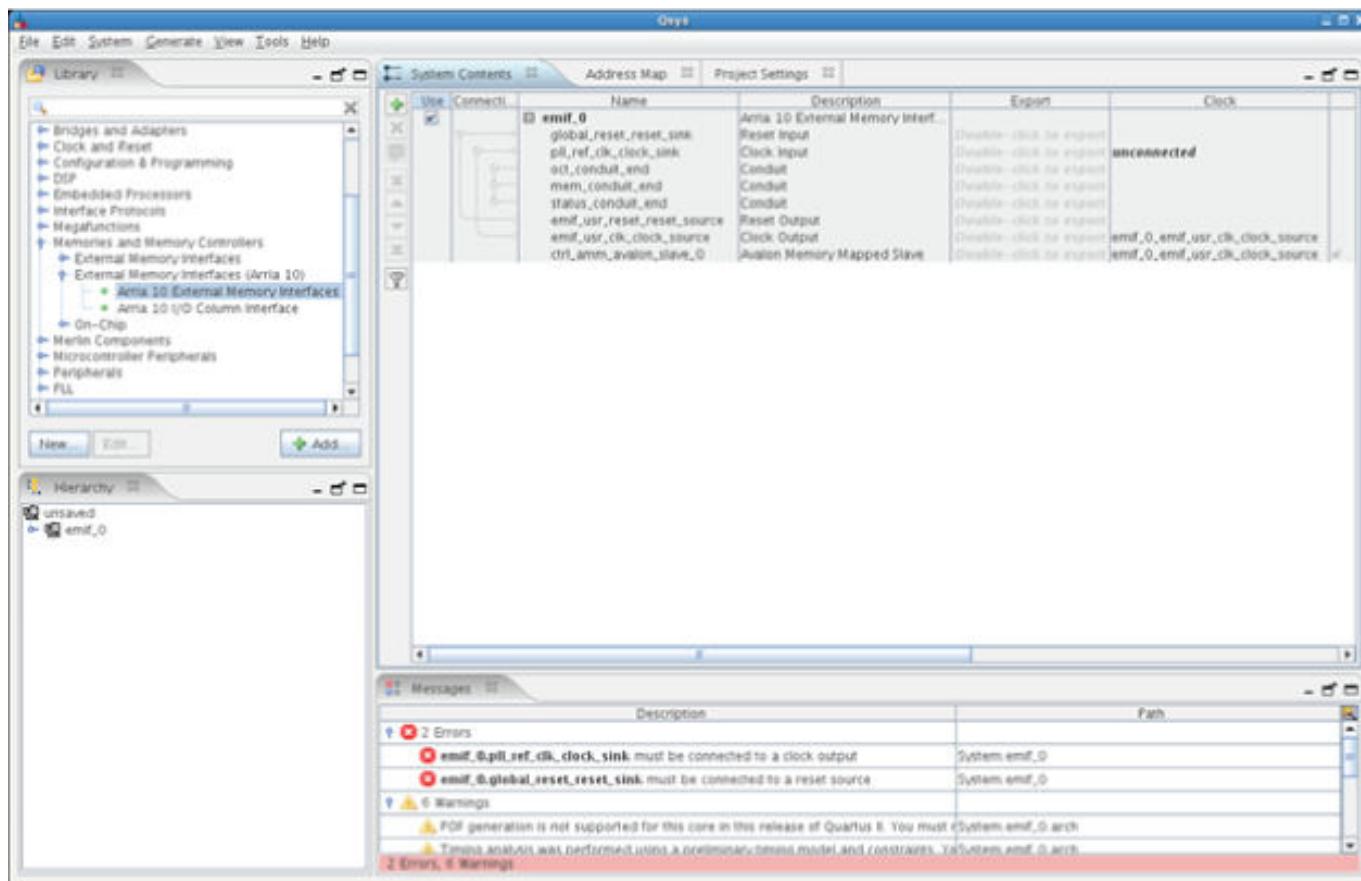
- A set of synthesizable files that you can integrate into a larger design
- A stand-alone synthesizable example design that you can use for hardware validation
- A set of simulation files that you can incorporate into a larger project
- A stand-alone simulation example project that you can use to observe controller and PHY operation
- A set of timing scripts that you can use to determine the maximum operating frequency of the memory interface based on external factors such as board skew, trace delays, and memory component timing parameters
- A customized data sheet specific to your memory interface configuration

## Instantiating Your Arria 10 EMIF IP in a Qsys Project

The following steps describe how to instantiate your Arria 10 EMIF IP in a Qsys project.

1. Within the Qsys interface, select **Memories and Memory Controllers** in the component Library tree.
2. Under **Memories and Memory Controllers**, select **External Memory Interfaces (Arria 10)**.
3. Under **External Memory Interfaces (Arria 10)**, select the **Arria 10 External Memory Interface** component.

Figure 2-14: Instantiating Arria 10 EMIF IP in Qsys



## Logical Connections

The following logical connections exist in an Arria 10 EMIF IP core.

Table 2-4: Logical Connections Table

Logical Connection	Description
afi_conduit_end (Conduit)	The Altera PHY Interface (AFI) connects a memory controller to the PHY. This interface is exposed only when you configure the memory interface in PHY-Only mode. The interface is synchronous to the <code>afi_clk</code> clock and <code>afi_reset_n</code> reset.

Logical Connection	Description
afi_clk_conduit_end (Conduit)	Use this clock signal to clock the soft controller logic. The afi_clk is an output clock coming from the PHY when the memory interface is in PHY-Only mode. The phase of afi_clk is adjusted dynamically by hard circuitry for the best data transfer between FPGA core logic and periphery logic with maximum timing margin. Multiple memory interface instances can share a single afi_clk using the <b>Core Clocks Sharing</b> option during IP generation.
afi_half_clk_conduit_end (Conduit)	This clock runs at half the frequency of afi_clk. It is exposed only when the memory interface is in PHY-Only mode.
afi_reset_n_conduit_end (Conduit)	This single-bit reset provides a synchronized reset output. Use this signal to reset all registers that are clocked by either afi_clk or afi_half_clk.
cal_debug_avalon_slave (Avalon Slave/Target)	This interface is exposed when the <b>EMIF Debug Toolkit/On-chip Debug Port</b> option is set to <b>Export</b> . This interface can be connected to an Arria 10 External Memory Interface Debug Component to allow EMIF Debug Toolkit access, or it can be used directly by user logic to access calibration diagnostic data.
cal_debug_clk_clock_sink (Clock Input)	This clock is used for the cal_debug_avalon_slave interface. It can be connected to the emif_usr_clk_clock_source interface.
cal_debug_reset_reset_sink (Reset Input)	This reset is used for the cal_debug_avalon_slave interface. It can be connected to the emif_usr_reset_reset_source interface.
cal_debug_out_avalon_master (Avalon Master/Source)	This interface is exposed when the <b>Enable Daisy-Chaining for EMIF Debug Toolkit/On-Chip Debug Port</b> option is enabled. Connect this interface to the cal_debug_avalon_slave interface of the next EMIF instance in the same I/O column.
cal_debug_out_clk_clock_source (Clock Output)	This interface should be connected to the cal_debug_clk_clock_sink interface of the next EMIF instance in the same I/O column (similar to cal_debug_out_avalon_master).
cal_debug_out_reset_reset_source (Reset Output)	This interface should be connected to the cal_debug_reset_reset_sink interface of the next EMIF instance in the same I/O column (similar to cal_debug_out_avalon_master).
effmon_csr_avalon_slave (Avalon Slave/Target)	This interface allows access to the Efficiency Monitor CSR. For more information, see the documentation on the UniPHY Efficiency Monitor.

Logical Connection	Description
global_reset_reset_sink (Reset Input)	This single-wire input port is the asynchronous reset input for the EMIF core.
pll_ref_clk_clock_sink (Clock Input)	This single-wire input port connects the external PLL reference clock to the EMIF core. Multiple EMIF cores may share a PLL reference clock source, provided the restrictions outlined in the PLL and PLL Reference Clock Network section are observed.
oct_conduit_end (Conduit)	This logical port is connected to an OCT pin and provides calibrated reference data for EMIF cores with pins that use signaling standards that require on-chip termination. Depending on the I/O standard, reference voltage, and memory protocol, multiple EMIF cores may share a single OCT pin.
mem_conduit_end (Conduit)	This logical conduit can attach an Altera Memory Model to an EMIF core for simulation. Memory models for various protocols are available under the <b>Memories and Memory Controllers — External Memory Interfaces — Memory Models</b> section of the component library in Qsys. You must ensure that all configuration parameters for the memory model match the configuration parameters of the EMIF core.
status_conduit_end (Conduit)	The status conduit exports two signals that can be sampled to determine if the calibration operation passed or failed for that core.
emif_usr_reset_reset_source (Reset Output)	This single-bit reset output provides a synchronized reset output that should be used to reset all components that are synchronously connected to the EMIF core. Assertion of the global reset input triggers an assertion of this output as well, therefore you should rely on this signal only as a reset source for all components connected to the EMIF core.
emif_usr_clk_clock_source (Clock Output)	Use this single-bit clock output to clock all logic connected to the EMIF core. The phase of this clock signal is adjusted dynamically by circuitry in the EMIF core such that data can be transferred between core logic and periphery registers with maximum timing margin. Drive all logic connected to the EMIF core with this clock signal. Other clock sources generated from the same reference clock or even the same PLL may have unknown phase relationships. Multiple EMIF cores can share a single core clock using the Core Clocks Sharing option described in the Example Design tab of the parameter editor.

Logical Connection	Description
ctrl_amm_avalon_slave (Avalon Slave/Target)	<p>This Avalon target port initiates read or write commands to the controller. Refer to the Avalon Interface Specification for more information on how to design cores that comply to the Avalon Bus Specification.</p> <p>For DDR3, DDR4, and LPDDR3 protocols with the hard PHY and hard controller configuration and an AVL slave interface exposed, <code>ctrl_amm_avalon_slave</code> is renamed to <code>ctrl_amm_avalon_slave_0</code>.</p> <p>For QDR II, QDR II+, and QDR II+ Xtreme interfaces with hard PHY and soft controller, separate read and write connections are used. <code>ctrl_amm_avalon_slave_0</code> is the read port and <code>ctrl_amm_avalon_slave_1</code> is the write port.</p> <p>For QDR-IV interfaces with hard PHY and soft controller operating at quarter rate, a total of eight separate Avalon interfaces (named <code>ctrl_amm_avalon_slave_0</code> to <code>ctrl_amm_avalon_slave_7</code>) are used to maximize bus efficiency.</p>

## File Sets

The Arria 10 EMIF IP core generates four output file sets for every EMIF IP core, arranged according to the following directory structure.

**Table 2-5: Generated File Sets**

Directory	Description
<code>&lt;core_name&gt;/*</code>	<p>This directory contains only the files required to integrate a generated EMIF core into a larger design. This directory contains:</p> <ul style="list-style-type: none"> <li>• Synthesizable HDL source files</li> <li>• Customized TCL timing scripts specific to the core (protocol and topology)</li> <li>• HEX files used by the calibration algorithm to identify the interface parameters</li> <li>• A customized data sheet that describes the operation of the generated core</li> </ul> <p><b>Note:</b> The top-level HDL file is generated in the root folder as <code>&lt;core_name&gt;.v</code> (or <code>&lt;core_name.vhd</code> for VHDL designs). You can reopen this file in the parameter editor if you want to modify the EMIF core parameters and regenerate the design.</p>

Directory	Description
<code>&lt;core_name&gt;_sim/*</code>	This directory contains the simulation fileset for the generated EMIF core. These files can be integrated into a larger simulation project. For convenience, simulation scripts for compiling the core are provided in the <b>/mentor</b> , <b>/cadence</b> , <b>/synopsys</b> , and <b>/riviera</b> subdirectories.  The top-level HDL file, <code>&lt;core_name&gt;.v</code> (or <code>&lt;core_name&gt;.vhdl</code> ) is located in this folder, and all remaining HDL files are placed in the <b>/altera_emif_arch_nf</b> subfolder, with the customized data sheet. The contents of this directory are not intended for synthesis.
<code>emif_&lt;instance_num&gt;_example_design/*</code>	This directory contains a set of TCL scripts, QSYS project files and README files for the complete synthesis and simulation example designs. You can invoke these scripts to generate a standalone fully-synthesizable project complete with an example driver, or a standalone simulation design complete with an example driver and a memory model.

## Customized readme.txt File

When you generate your Arria 10 EMIF IP, the system produces a customized readme file, containing data indicative of the settings in your IP core.

The readme file is `<variation_name>/altera_emif_arch_nf_<version_number>/<synth|sim>/<variation_name>_altera_emif_arch_nf_<version_number>_<unique ID>_readme.txt`, and contains a summary of Arria 10 EMIF information, and details specific to your IP core, including:

- Pin location guidelines
- External port names, directions, and widths
- Internal port names, directions, and widths
- Avalon interface configuration details (if applicable)
- Calibration mode
- A brief summary of all configuration settings for the generated IP

You should review the generated readme file for implementation guidelines specific to your IP core.

## Clock Domains

The Arria 10 EMIF IP core provides a single clock domain to drive all logic connected to the EMIF core.

The frequency of the clock depends on the core-clock to memory-clock interface rate ratio. For example, a quarter-rate interface with an 800 MHz memory clock would provide a 200 MHz clock to the core ( $800 \text{ MHz} / 4 = 200 \text{ MHz}$ ). The EMIF IP dynamically adjusts the phase of the core clock with respect to the periphery clock to maintain the optimum alignment for transferring data between the core and periphery. Independent EMIF IP cores driven from the same reference clock have independent core clock domains. You should employ one of the following strategies if you are implementing multiple EMIF cores:

1. Treat all crossing between independent EMIF-generated clock domains as asynchronous, even though they are generated from the same reference clock.
2. Use the **Core clock sharing** option to enforce that multiple EMIF cores share the same core clock. You must enable this option during IP generation. This option is applicable only for cores that reside in the same I/O column.

## ECC in Arria 10 EMIF IP

The ECC (error correction code) is a soft component of the Arria 10 EMIF IP that reduces the chance of errors when reading and writing to external memory. ECC allows correction of single-bit errors and reduces the chances of system failure.

The ECC component includes an encoder, decoder, write FIFO buffer, and modification logic, to allow read-modify-write operations. The ECC code employs standard Hamming logic to correct single-bit errors and to detect double-bit errors. ECC is available in 16, 24, 40, and 72 bit widths.

When writing data to memory, the encoder creates ECC bits and writes them together with the regular data. When reading from memory, the decoder checks the ECC bits and regular data, and passes the regular data unchanged if no errors are detected. If a single-bit error is detected, the ECC logic corrects the error and passes the regular data. If more than a single-bit error is detected, the ECC logic sets a flag to indicate the error.

Read-modify-write operations can occur in the following circumstances:

- A partial write in data mask mode (with or without ECC), where at least one memory burst of byte-enable is not all ones or all zeros.
- Auto-correction with ECC enabled. This is usually a dummy write issued by the auto-correction logic to correct the memory content when a single-bit error is detected. The read-modify-write reads back the data, corrects the single-bit error, and writes the data back.

## ECC Components

The ECC logic communicates with user logic via an Avalon-MM interface, and with the hard memory controller via an Avalon-ST interface

### ECC Encoder

The ECC encoder consists of a x64/x72 encoder IP core capable of single-bit error correction and double-bit error detection. The encoder takes 64 bits input and converts it to 72 bits output, where the 8 additional bits are ECC code. The encoder supports any input data width less than 64-bits. Any unused input data bits are set to zero.

### ECC Decoder

The ECC decoder consists of a x72/x64 decoder IP core capable of double-bit error detection. The decoder takes 72 bits input and converts it to 64 bits output. The decoder also produces single-bit error and double-bit error information. The decoder controls the user read data valid signal; when read data is intended for partial write, the user read data valid signal is deasserted, because the read data is meant for merging, not for the user.

### Partial Write Data FIFO Buffer

The Partial Write Data FIFO Buffer is implemented in soft logic to store partial write data and byte enable. Data and byte enable are popped and merged with the returned read data. A partial write can occur in the following situations:

- At least one memory burst of byte enable is not all ones or all zeroes.
- Non data masked mode, where all memory bursts of byte enable are not all ones.
- A dummy write with auto-correction logic, where all memory bursts of byte enable are all zeroes. (You might use a dummy write when correcting memory content with a single-bit error.)

## Merging Logic

Merge return partial read data with write data based on byte enabled popped from the FIFO buffer, and send it to the ECC encoder.

## Pointer FIFO Buffer

The pointer FIFO buffer is implemented in soft logic to store write data pointers. The ECC logic refers to the pointers when sending write data to DBC. The pointers serve to overwrite existing write data in the data buffer during a read-modify-write process.

## Partial Logic

Partial logic decodes byte enable information and distinguishes between normal and partial writes.

## Memory Mode Register Interface

The Memory Mode Register interface is an Avalon-based interface through which core logic can access debug signals and sideband operation requests in the hard memory controller.

The MMR logic routes ECC-related operations to an MMR register implemented in soft logic, and returns the ECC information via an Avalon-MM interface. The MMR logic tracks single-bit and double-bit error status, and provides the following information:

- Interrupt status.
- Single-bit error and double-bit error status.
- Single-bit error and double-bit error counts (to a maximum of 15; if more than 15 errors occur, the count will overflow).
- Address of the last error.

## ECC User Interface Controls

You can enable the ECC logic from the **Configuration, Status, and Error Handling** section of the **Controller** tab in the parameter editor.

There are three user interface settings related to ECC:

- **Enabled Memory-Mapped Configuration and Status Register (MMR) Interface:** Allows run-time configuration of the memory controller. You can enable this option together with ECC to retrieve error detection information from the ECC logic.
- **Enabled Error Detection and Correction Logic:** Enables ECC logic for single-bit error correction and double-bit error detection.
- **Enable Auto Error Correction:** Allows the controller to automatically correct single-bit errors detected by the ECC logic.

## Examples of External Memory Interface Implementations for DDR4

The following figures are examples of external memory interface implementations for different DDR4 memory widths. The figures show the locations of the address/command and data pins in relation to the locations of the memory controllers.

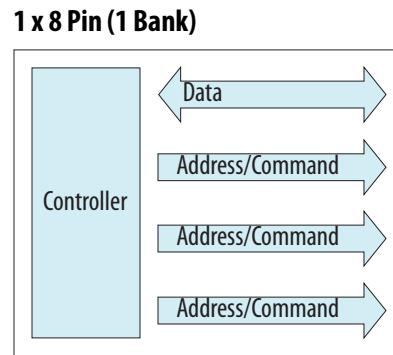
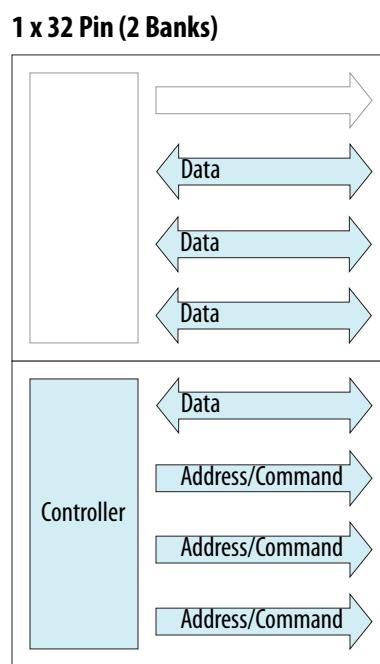
**Figure 2-15: DDR4 1x8 Implementation Example (One I/O Bank)****Figure 2-16: DDR4 1x32 Implementation Example (Two I/O Banks)**

Figure 2-17: DDR4 1x72 Implementation Example (Three I/O Banks)

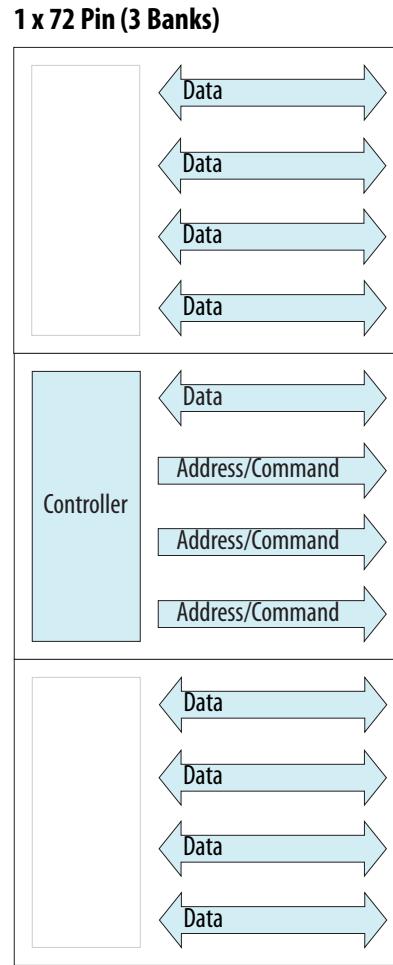


Figure 2-18: DDR4 2x16 Implementation Example with Controllers in Non-Adjacent Banks (Three I/O Banks)

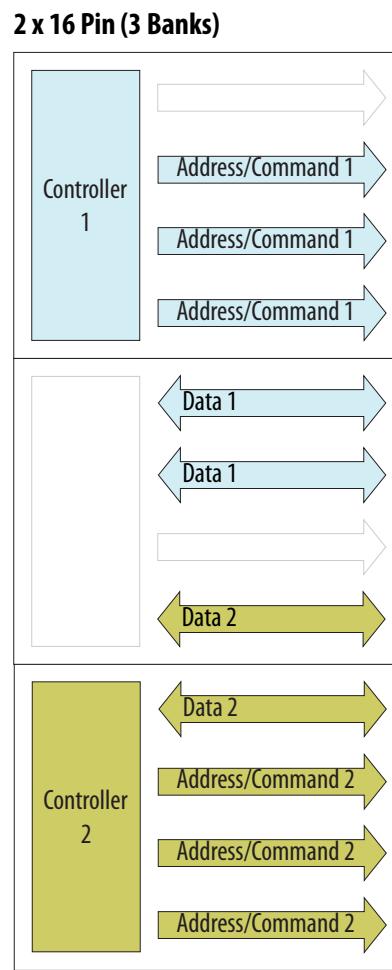
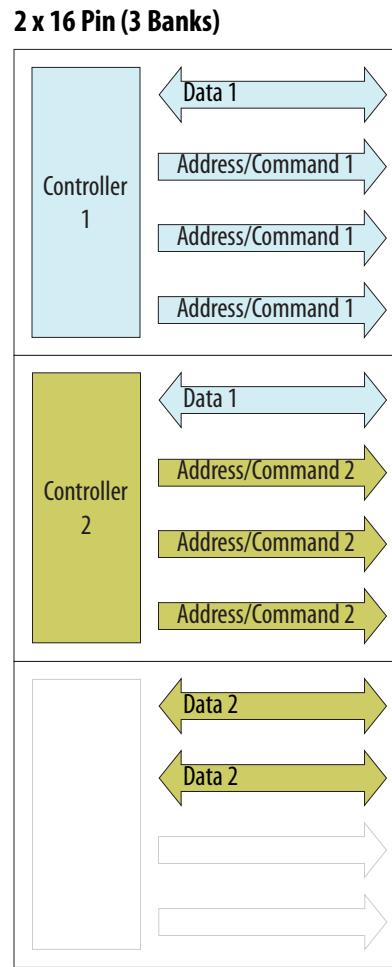
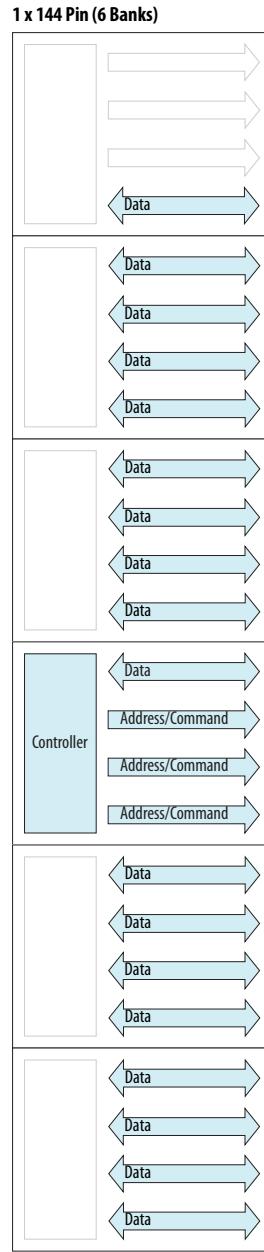


Figure 2-19: DDR4 2x16 Implementation Example with Controllers in Adjacent Banks (Three I/O Banks)



**Figure 2-20: DDR4 1x144 Implementation Example (Six I/O Banks)**

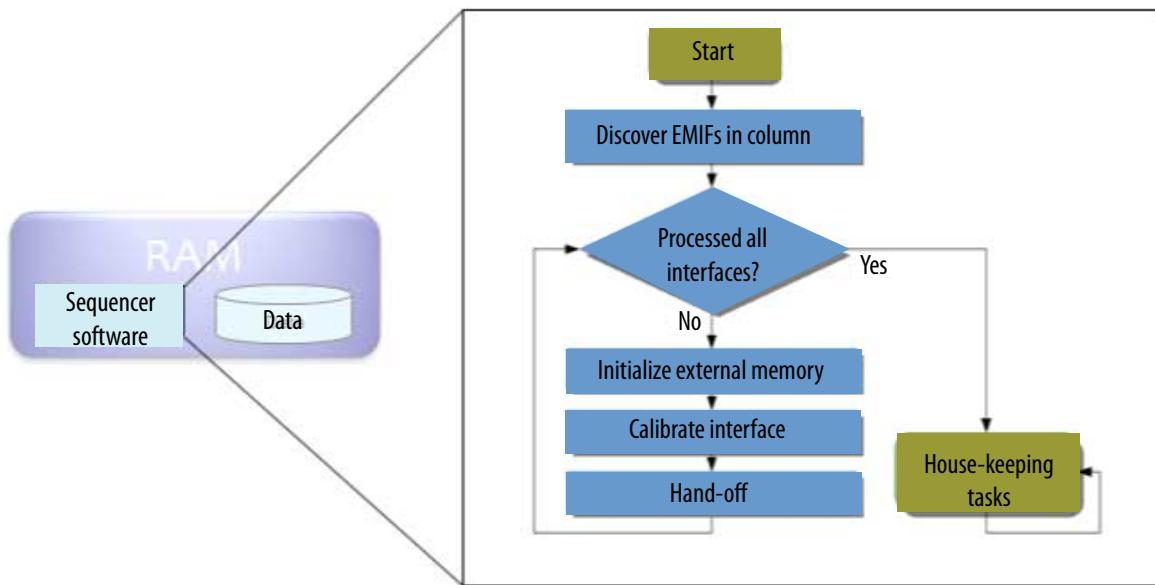
## Arria 10 EMIF Sequencer

The Arria 10 EMIF sequencer is fully hardened in silicon, with executable code to handle protocols and topologies. Hardened RAM contains the calibration algorithm.

The Arria 10 EMIF sequencer is responsible for the following operations:

- Initializes memory devices.
- Calibrates the external memory interface.
- Governs the hand-off of control to the memory controller.
- Handles recalibration requests and debug requests.
- Handles all supported protocols and configurations.

**Figure 2-21: Arria 10 EMIF Sequencer Operation**



## Arria 10 EMIF DQS Tracking

DQS tracking is enabled for QDR II / II+ / QDR II+ Xtreme, RLDRAM 3, and LPDDR3 protocols. DQS tracking is not available for DDR3 and DDR4 protocols.

## Arria 10 EMIF Calibration

The calibration process compensates for skews and delays in the external memory interface.

The calibration process enables the system to compensate for the effects of factors such as the following:

- Timing and electrical constraints, such as setup/hold time and Vref variations.
- Circuit board and package factors, such as skew, fly-by effects, and manufacturing variations.
- Environmental uncertainties, such as variations in voltage and temperature.
- The demanding effects of small margins associated with high-speed operation.

## Calibration Stages

At a high level, the calibration routine consists of address and command calibration, read calibration, and write calibration.

The stages of calibration vary, depending on the protocol of the external memory interface.

**Table 2-6: Calibration Stages by Protocol**

Stage	DDR4	DDR3	LPDDR3	RLDRAM II/3	QDR-IV	QDR II/II+
<b>Address and command</b>						
Leveling	Yes	Yes	—	—	—	—
Deskew	Yes	—	Yes	—	Yes	—
<b>Read</b>						
DQSen	Yes	Yes	Yes	Yes	Yes	Yes
Deskew	Yes	Yes	Yes	Yes	Yes	Yes
VREF-In	Yes	—	—	—	Yes	—
LFIFO	Yes	Yes	Yes	Yes	Yes	Yes
<b>Write</b>						
Leveling	Yes	Yes	Yes	Yes	Yes	—
Deskew	Yes	Yes	Yes	Yes	Yes	Yes
VREF-Out	Yes	—	—	—	—	—

## Calibration Stages Descriptions

The various stages of calibration perform address and command calibration, read calibration, and write calibration.

### Address and Command Calibration

The goal of address and command calibration is to delay address and command signals as necessary to optimize the address and command window. This stage is not available for all protocols, and cannot compensate for an inefficient board design.

Address and command calibration consists of the following parts:

- Leveling calibration— Centers the CS# signal and the entire address and command bus, relative to the CK clock. This operation is available only for DDR3 and DDR4 interfaces.
- Deskew calibration— Provides per-bit deskew for the address and command bus (except CS#), relative to the CK clock. This operation is available for DDR4 and QDR-IV interfaces only.

### Read Calibration

Read calibration consists of the following parts:

- DQSen calibration— Calibrates the timing of the read capture clock gating and ungating, so that the PHY can gate and ungate the read clock at precisely the correct time—if too early or too late, data corruption can occur. The algorithm for this stage varies, depending on the memory protocol.
- Deskew calibration— Performs per-bit deskew of read data relative to the read strobe or clock.
- VREF-IN calibration— Calibrates the Vref level at the FPGA.
- LFIFO calibration: Normalizes differences in read delays between groups due to fly-by, skews, and other variables and uncertainties.

## Write Calibration

Write calibration consists of the following parts:

- Leveling calibration— Aligns the write strobe and clock to the memory clock, to compensate for skews, especially those associated with fly-by topology. The algorithm for this stage varies, depending on the memory protocol.
- Deskew calibration— Performs per-bit deskew of write data relative to the write strobe and clock.
- VREF-Out calibration— Calibrates the VREF level at the memory device.

## Calibration Algorithms

The calibration algorithms sometimes vary, depending on the targeted memory protocol.

### Address and Command Calibration

Address and command calibration consists of the following parts:

- Leveling calibration— (DDR3 and DDR4 only) Toggles the CS# and CAS# signals to send read commands while keeping other address and command signals constant. The algorithm monitors for incoming DQS signals, and if the DQS signal toggles, it indicates that the read commands have been accepted. The algorithm then repeats using different delay values, to find the optimal window.
- Deskew calibration— (DDR4, QDR-IV, and LPDDR3 only)
  - (DDR4) Uses the DDR4 address and command parity feature. The FPGA sends the address and command parity bit, and the DDR4 memory device responds with an alert signal if the parity bit is detected. The alert signal from the memory device tells the FPGA that the parity bit was received. Deskew calibration requires use of the PAR/ALERT# pins, so you should not omit these pins from your design. One limitation of deskew calibration is that it cannot deskew ODT and CKE pins.
  - (QDR-IV) Uses the QDR-IV loopback mode. The FPGA sends address and command signals, and the memory device sends back the address and command signals which it captures, via the read data pins. The returned signals indicate to the FPGA what the memory device has captured. Deskew calibration can deskew all synchronous address and command signals.
  - (LPDDR3) Uses the LPDDR3 CA training mode. The FPGA sends signals onto the LPDDR3 CA bus, and the memory device sends back those signals that it captures, via the DQ pins. The returned signals indicate to the FPGA what the memory device has captured. Deskew calibration can deskew all signals on the CA bus. The remaining command signals (CS, CKE, and ODT) are calibrated based on the average of the deskewed CA bus.

### Read Calibration

- DQSen calibration— (DDR3, DDR4, LPDDR3, RLDRAMx and QDRx) DQSen calibration occurs before Read deskew, therefore only a single DQ bit is required to pass in order to achieve a successful read pass.
  - (DDR3, DDR4, and LPDDR3) The DQSen calibration algorithm searches the DQS preamble using a hardware state machine. The algorithm sends many back-to-back reads with a one clock cycle gap between. The hardware state machine searches for the DQS gap while sweeping DQSen delay values. The algorithm then increments the VFIFO value, and repeats the process until a pattern is found. The process is then repeated for all other read DQS groups.
  - (RLDRAMx and QDRx) The DQSen calibration algorithm does not use a hardware state machine; rather, it calibrates cycle-level delays using software and subcycle delays using DQS tracking hardware. The algorithm requires good data in memory, and therefore relies on guaranteed writes. (Writing a burst of 0s to one location, and a burst of 1s to another; back-to-back reads from these two locations are used for read calibration.)

The algorithm enables DQS tracking to calibrate the phase component of DQS enable. It then issues a guaranteed write, followed by back-to-back reads. The algorithm sweeps DQSen values cycle by cycle until the read operation succeeds. The process is then repeated for all other read groups.

- Deskew calibration— Read deskew calibration is performed before write leveling, and must be performed at least twice: once before write calibration, using simple data patterns from guaranteed writes, and again after write calibration, using complex data patterns.

The deskew calibration algorithm performs a guaranteed write, and then sweeps `dqs_in` delay values from low to high, to find the right edge of the read window. The algorithm then sweeps `dq_in` delay values low to high, to find the left edge of the read window. Updated `dqs_in` and `dq_in` delay values are then applied to center the read window. The algorithm then repeats the process for all data pins.

- Vref-In calibration— Read `vref_in` calibration begins by programming `vref_in` with an arbitrary value. The algorithm then sweeps the `vref_in` value from the starting value to both ends, and measures the read window for each value. The algorithm selects the `vref_in` value which provides the maximum read window.
- LFIFO calibration— Read LFIFO calibration normalizes read delays between groups. The PHY must present all data to the controller as a single data bus. The LFIFO latency should be large enough for the slowest read data group, and large enough to allow proper synchronization across FIFOs.

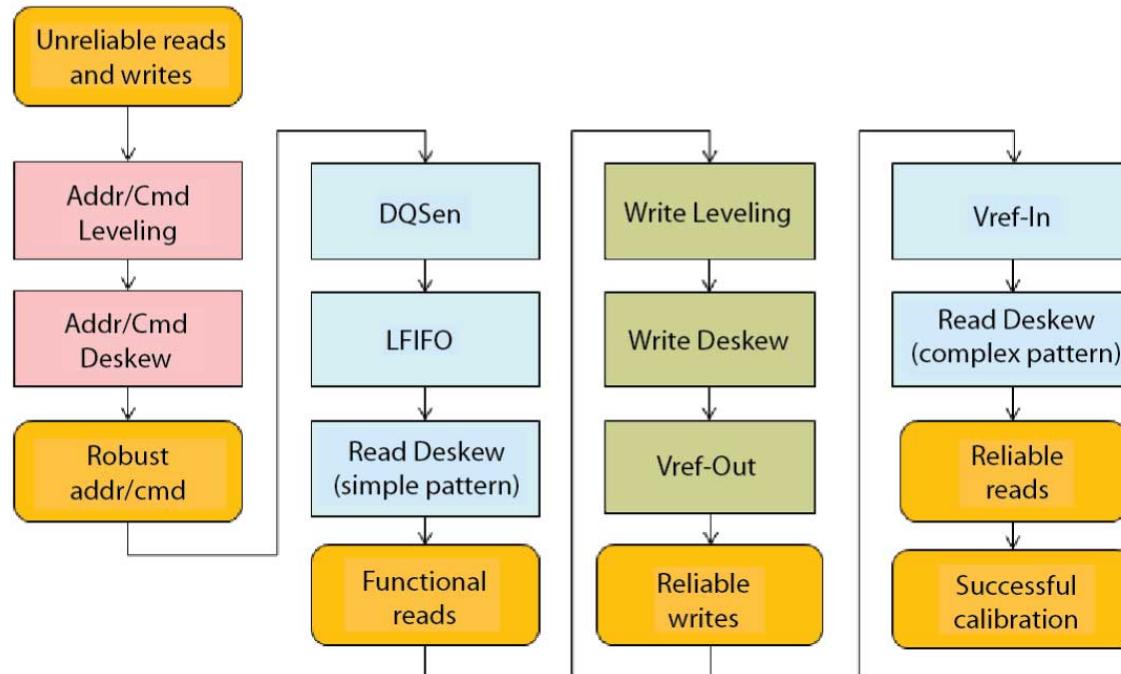
## Write Calibration

- Leveling calibration— Write leveling calibration aligns the write strobe and clock to the memory clock, to compensate for skews. In general, leveling calibration tries a variety of delay values to determine the edges of the write window, and then selects an appropriate value to center the window. The details of the algorithm vary, depending on the memory protocol.
  - (DDR $x$ , LPDDR $3$ ) Write leveling occurs before write deskew, therefore only one successful DQ bit is required to register a pass. Write leveling staggers the DQ bus to ensure that at least one DQ bit falls within the valid write window.
  - (RLDRAM $x$ ) Optimizes for the CK versus DK relationship.
  - (QDR-IV) Optimizes for the CK versus DK relationship. Is covered by address and command deskew using the loopback mode.
  - (QDR II/II+/Xtreme) The K clock is the only clock, therefore write leveling is not required.
- Deskew calibration— Performs per-bit deskew of write data relative to the write strobe and clock. Write deskew calibration does not change `dqs_out` delays; the write clock is aligned to the CK clock during write leveling.
- VREF-Out calibration— (DDR4) Calibrates the VREF level at the memory device. The VREF-Out calibration algorithm is similar to the VREF-In calibration algorithm.

## Calibration Flowchart

The following flowchart illustrates the calibration flow.

**Figure 2-22: Calibration Flowchart**



## Periodic OCT Recalibration

Periodic OCT recalibration improves the accuracy of the on-chip termination values used by DDR4 Pseudo-open Drain (POD) I/Os. This feature periodically invokes the user-mode OCT calibration engine

and updates the I/O buffer termination settings to compensate for variations in calibrated OCT settings caused by large changes in device operating temperature.

This feature is automatically enabled for DDR4 memory interfaces unless the IP does not meet the technical requirements, or if you explicitly disable the feature in the parameter editor.

## Operation

The Periodic OCT recalibration engine refreshes the calibrated OCT settings for DDR4 I/O buffers every 500ms. To ensure data integrity, there is a momentary pause in user traffic as the OCT settings are refreshed; however, the process of OCT calibration is decoupled from the actual update to the I/O buffers, to minimize disruption of user traffic.

The calibration process uses the external RZQ reference resistor to determine the optimal settings for the I/O buffer, to meet the specified calibrated I/O standards on the FPGA. OCT Calibration only affects the I/O pin that is connected to the RZQ resistor; therefore, memory traffic is not interrupted during the calibration phase.

Upon completion of the calibration process, the updated calibration settings are applied to the I/O buffers. The memory traffic is halted momentarily by placing the memory into self-refresh mode; this ensures that the data bus is idle and no glitches are created by the I/O buffers during the buffer update. The buffer is updated as soon as the memory enters self-refresh mode. The memory interface exits self-refresh mode when the buffer update is complete and new read or write requests are detected on the Avalon bus. The controller remains in self-refresh mode until a new command is detected. OCT calibration continues to occur even if the memory is still in self refresh mode. Upon detection of a new command, the controller issues a self-refresh exit command to the memory, followed by a memory-side ZQ calibration short duration (ZQCS) command. Memory traffic resumes when the memory DLL has relocked.

If you disable the periodic OCT recalibration engine, the calibration process occurs only once during device configuration. In this operating mode, the calibrated OCT settings can vary across temperature as specified by the calibration accuracy ranges listed in the Arria 10 Device Handbook. The DDR external timing report automatically factors in the effect of enabling or disabling the periodic OCT recalibration engine when calculating the total amount of external I/O transfer margin.

## Technical Restrictions

Certain criteria must be met in order to use periodic OCT recalibration.

The periodic OCT recalibration engine is enabled only when all of the following criteria are met:

- The memory interface is configured to use the Altera Hard Memory Controller for DDR4.
- The memory interface is configured for either DDR4 UDIMM or component topologies. RDIMM and LRDIMM topologies are not supported.
- The memory interface is not used with the hardened processor subsystem.
- The memory interface does not use Ping-Pong PHY.
- The memory interface does not use calibrated I/O standards for address, command, or clock signals.
- The memory interface uses calibrated I/O standards for the data bus.
- The memory does not use the memory mapped register (MMR) interface of the HMC, including ECC modes.
- You have not explicitly disabled periodic OCT recalibration in the parameter editor.
- The specified device is a production level device (that is, not an ES/ES2/ES3 class device).

Periodic OCT recalibration requires that each EMIF instance in the design employ a dedicated RZQ resistor. Because this restriction cannot be detected at IP generation time, you must explicitly disable the

periodic OCT recalibration engine for a given interface if it shares an RZQ resistor with another interface. You should also be aware of this restriction when automatically upgrading EMIF IP from older versions of the Quartus Prime software.

## Efficiency Impact

The Periodic OCT recalibration engine must interrupt user traffic for a short period of time in order to update I/O buffer termination settings.

The exact flow of operations executed by the recalibration engine that affects memory traffic is described below:

1. Enter Self-Refresh Mode. The EMIF calibration CPU triggers self-refresh entry on the hard memory controller. The controller flushes all pending operations, precharges all banks and issues the self-refresh command. This operation introduces a delay of approximately 25 Memory clock cycles (precharge all and self-refresh entry commands).
2. Confirm Self-Refresh Mode. The EMIF calibration CPU polls the hard memory controller to confirm that the clocks have stopped. This operation introduces no delay.
3. Issue codeword update. The EMIF calibration CPU triggers user-mode OCT logic to update code words. This operation introduces a delay of 50-100ns, depending on the device speed grade.
4. Allow Exit Self-Refresh Mode. The EMIF calibration CPU enables automatic self-refresh exit logic. This operation introduces a delay of 50-100ns, depending on the device speed grade.
5. Wait for Memory Traffic. The hard memory controller waits for an incoming read or write command on the Avalon bus. The delay introduced by this operation varies, depending on the user application.
6. Exit Self Refresh Mode. The hard memory controller issues the Self-Refresh Exit command and a simultaneous memory-side RZQ calibration (ZQCS) command. The delay introduced by this operation varies according to the device speed bin (up to ~1000 memory clock cycles for fastest memory devices).

The efficiency impact on throughput-sensitive work loads is less than one percent, even under worst-case scenarios with all banks active. However, be aware that the first command issued after the hard memory controller exits self-refresh mode will incur the latency overhead of waiting for the memory DLL to re-lock when the Self-Refresh Exit command is issued by the hard memory controller. Contact Altera Technical Services for information on how to manually trigger or inhibit periodic OCT updates for applications that are sensitive to latency.

## Arria 10 EMIF and SmartVID

Arria 10 EMIF IP can be used with the SmartVID voltage management system, to achieve reduced power consumption.

The SmartVID controller allows the FPGA to operate at a reduced  $V_{cc}$ , while maintaining performance. Because the SmartVID controller can adjust  $V_{cc}$  up or down in response to power requirements and temperature, it can have an impact on external memory interface performance. When used with the SmartVID controller, the EMIF IP implements a handshake protocol to ensure that EMIF calibration does not begin until after voltage adjustment has completed.

In extended speed grade devices, voltage adjustment occurs once when the FPGA is powered up, and no further voltage adjustments occur. The external memory calibration occurs after this initial voltage adjustment is completed. EMIF specifications are expected to be slightly lower in extended speed grade devices using SmartVID, than in devices not using SmartVID.

In industrial speed grade devices, voltage adjustment occurs at power up, and may also occur during operation, in response to temperature changes. External memory interface calibration does not occur until after the initial voltage adjustment at power up. However, the external memory interface is not recalibrated in response to subsequent voltage adjustments that occur during operation. As a result, EMIF specifications for industrial speed grade devices using SmartVID are expected to be lower than for extended speed grade devices.

### Using Arria 10 EMIF IP with SmartVID

To employ Arria 10 EMIF IP with SmartVID, follow these steps:

1. Ensure that the Quartus Prime project and Qsys system are configured to use VID components. This step exposes the `vid_cal_done_persist` interface on instantiated EMIF IP, which is required for communicating with the SmartVID controller.

2. Instantiate the SmartVID controller, using an I/O PLL IP core to drive the 125MHz `vid_clk` and the 25MHz `jtag_core_clk` inputs of the Smart VID controller.

**Note:** Do not connect the `emif_usr_clk` signal to either the `vid_clk` or `jtag_core_clk` inputs. Doing so would hold both the EMIF IP and the SmartVID controller in a perpetual reset condition.

3. Instantiate the Arria 10 EMIF IP.

4. Connect the `vid_cal_done_persist` signal from the EMIF IP with the `cal_done_persistent` signal on the SmartVID controller. This connection enables handshaking between the EMIF IP and the SmartVID controller, which allows the EMIF IP to delay memory calibration until after voltage levels are stabilized.

**Note:** The EMIF `vid_cal_done_persist` interface becomes available only when a VID-enabled device is selected.

#### Related Information

[SmartVID Controller IP Core User Guide](#)

## Hard Memory Controller Rate Conversion Feature

The hard memory controller's rate conversion feature allows the hard memory controller and PHY to run at half-rate, even though user logic is configured to run at quarter-rate.

To facilitate timing closure, you may choose to clock your core user logic at quarter-rate, resulting in easier timing closure at the expense of increased area and latency. To improve efficiency and help reduce overall latency, you can run the hard memory controller and PHY at half rate.

The rate conversion feature converts traffic from the FPGA core to the hard memory controller from quarter-rate to half-rate, and traffic from the hard memory controller to the FPGA core from half-rate to quarter-rate. From the perspective of user logic inside the FPGA core, the effect is the same as if the hard memory controller were running at quarter-rate.

The rate conversion feature is enabled automatically during IP generation whenever all of the following conditions are met:

- The hard memory controller is in use.
- User logic runs at quarter-rate.
- The interface targets either an ES2 or production device.
- Running the hard memory controller at half-rate does not exceed the fMax specification of the hard memory controller and hard PHY.

When the rate conversion feature is enabled, you should see the following info message displayed in the IP generation GUI:

PHY and controller running at 2x the frequency of user logic for improved efficiency.

#### Related Information

[Arria 10 Core Fabric and General Purpose I/Os Handbook](#)

## Compiling Arria 10 EMIF IP with the Quartus Prime Software

### Instantiating the Arria 10 EMIF IP

Depending on your work flow, you may instantiate your IP with Qsys or with the IP Catalog.

#### Instantiating with Qsys

If you instantiate your IP as part of a system in Qsys, follow the Qsys documentation for information on instantiating the IP in a Quartus Prime project.

#### Instantiating with the IP Catalog

If you generated your IP with the IP Catalog, you must add the Quartus Prime IP file (**.qip**) to your Quartus Prime project. The **.qip** file identifies the names and locations of the files that compose the IP. After you add the **.qip** file to your project, you can instantiate the memory interface in the RTL.

### Setting I/O Assignments in Arria 10 EMIF IP

The **.qip** file contains the I/O standard and I/O termination assignments required by the memory interface pins for proper operation. The assignment values are based on input that you provide during generation.

Unlike earlier device families, for Arria 10 EMIF IP you do not need to run a `<instance_name>.pin_assignments.tcl` script to add the assignments into the Quartus Prime Settings File (**.qsf**). The system reads and applies the assignments from the **.qip** file during every compilation, regardless of how you name the memory interface pins in the top-level design component. No new assignments are created in the project's **.qsf** file during compilation.

Note that I/O assignments in the **.qsf** file must specify the names of your top-level pins as target (-**to**), and you must not include the -entity or -library options.

Consult the generated **.qip** file for the set of I/O assignments that are provided with the IP.

#### Changing I/O Assignments

You should not make changes to the generated **.qip** file, because any changes are overwritten and lost when you regenerate the IP. If you want to override an assignment made in the **.qip** file, add the desired assignment to the project's **.qsf** file. Assignments in the **.qsf** file always take precedence over assignments in the **.qip** file.

## Debugging Arria 10 EMIF IP

You can debug hardware failures by connecting to the EMIF Debug Toolkit or by exporting an Avalon-MM slave port, from which you can access information gathered during calibration. You can also connect to this port to mask ranks and to request recalibration.

You can access the exported Avalon-MM port in two ways:

- Via the External Memory Interface Debug Toolkit
- Via On-Chip Debug (core logic on the FPGA)

### External Memory Interface Debug Toolkit

The External Memory Interface Debug Toolkit provides access to data collected by the Nios II sequencer during memory calibration, and allows you to perform certain tasks.

The External Memory Interface Debug Toolkit provides access to data including the following:

- General interface information, such as protocol and interface width
- Calibration results per group, including pass/fail status, failure stage, and delay settings

You can also perform the following tasks:

- Mask ranks from calibration (you might do this to skip specific ranks)
- Request recalibration of the interface

### On-Chip Debug for Arria 10

The On-Chip Debug feature allows user logic to access the same debug capabilities as the External Memory Interface Toolkit. You can use On-Chip Debug to monitor the calibration results of an external memory interface, without a connected computer.

To use On-Chip Debug, you need a C header file which is provided as part of the external memory interface IP. The C header file defines data structures that contain calibration data, and definitions of the commands that can be sent to the memory interface.

The On-Chip Debug feature accesses the data structures through the Avalon-MM port that is exposed by the EMIF IP when you turn on debugging features.

### Configuring Your EMIF IP for Use with the Debug Toolkit

The Arria 10 EMIF Debug Interface IP core contains the access point through which the EMIF Debug Toolkit reads calibration data collected by the Nios II sequencer.

#### Connecting an EMIF IP Core to an Arria 10 EMIF Debug Interface

For the EMIF Debug Toolkit to access the calibration data for an Arria 10 EMIF IP core, you must connect one of the EMIF cores in each I/O column to an Arria 10 EMIF Debug Interface IP core. Subsequent EMIF IP cores in the same column must connect in a daisy chain to the first.

There are two ways that you can add the Arria 10 EMIF Debug Interface IP core to your design:

- When you generate your EMIF IP core, on the **Diagnostics** tab, select **Add EMIF Debug Interface** for the **EMIF Debug Toolkit/On-Chip Debug Port**; you do not have to separately instantiate an Arria 10 EMIF Debug Interface core. This method does not export an Avalon-MM slave port. You can use this method if you require only EMIF Debug Toolkit access to this I/O column; that is, if you do not require On-Chip Debug Port access, or PHYLITE reconfiguration access.
- When you generate your EMIF IP core, on the **Diagnostics** tab, select **Export** for the **EMIF Debug Toolkit/On-Chip Debug Port**. Then, separately instantiate an Arria 10 EMIF Debug Interface core and connect its `to_ioaux` interface to the `cal_debug` interface on the EMIF IP core. This method is appropriate if you want to also have On-Chip Debug Port access to this I/O column, or PHYLITE reconfiguration access.

For each of the above methods, you must assign a unique interface ID for each external memory interface in the I/O column, to identify that interface in the Debug Toolkit. You can assign an interface ID using the dropdown list that appears when you enable the **Debug Toolkit/On-Chip Debug Port** option.

### Daisy-Chaining Additional EMIF IP Cores for Debugging

After you have connected an Arria 10 EMIF Debug Interface to one of the EMIF IP cores in an I/O column, you must then connect subsequent EMIF IP cores in that column in a daisy-chain manner. If you don't require debug capabilities for a particular EMIF IP core, you do not have to connect that core to the daisy chain.

To create a daisy chain of EMIF IP cores, follow these steps:

1. On the first EMIF IP core, select **Enable Daisy-Chaining for EMIF Debug Toolkit/On-Chip Debug Port** to create an Avalon-MM interface called `cal_debug_out`.
2. On the second EMIF IP core, select **Export** as the **EMIF Debug Toolkit/On-Chip Debug Port** mode, to export an Avalon-MM interface called `cal_debug`.
3. Connect the `cal_debug_out` interface of the first EMIF core to the `cal_debug` interface of the second EMIF core.
4. To connect more EMIF cores to the daisy chain, select the **Enable Daisy-Chaining for EMIF Debug Toolkit/On-Chip Debug Port** option on the second core, connect it to the next core using the **Export** option as described above. Repeat the process for subsequent EMIF cores.

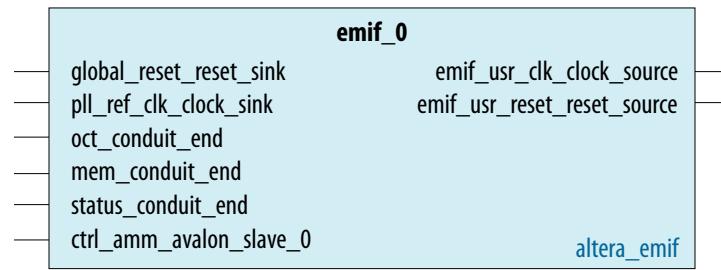
If you place any PHYLITE cores with dynamic reconfiguration enabled into the same I/O column as an EMIF IP core, you should instantiate and connect the PHYLITE cores in a similar way. See the *Altera PHYLITE for Memory Megafunction User Guide* for more information.

## Arria 10 EMIF Debugging Examples

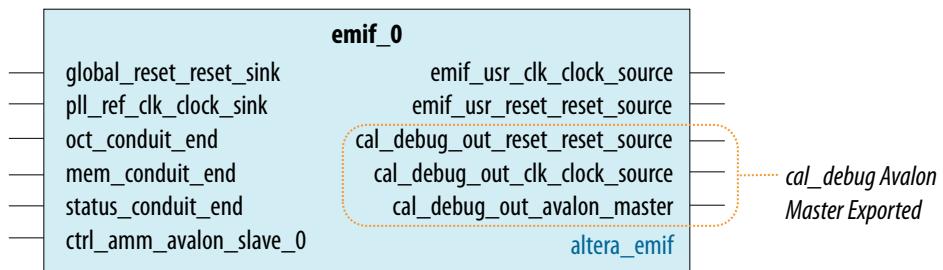
This topic provides examples of debugging a single external memory interface, and of adding additional EMIF instances to an I/O column.

### Debugging a Single External Memory Interface

1. Under **EMIF Debug Toolkit/On-Chip Debug Port**, select **Add EMIF Debug Interface**.  
(If you want to use the On-Chip Debug Port instead of the EMIF Debug Toolkit, select **Export** instead.)

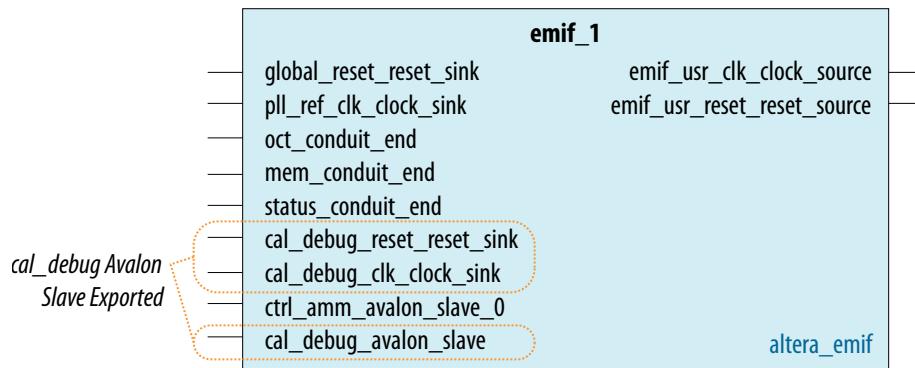
**Figure 2-23: EMIF With Debug Interface Added (No Additional Ports)**

2. If you want to connect additional EMIF or PHYLite components in this I/O column, select *Enable Daisy Chaining for EMIF Debug Toolkit/On-Chip Debug Port*.

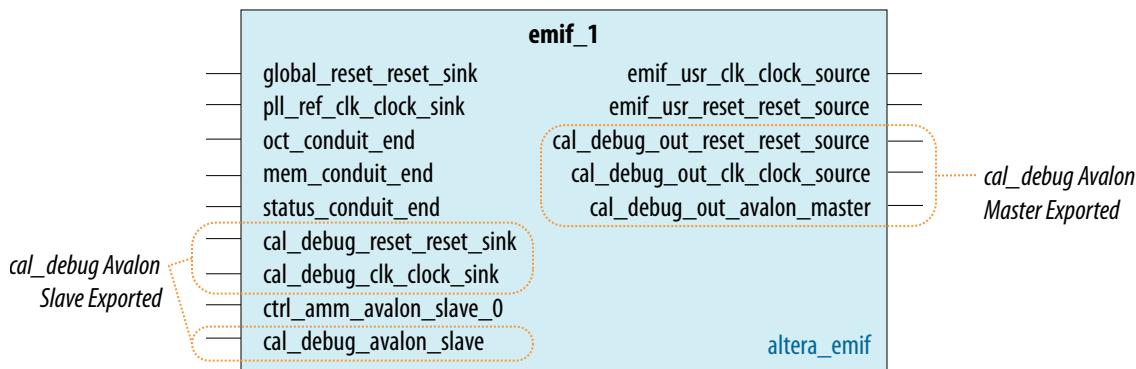
**Figure 2-24: EMIF With cal\_debug Avalon Master Exported**

### Adding Additional EMIF Instances to an I/O Column

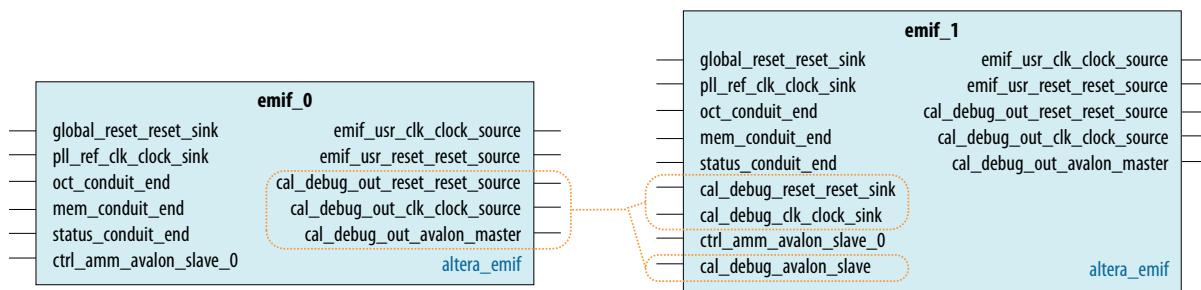
1. Under EMIF Debug Toolkit/On-Chip Debug Port, select **Export**.

**Figure 2-25: EMIF With cal\_debug Avalon Slave Exported**

2. Specify a unique interface ID for this EMIF instance.
3. If you want to connect additional EMIF or PHYLite components in this I/O column, select *Enable Daisy Chaining for EMIF Debug Toolkit/On-Chip Debug Port*.

**Figure 2-26: EMIF With Both cal\_debug Master and Slave Exported**

4. Connect the `cal_debug` Avalon Master, clock, and reset interfaces of the previous component to the `cal_debug` Avalon Slave, clock, and reset interfaces of this component.

**Figure 2-27: EMIF Components Connected**

## Arria 10 EMIF for Hard Processor Subsystem

The Arria 10 EMIF IP can enable the Arria 10 Hard Processor Subsystem (HPS) to access external DRAM memory devices.

To enable connectivity between the Arria 10 HPS and the Arria 10 EMIF IP, you must create and configure an instance of the Arria 10 External Memory Interface for HPS IP core, and use Qsys to connect it to the Arria 10 Hard Processor Subsystem instance in your system.

### Supported Modes

The Arria 10 Hard Processor Subsystem is compatible with the following external memory configurations:

Protocol	DDR3, DDR4
Maximum memory clock frequency	DDR3: 1.067 GHz DDR4: 1.333 GHz
Configuration	Hard PHY with hard memory controller
Clock rate of PHY and hard memory controller	Half-rate

Data width (without ECC)	16-bit, 32-bit, 64-bit <sup>2</sup>
Data width (with ECC)	24-bit, 40-bit, 72-bit <sup>2</sup>
DQ width per group	x8
Maximum number of I/O lanes for address/ command	3
Memory format	Discrete, UDIMM, SODIMM, RDIMM
Ranks / CS# width	Up to 2

Notes to table:

1. Only Arria 10 devices with a special ordering code support 64-bit and 72-bit data widths; all other devices support only to 32-bit data widths.

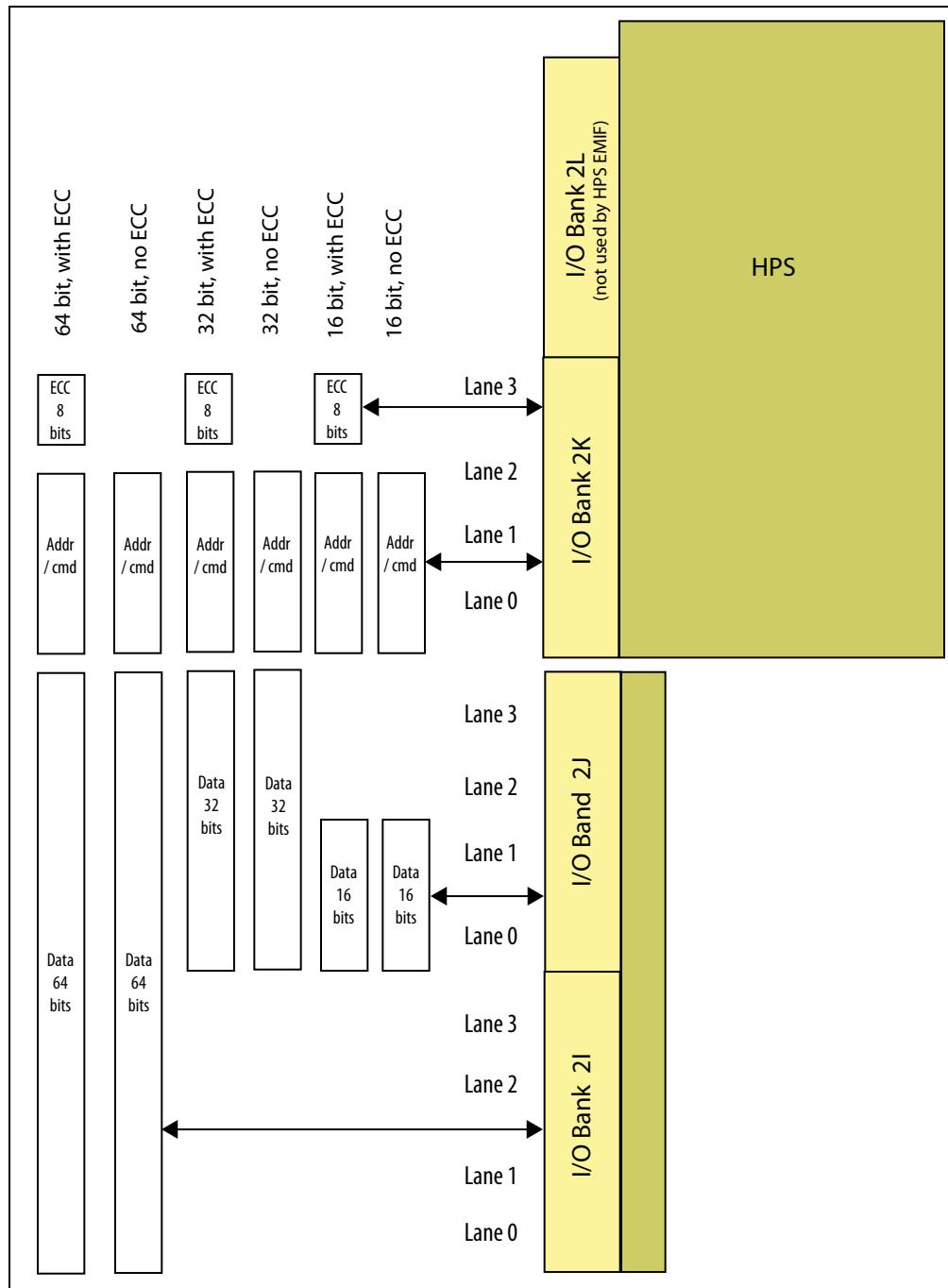
## Restrictions on I/O Bank Usage for Arria 10 EMIF IP with HPS

Only certain Arria 10 I/O banks can be used to implement Arria 10 EMIF IP with the Arria 10 Hard Processor Subsystem.

The restrictions on I/O bank usage result from the Arria 10 HPS having hard-wired connections to the EMIF circuits in the I/O banks closest to the HPS. For any given EMIF configuration, the pin-out of the EMIF-to-HPS interface is fixed.

The following diagram illustrates the use of I/O banks and lanes for various EMIF-HPS data widths:

Figure 2-28: I/O Banks and Lanes Usage



You should refer to the pinout file for your device and package for detailed information. Banks and pins used for HPS access to a DDR interface are labeled HPS\_DDR in the HPS Function column of the pinout file.

By default, the Arria 10 External Memory Interface for HPS IP core together with the Quartus Prime Fitter automatically implement the correct pin-out for HPS EMIF without you having to implement

additional constraints. If, for any reason, you must modify the default pin-out, you must adhere to the following requirements, which are specific to HPS EMIF:

1. Within a single data lane (which implements a single x8 DQS group):
  - a. DQ pins must use pins at indices 1, 2, 3, 6, 7, 8, 9, 10. You may swap the locations between the DQ bits (that is, you may swap location of DQ[0] and DQ[3]) so long as the resulting pin-out uses pins at these indices only.
  - b. DM/DBI pin must use pin at index 11. There is no flexibility.
  - c. DQS/DQS# must use pins at index 4 and 5. There is no flexibility.
2. Assignment of data lanes must be as illustrated in the above figure. You are allowed to swap the locations of entire byte lanes (that is, you may swap locations of byte 0 and byte 3) so long as the resulting pin-out uses only the lanes permitted by your HPS EMIF configuration, as shown in the above figure.
3. You must not change placement of the address and command pins from the default.
4. You may place the `alert#` pin at any available pin location in either a data lane or an address and command lane.

To override the default generated pin assignments, comment out the relevant `HPS_LOCATION` assignments in the `.qip` file, and add your own location assignments (using `set_location_assignment`) in the `.qsf` file.

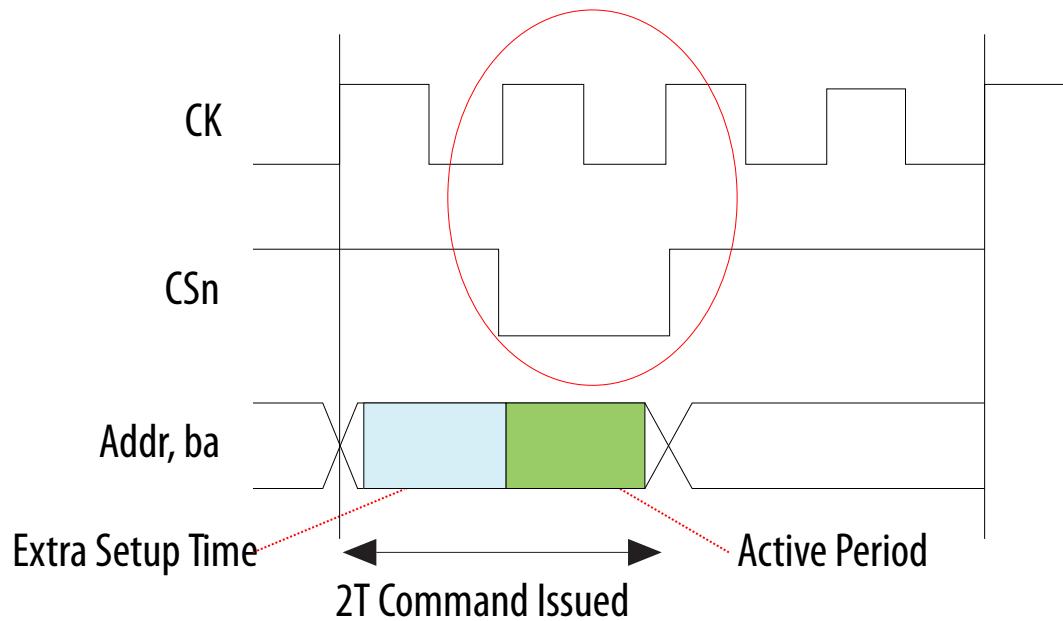
## Arria 10 EMIF Ping Pong PHY

Ping Pong PHY allows two memory interfaces to share the address and command bus through time multiplexing. Compared to having two independent interfaces that allocate address and command lanes separately, Ping Pong PHY achieves the same throughput with fewer resources, by sharing the address and command lanes.

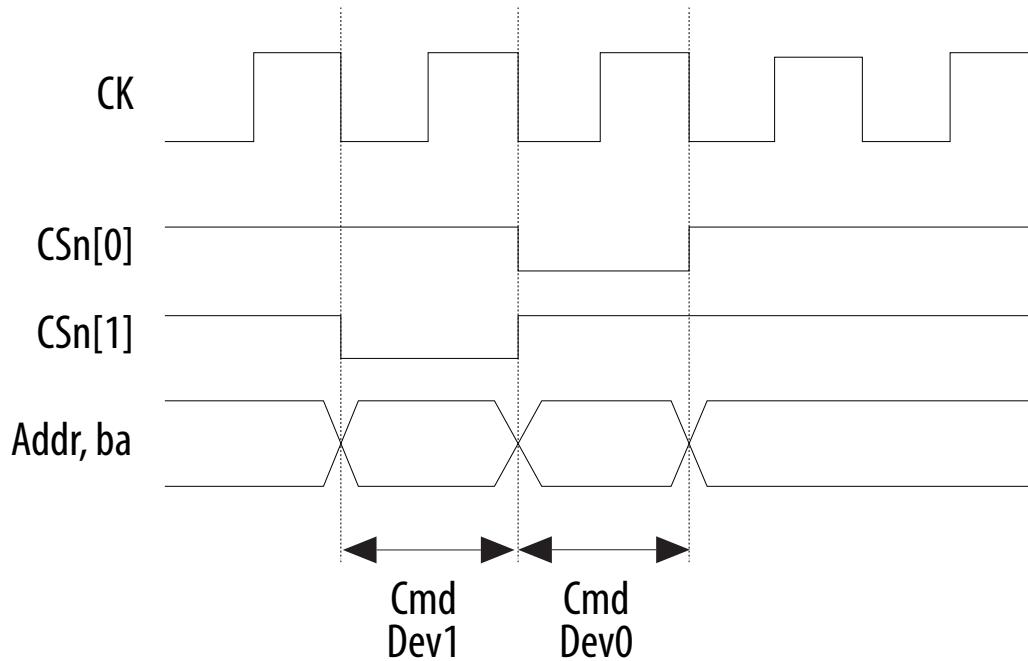
In Arria 10 EMIF, Ping Pong PHY supports both half-rate and quarter-rate interfaces for DDR3, and quarter-rate for DDR4.

### Ping Pong PHY Feature Description

Conventionally, the address and command buses of a DDR3 or DDR4 half-rate or quarter-rate interface use 2T time—meaning that commands are issued for two full-rate clock cycles, as illustrated below.

**Figure 2-29: 2T Command Timing**

With the Ping Pong PHY, address and command signals from two independent controllers are multiplexed onto shared buses by delaying one of the controller outputs by one full-rate clock cycle. The result is 1T timing, with a new command being issued on each full-rate clock cycle. The following figure shows address and command timing for the Ping Pong PHY.

**Figure 2-30: 1T Command Timing Use by Ping Pong PHY**

## Ping Pong PHY Architecture

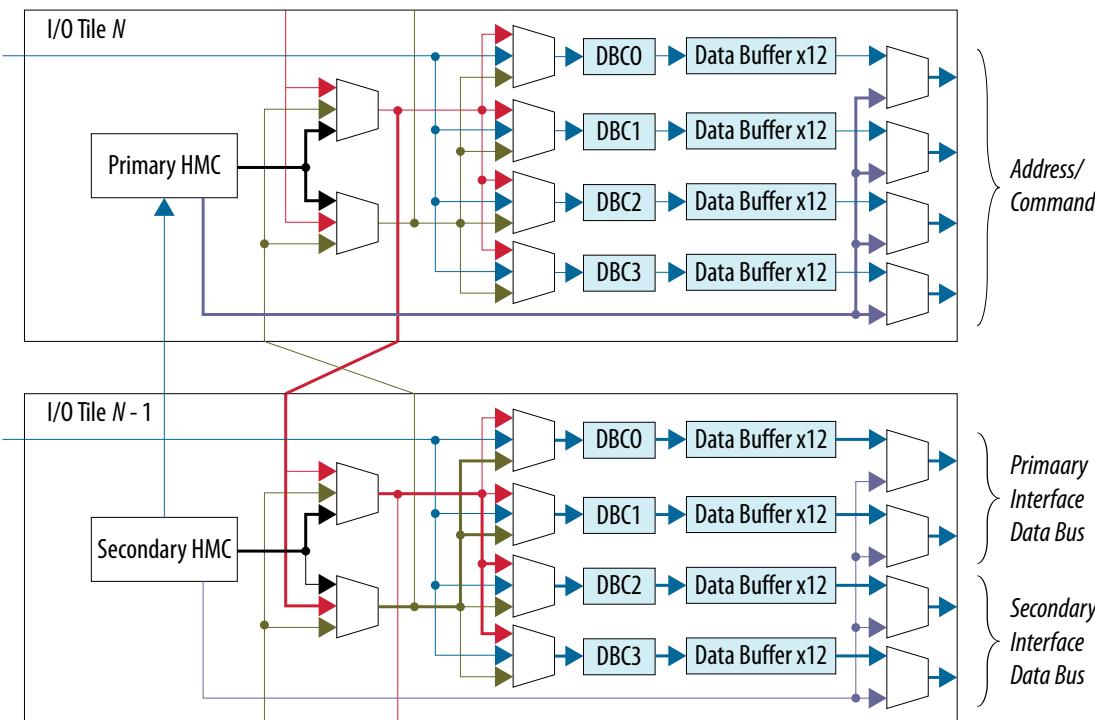
In Arria 10 EMIF, the Ping Pong PHY feature can be enabled only with the hard memory controller, where two hard memory controllers are instantiated—one for the primary interface and one for the secondary interface.

The hard memory controller I/O bank of the primary interface is used for address and command and is always adjacent and above the hard memory controller bank of the secondary interface. All four lanes of the primary hard memory controller bank are used for address and command. The I/O bank containing the secondary hard memory controller must have at least one lane from the secondary interface.

The following example shows a 2x16 Ping Pong PHY bank-lane configuration. The upper bank (I/O bank N) is the address and command bank, which serves both the primary and secondary interfaces. The primary hard memory controller is linked to the secondary interface by the Ping Pong bus. The lower bank (I/O bank N-1) is the secondary interface bank, which carries the data buses for both primary and secondary interfaces. In the 2x16 case a total of four I/O banks are required for data, hence two banks in total are sufficient for the implementation.

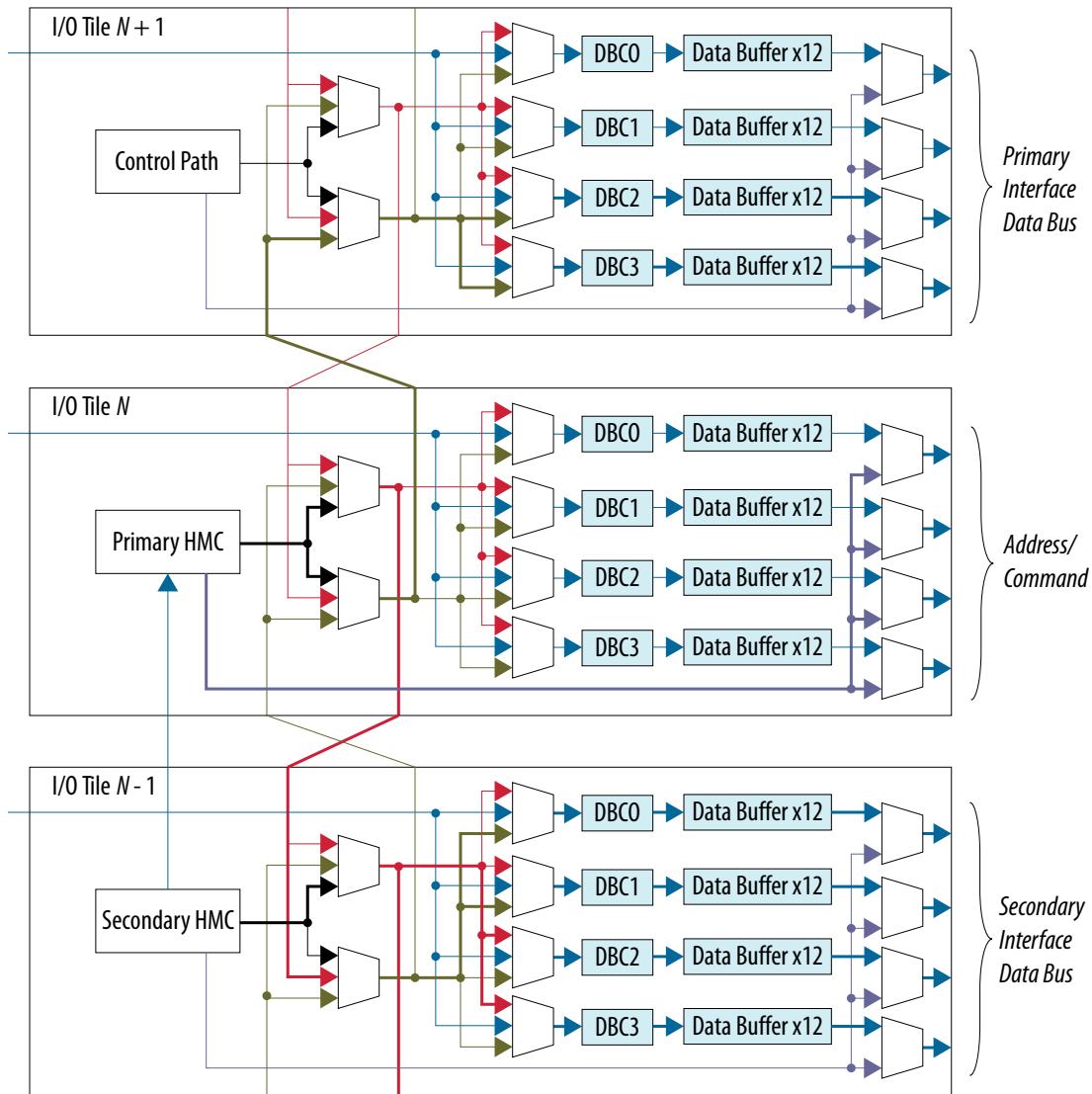
The data for the primary interface is routed down to the top two lanes of the secondary I/O bank, and the data for the secondary interface is routed to the bottom two lanes of the secondary I/O bank.

**Figure 2-31: 2x16 Ping Pong PHY I/O Bank-Lane Configuration**



A 2x32 interface can be implemented using three tiles, so long as the tile containing the secondary hard memory controller has at least one secondary data lane. The order of the lanes does not matter.

Figure 2-32: 2x32 Ping Pong PHY I/O Bank-Lane Configuration.



## Ping Pong PHY Limitations

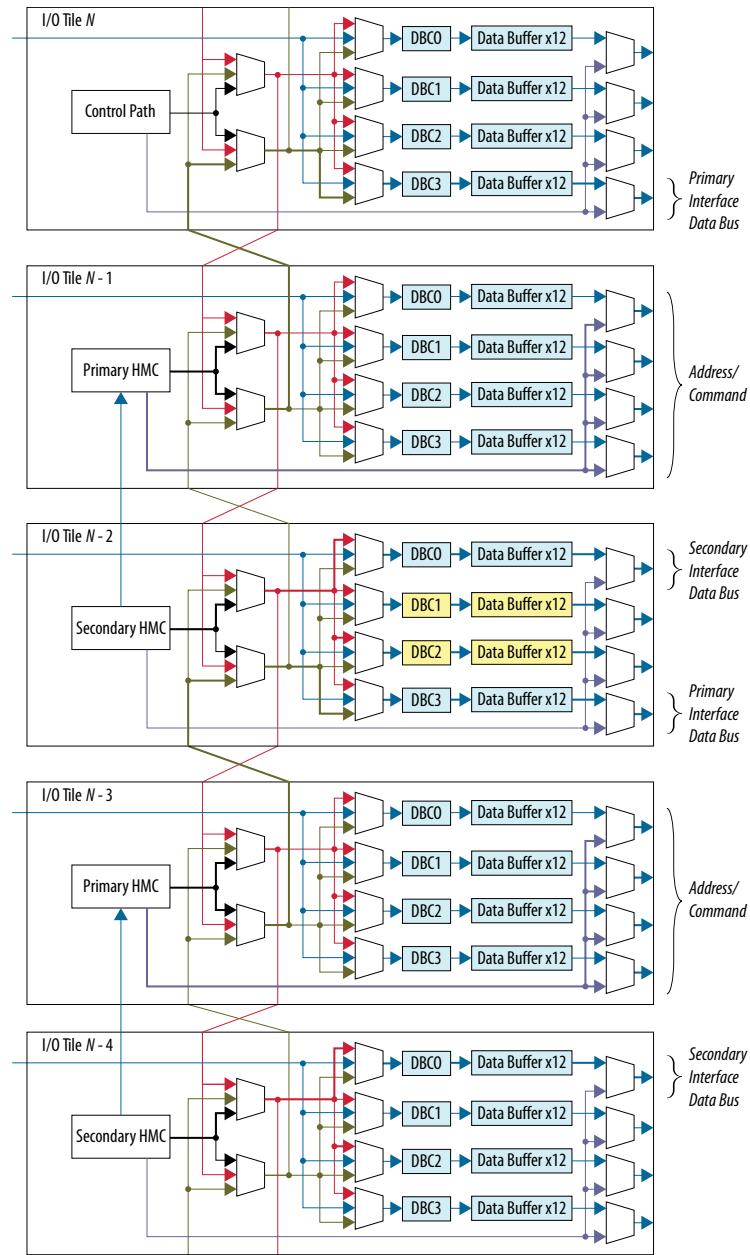
Ping Pong PHY supports up to two ranks per memory interface. In addition, the maximum data width is x72, which is half the maximum width of x144 for a single interface.

Ping Pong PHY uses all lanes of the address and command I/O bank as address and command. For information on the pin allocations of the DDR3 and DDR4 address and command I/O bank, refer to *DDR3 Scheme 1* and *DDR4 Scheme 3*, in *External Memory Interface Pin Information for Arria 10 Devices*, on the Altera web site.

An additional limitation is that I/O lanes may be left unused when you instantiate multiple pairs of Ping Pong PHY interfaces. The following diagram shows two pairs of x8 Pin Pong controllers (a total of 4 interfaces). Lanes highlighted in yellow are not driven by any memory interfaces (unused lanes and pins can still serve as general purpose I/Os). Even with some I/O lanes left unused, the Ping Pong PHY approach is still beneficial in terms of resource usage, compared to independent interfaces. Memory

widths of 24 bits and 40 bits have a similar situation, while 16 bit, 32 bit, and 64 bit memory widths do not suffer this limitation.

**Figure 2-33: Two Pairs of x8 Pin-Pong PHY Controllers**



#### Related Information

[External Memory Interface Pin Information for Arria 10 Devices](#)

## Ping Pong PHY Calibration

A Ping Pong PHY interface is calibrated as a regular interface of double width.

Calibration of a Ping Pong PHY interface incorporates two sequencers, one on the primary hard memory controller I/O bank, and one on the secondary hard memory controller I/O bank. To ensure that the two sequencers issue instructions on the same memory clock cycle, the Nios II processor configures the sequencer on the primary hard memory controller to receive a token from the secondary interface, ignoring any commands from the Avalon bus. Additional delays are programmed on the secondary interface to allow for the passing of the token from the sequencer on the secondary hard memory controller tile to the sequencer on the primary hard memory controller tile. During calibration, the Nios II processor assumes that commands are always issued from the sequencer on the primary hard memory controller I/O bank. After calibration, the Nios II processor adjusts the delays for use with the primary and secondary hard memory controllers.

## Using the Ping Pong PHY

The following steps describe how to use the Ping Pong PHY for Arria 10 EMIF.

1. Configure a single memory interface according to your requirements.
2. Select **Instantiate two controllers sharing a Ping Pong PHY** on the **General** tab in the parameter editor.

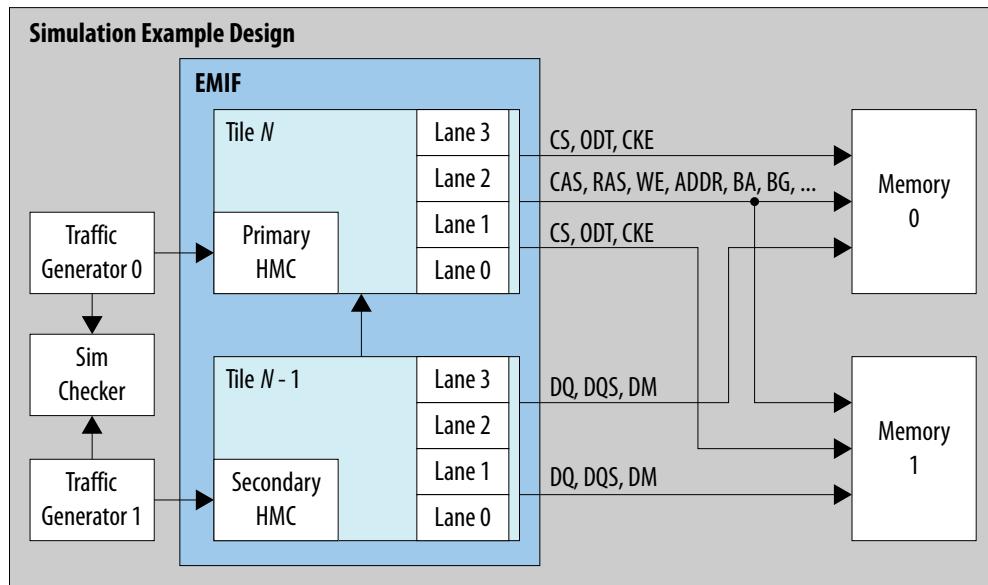
The Quartus Prime software replicates the interface, resulting in two memory controllers and a shared PHY. The system configures the I/O bank-lane structure, without further input from you.

## Ping Pong PHY Simulation Example Design

The following figure illustrates a top-level block diagram of a generated Ping Pong PHY simulation example design, using two I/O banks.

Functionally, the IP interfaces with user traffic separately, as it would with two independent memory interfaces. You can also generate synthesizable example designs, where the external memory interface IP interfaces with a traffic generator.

**Figure 2-34: Ping Pong PHY Simulation Example Design**



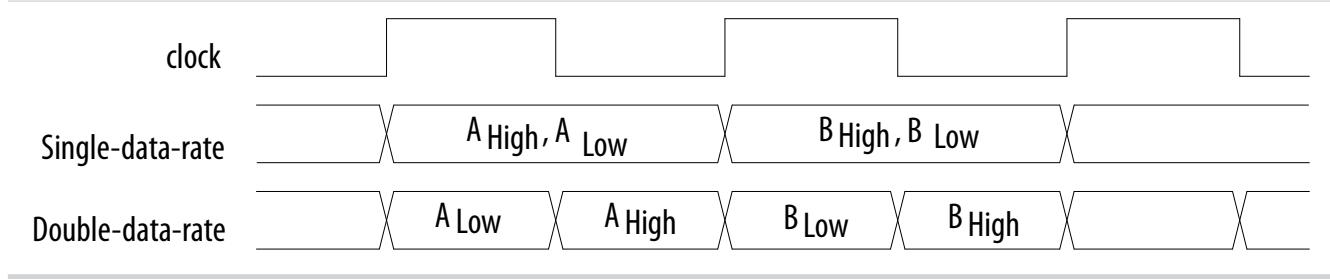
## AFI 4.0 Specification

The Altera PHY interface (AFI) 4.0 defines communication between the controller and physical layer (PHY) in the external memory interface IP.

The AFI is a single-data-rate interface, meaning that data is transferred on the rising edge of each clock cycle. Most memory interfaces, however, operate at double-data-rate, transferring data on both the rising and falling edges of the clock signal. If the AFI interface is to directly control a double-data-rate signal, two single-data-rate bits must be transmitted on each clock cycle; the PHY then sends out one bit on the rising edge of the clock and one bit on the falling edge.

The AFI convention is to send the low part of the data first and the high part second, as shown in the following figure.

**Figure 2-35: Single Versus Double Data Rate Transfer**



## Bus Width and AFI Ratio

In cases where the AFI clock frequency is one-half or one-quarter of the memory clock frequency, the AFI data must be twice or four times as wide, respectively, as the corresponding memory data. The ratio between AFI clock and memory clock frequencies is referred to as the AFI ratio. (A half-rate AFI interface has an AFI ratio of 2, while a quarter-rate interface has an AFI ratio of 4.)

In general, the width of the AFI signal depends on the following three factors:

- The size of the equivalent signal on the memory interface. For example, if `a[15:0]` is a DDR3 address input and the AFI clock runs at the same speed as the memory interface, the equivalent `afi_addr` bus will be 16-bits wide.
- The data rate of the equivalent signal on the memory interface. For example, if `d[7:0]` is a double-data-rate QDR II input data bus and the AFI clock runs at the same speed as the memory interface, the equivalent `afi_write_data` bus will be 16-bits wide.
- The AFI ratio. For example, if `cs_n` is a single-bit DDR3 chip select input and the AFI clock runs at half the speed of the memory interface, the equivalent `afi_cs_n` bus will be 2-bits wide.

The following formula summarizes the three factors described above:

$$\text{AFI\_width} = \text{memory\_width} * \text{signal\_rate} * \text{AFI\_RATE\_RATIO}$$

**Note:** The above formula is a general rule, but not all signals obey it. For definite signal-size information, refer to the specific table.

## AFI Parameters

The following tables list Altera PHY interface (AFI) parameters for AFI 4.0. Not all parameters are used for all protocols.

The parameters described in the following tables affect the width of AFI signal buses. Parameters prefixed by `MEM_IF_` refer to the signal size at the interface between the PHY and memory device.

**Table 2-7: Ratio Parameters**

Parameter Name	Description
AFI_RATE_RATIO	The ratio between the AFI clock frequency and the memory clock frequency. For full-rate interfaces this value is 1, for half-rate interfaces the value is 2, and for quarter-rate interfaces the value is 4.
DATA_RATE_RATIO	The number of data bits transmitted per clock cycle. For single-data rate protocols this value is 1, and for double-data rate protocols this value is 2.
ADDR_RATE_RATIO	The number of address bits transmitted per clock cycle. For single-data rate address protocols this value is 1, and for double-data rate address protocols this value is 2.

**Table 2-8: Memory Interface Parameters**

Parameter Name	Description
MEM_IF_ADDR_WIDTH	The width of the address bus on the memory device(s). For LPDDR3, the width of the CA bus, which encodes commands and addresses together.
MEM_IF_BANKGROUP_WIDTH	The width of the bank group bus on the interface to the memory device(s). Applicable to DDR4 only.
MEM_IF_BANKADDR_WIDTH	The width of the bank address bus on the interface to the memory device(s). Typically, the $\log_2$ of the number of banks. Not applicable to DDR4.
MEM_IF_CS_WIDTH	The number of chip selects on the interface to the memory device(s).
MEM_IF_CKE_WIDTH	Number of CKE signals on the interface to the memory device(s). This usually equals to <code>MEM_IF_CS_WIDTH</code> except for certain DIMM configurations.
MEM_IF_ODT_WIDTH	Number of ODT signals on the interface to the memory device(s). This usually equals to <code>MEM_IF_CS_WIDTH</code> except for certain DIMM configurations.
MEM_IF_WRITE_DQS_WIDTH	The number of DQS (or write clock) signals on the write interface. For example, the number of DQS groups.
MEM_IF_CLK_PAIR_COUNT	The number of CK/CK# pairs.

Parameter Name	Description
MEM_IF_DQ_WIDTH	The number of DQ signals on the interface to the memory device(s). For single-ended interfaces such as QDR II, this value is the number of D or Q signals.
MEM_IF_DM_WIDTH	The number of data mask pins on the interface to the memory device(s).

**Table 2-9: Derived AFI Parameters**

Parameter Name	Derivation Equation
AFI_ADDR_WIDTH	MEM_IF_ADDR_WIDTH * AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_BANKGROUP_WIDTH	MEM_IF_BANKGROUP_WIDTH * AFI_RATE_RATIO * ADDR_RATE_RATIO. Applicable to DDR4 only.
AFI_BANKADDR_WIDTH	MEM_IF_BANKADDR_WIDTH * AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_CONTROL_WIDTH	AFI_RATE_RATIO * ADDR_RATE_RATIO
AFI_CS_WIDTH	MEM_IF_CS_WIDTH * AFI_RATE_RATIO
AFI_CKE_WIDTH	MEM_IF_CKE_WIDTH * AFI_RATE_RATIO
AFI_ODT_WIDTH	MEM_IF_ODT_WIDTH * AFI_RATE_RATIO
AFI_DM_WIDTH	MEM_IF_DM_WIDTH * AFI_RATE_RATIO * DATA_RATE_RATIO
AFI_DQ_WIDTH	MEM_IF_DQ_WIDTH * AFI_RATE_RATIO * DATA_RATE_RATIO
AFI_WRITE_DQS_WIDTH	MEM_IF_WRITE_DQS_WIDTH * AFI_RATE_RATIO
AFI_LAT_WIDTH	6
AFI_RLAT_WIDTH	AFI_LAT_WIDTH
AFI_WLAT_WIDTH	AFI_LAT_WIDTH * MEM_IF_WRITE_DQS_WIDTH
AFI_CLK_PAIR_COUNT	MEM_IF_CLK_PAIR_COUNT
AFI_WRANK_WIDTH	Number of ranks * MEM_IF_WRITE_DQS_WIDTH * AFI_RATE_RATIO
AFI_RRANK_WIDTH	Number of ranks * MEM_IF_READ_DQS_WIDTH * AFI_RATE_RATIO

## AFI Signals

The following tables list Altera PHY interface (AFI) signals grouped according to their functions.

In each table, the **Direction** column denotes the direction of the signal relative to the PHY. For example, a signal defined as an output passes out of the PHY to the controller. The AFI specification does not include any bidirectional signals.

Not all signals are used for all protocols.

### AFI Clock and Reset Signals

The AFI interface provides up to two clock signals and an asynchronous reset signal.

**Table 2-10: Clock and Reset Signals**

Signal Name	Direction	Width	Description
afi_clk	Output	1	Clock with which all data exchanged on the AFI bus is synchronized. In general, this clock is referred to as full-rate, half-rate, or quarter-rate, depending on the ratio between the frequency of this clock and the frequency of the memory device clock.
afi_half_clk	Output	1	Clock signal that runs at half the speed of the afi_clk. The controller uses this signal when the half-rate bridge feature is in use. This signal is optional.
afi_reset_n	Output	1	Asynchronous reset output signal. You must synchronize this signal to the clock domain in which you use it.

### AFI Address and Command Signals

The address and command signals for AFI 4.0 encode read/write/configuration commands to send to the memory device. The address and command signals are single-data rate signals.

**Table 2-11: Address and Command Signals**

Signal Name	Direction	Width	Description
afi_addr	Input	AFI_ADDR_WIDTH	Address or CA bus (LPDDR3 only). ADDR_RATE_RATIO is 2 for LPDDR3 CA bus.
afi_bg	Input	AFI_BANKGROUP_WIDTH	Bank group (DDR4 only).
afi_ba	Input	AFI_BANKADDR_WIDTH	Bank address. (Not applicable for LPDDR3.)
afi_cke	Input	AFI_CLK_EN_WIDTH	Clock enable.

Signal Name	Direction	Width	Description
afi_cs_n	Input	AFI_CS_WIDTH	Chip select signal. (The number of chip selects may not match the number of ranks; for example, RDIMMs and LRDIMMs require a minimum of 2 chip select signals for both single-rank and dual-rank configurations. Consult your memory device data sheet for information about chip select signal width.) (Matches the number of ranks for LPDDR3.)
afi_ras_n	Input	AFI_CONTROL_WIDTH	RAS# (for DDR2 and DDR3 memory devices.)
afi_we_n	Input	AFI_CONTROL_WIDTH	WE# (for DDR2, DDR3, and RLDRAM II memory devices.)
afi_rw_n	Input	AFI_CONTROL_WIDTH * 2	RWA/B# (QDR-IV).
afi_cas_n	Input	AFI_CONTROL_WIDTH	CAS# (for DDR2 and DDR3 memory devices.)
afi_act_n	Input	AFI_CONTROL_WIDTH	ACT# (DDR4).
afi_ref_n	Input	AFI_CONTROL_WIDTH	REF# (for RLDRAM II memory devices.)
afi_RST_n	Input	AFI_CONTROL_WIDTH	RESET# (for DDR3 and DDR4 memory devices.)
afi_odt	Input	AFI_CLK_EN_WIDTH	On-die termination signal for DDR2, DDR3, and LPDDR3 memory devices. (Do not confuse this memory device signal with the FPGA's internal on-chip termination signal.)
afi_par	Input	AFI_CS_WIDTH	Address and command parity input. (DDR4) Address parity input. (QDR-IV)
afi_alert_n	Output	AFI_CS_WIDTH	Alert signal to indicate parity or CRC error. (DDR4) Address parity error, PE# (QDR-IV)
afi_ainv	Input	AFI_CONTROL_WIDTH	Address inversion. (QDR-IV)

Signal Name	Direction	Width	Description
afi_mem_clk_disable	Input	AFI_CLK_PAIR_COUNT	When this signal is asserted, mem_clk and mem_clk_n are disabled. This signal is used in low-power mode.
afi_wps_n	Output	AFI_CS_WIDTH	WPS (for QDR II/II+ memory devices.)
afi_rps_n	Output	AFI_CS_WIDTH	RPS (for QDR II/II+ memory devices.)

## AFI Write Data Signals

Write Data Signals for AFI 4.0 control the data, data mask, and strobe signals passed to the memory device during write operations.

**Table 2-12: Write Data Signals**

Signal Name	Direction	Width	Description
afi_dqs_burst	Input	AFI_RATE_RATIO	Controls the enable on the strobe (DQS) pins for DDR2, DDR3, LPDDR2, and LPDDR3 memory devices. When this signal is asserted, mem_dqs and mem_dqsn are driven.  This signal must be asserted before afi_wdata_valid to implement the write preamble, and must be driven for the correct duration to generate a correctly timed mem_dqs signal.
afi_wdata_valid	Input	AFI_RATE_RATIO	Write data valid signal. This signal controls the output enable on the data and data mask pins.
afi_wdata	Input	AFI_DQ_WIDTH	Write data signal to send to the memory device at double-data rate. This signal controls the PHY's mem_dq output.
afi_dm / afi_db	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_dm signal for DDR2, DDR3, LPDDR2, LPDDR3, and RLDRAM II memory devices.)  Also directly controls the PHY's mem_db signal for DDR4.  The mem_dm and mem_db features share the same port on the memory device.

Signal Name	Direction	Width	Description
afi_bws_n	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_bws_n signal for QDR II/II+ memory devices.
afi_dinv	Input	AFI_WRITE_DQS_WIDTH * 2	Data inversion. It directly controls the PHY's mem_dinva/b signal for QDR-IV devices.

### AFI Read Data Signals

Read Data Signals for AFI 4.0 control the data sent from the memory device during read operations.

**Table 2-13: Read Data Signals**

Signal Name	Direction	Width	Description
afi_rdata_en_full	Input	AFI_RATE_RATIO	Read data enable full. Indicates that the memory controller is currently performing a read operation. This signal is held high for the entire read burst. If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rdata	Output	AFI_DQ_WIDTH	Read data from the memory device. This data is considered valid only when afi_rdata_valid is asserted by the PHY.
afi_rdata_valid	Output	AFI_RATE_RATIO	Read data valid. When asserted, this signal indicates that the afi_rdata bus is valid. If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).

### AFI Calibration Status Signals

The PHY instantiates a sequencer which calibrates the memory interface with the memory device and some internal components such as read FIFOs and valid FIFOs. The sequencer reports the results of the calibration process to the controller through the Calibration Status Signals in the AFI interface.

**Table 2-14: Calibration Status Signals**

Signal Name	Direction	Width	Description
afi_cal_success	Output	1	Asserted to indicate that calibration has completed successfully.

Signal Name	Direction	Width	Description
afi_cal_fail	Output	1	Asserted to indicate that calibration has failed.
afi_cal_req	Input	1	Effectively a synchronous reset for the sequencer. When this signal is asserted, the sequencer returns to the reset state; when this signal is released, a new calibration sequence begins.
afi_wlat	Output	AFI_WLAT_WIDTH	The required write latency in afi_clk cycles, between address/command and write data being issued at the PHY/controller interface. The afi_wlat value can be different for different groups; each group's write latency can range from 0 to 63. If write latency is the same for all groups, only the lowest 6 bits are required.
afi_rlat (1)	Output	AFI_RLAT_WIDTH	The required read latency in afi_clk cycles between address/command and read data being returned to the PHY/controller interface. Values can range from 0 to 63.

Note to Table:

1. The afi\_rlat signal is not supported for PHY-only designs. Instead, you can sample the afi\_rdata\_valid signal to determine when valid read data is available.

## AFI Tracking Management Signals

When tracking management is enabled, the sequencer can take control over the AFI 4.0 interface at given intervals, and issue commands to the memory device to track the internal DQS Enable signal alignment to the DQS signal returning from the memory device. The tracking management portion of the AFI 4.0 interface provides a means for the sequencer and the controller to exchange handshake signals.

**Table 2-15: Tracking Management Signals**

Signal Name	Direction	Width	Description
afi_ctl_refresh_done	Input	4	Handshaking signal from controller to tracking manager, indicating that a refresh has occurred and waiting for a response.
afi_seq_busy	Output	4	Handshaking signal from sequencer to controller, indicating when DQS tracking is in progress.

Signal Name	Direction	Width	Description
afi_ctl_long_idle	Input	4	Handshaking signal from controller to tracking manager, indicating that it has exited low power state without a periodic refresh, and waiting for response.

## AFI Shadow Register Management Signals

Shadow registers are a feature that enables high-speed multi-rank support. Shadow registers allow the sequencer to calibrate each rank separately, and save the calibrated settings—such as deskew delay-chain configurations—of each rank in its own set of shadow registers.

During a rank-to-rank switch, the correct set of calibrated settings is restored just in time to optimize the data valid window. The PHY relies on additional AFI signals to control which set of shadow registers to activate.

**Table 2-16: Shadow Register Management Signals**

Signal Name	Direction	Width	Description
afi_wrrank	Input	AFI_WRANK_WIDTH	Signal from controller specifying which rank the write data is going to. The signal timing is identical to that of afi_dqs_burst. That is, afi_wrrank must be asserted at the same time and must last the same duration as the afi_dqs_burst signal.
afi_rrank	Output	AFI_RRANK_WIDTH	Signal from controller specifying which rank is being read. The signal must be asserted at the same time as the afi_rdata_en signal when issuing a read command, but unlike afi_rdata_en, afi_rrank is stateful. That is, once asserted, the signal value must remain unchanged until the controller issues a new read command to a different rank.

Both the afi\_wrrank and afi\_rrank signals encode the rank being accessed using the one-hot scheme (e.g. in a quad-rank interface, 0001, 0010, 0100, 1000 refer to the 1<sup>st</sup>, 2<sup>nd</sup>, 3<sup>rd</sup>, 4<sup>th</sup> rank respectively). The ordering within the bus is the same as other AFI signals. Specifically the bus is ordered by time slots, for example:

```
Half-rate afi_w/rrank = {T1, T0}
```

```
Quarter-rate afi_w/rrank = {T3, T2, T1, T0}
```

Where  $T_x$  is a number of rank-bit words that one-hot encodes the rank being accessed at the  $y^{\text{th}}$  full-rate cycle.

### Additional Requirements for Arria 10 Shadow Register Support

To ensure that the hardware has enough time to switch from one shadow register to another, the controller must satisfy the following minimum rank-to-rank-switch delays (tRTRS):

- Two read commands going to different ranks must be separated by a minimum of 3 full-rate cycles (in addition to the burst length delay needed to avoid collision of data bursts).
- Two write commands going to different rank must be separated by a minimum of 4 full-rate cycles (in addition to the burst length delay needed to avoid collision of data bursts).

The Arria 10 device family supports a maximum of 4 sets of shadow registers, each for an independent set of timings. More than 4 ranks are supported if those ranks have four or fewer sets of independent timing. For example, the rank multiplication mode of an LRDIMM allows more than one physical rank to share a set of timing data as a single logical rank. Therefore Arria 10 devices can support up to 4 logical ranks, though that means more than 4 physical ranks.

## AFI 4.0 Timing Diagrams

### AFI Address and Command Timing Diagrams

Depending on the ratio between the memory clock and the PHY clock, different numbers of bits must be provided per PHY clock on the AFI interface. The following figures illustrate the AFI address/command waveforms in full, half and quarter rate respectively.

The waveforms show how the AFI command phase corresponds to the memory command output. AFI command 0 corresponds to the first memory command slot, AFI command 1 corresponds to the second memory command slot, and so on.

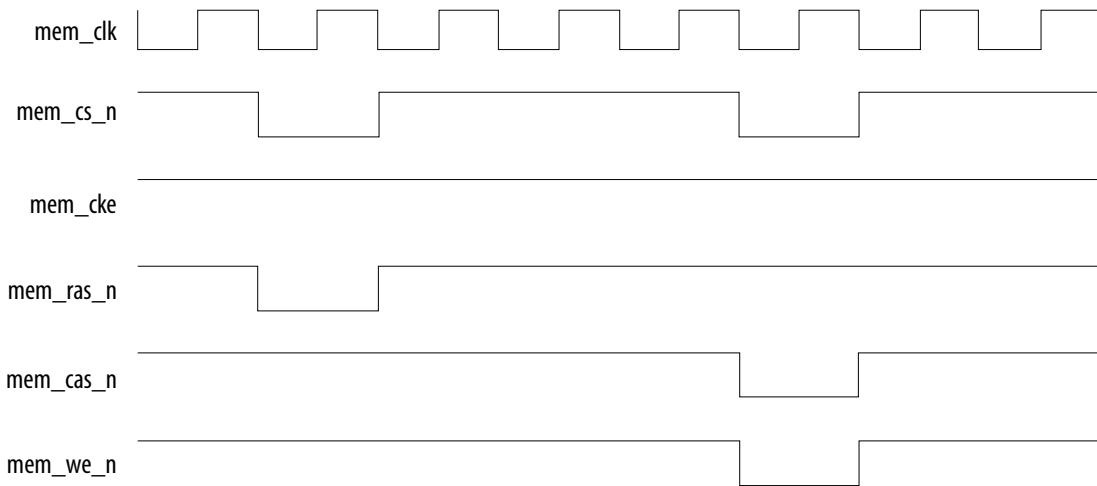
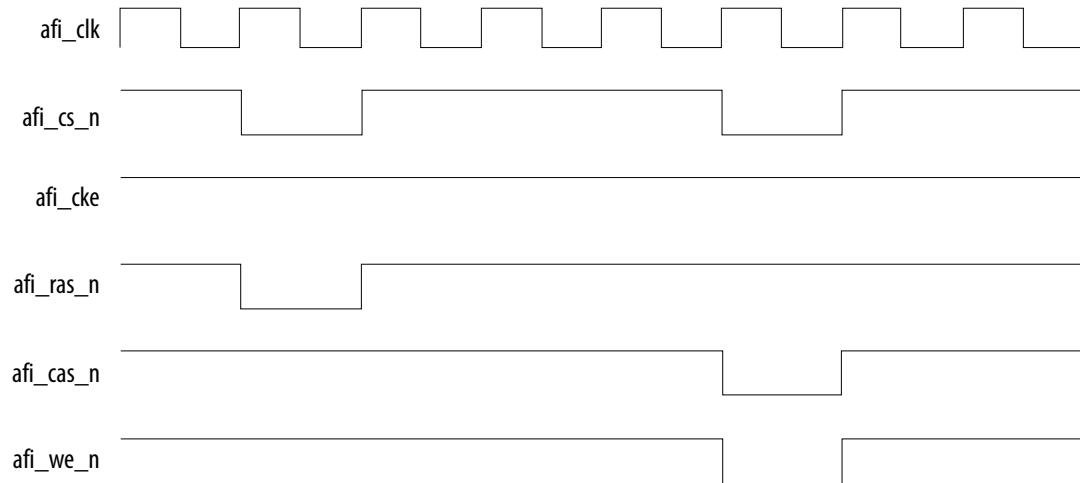
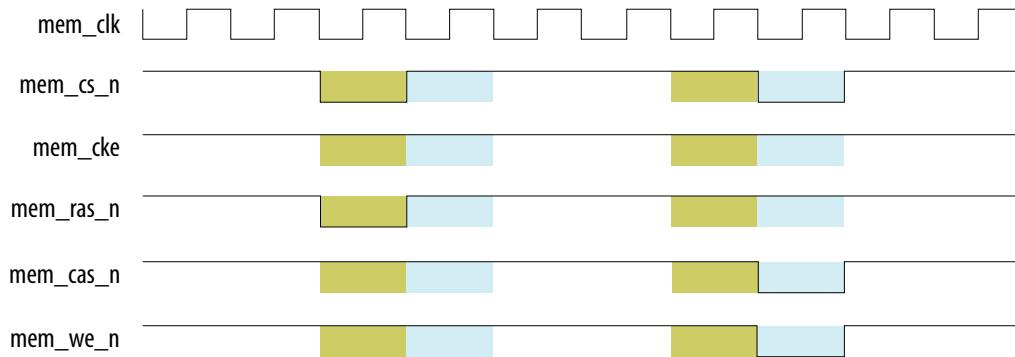
**Figure 2-36: AFI Address and Command Full-Rate****Memory Interface****AFI Interface**

Figure 2-37: AFI Address and Command Half-Rate

### Memory Interface



### AFI Interface

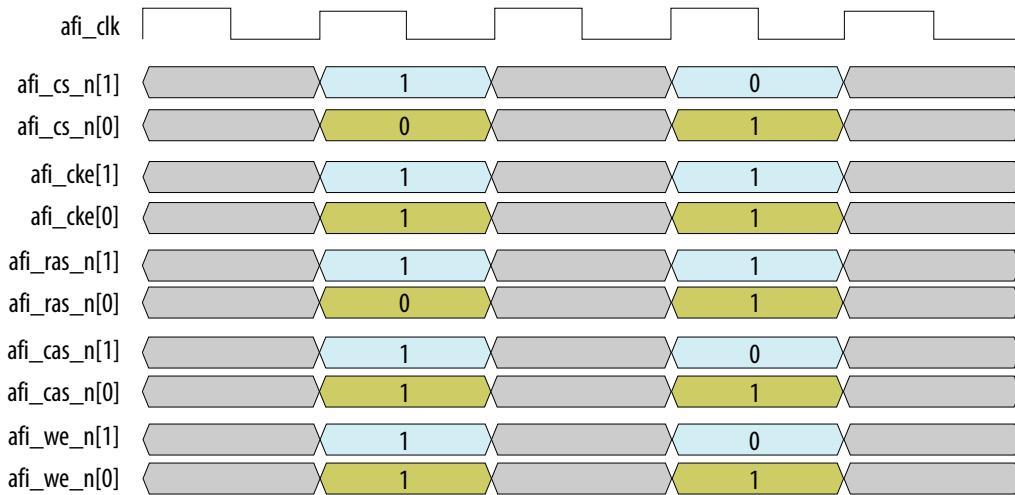
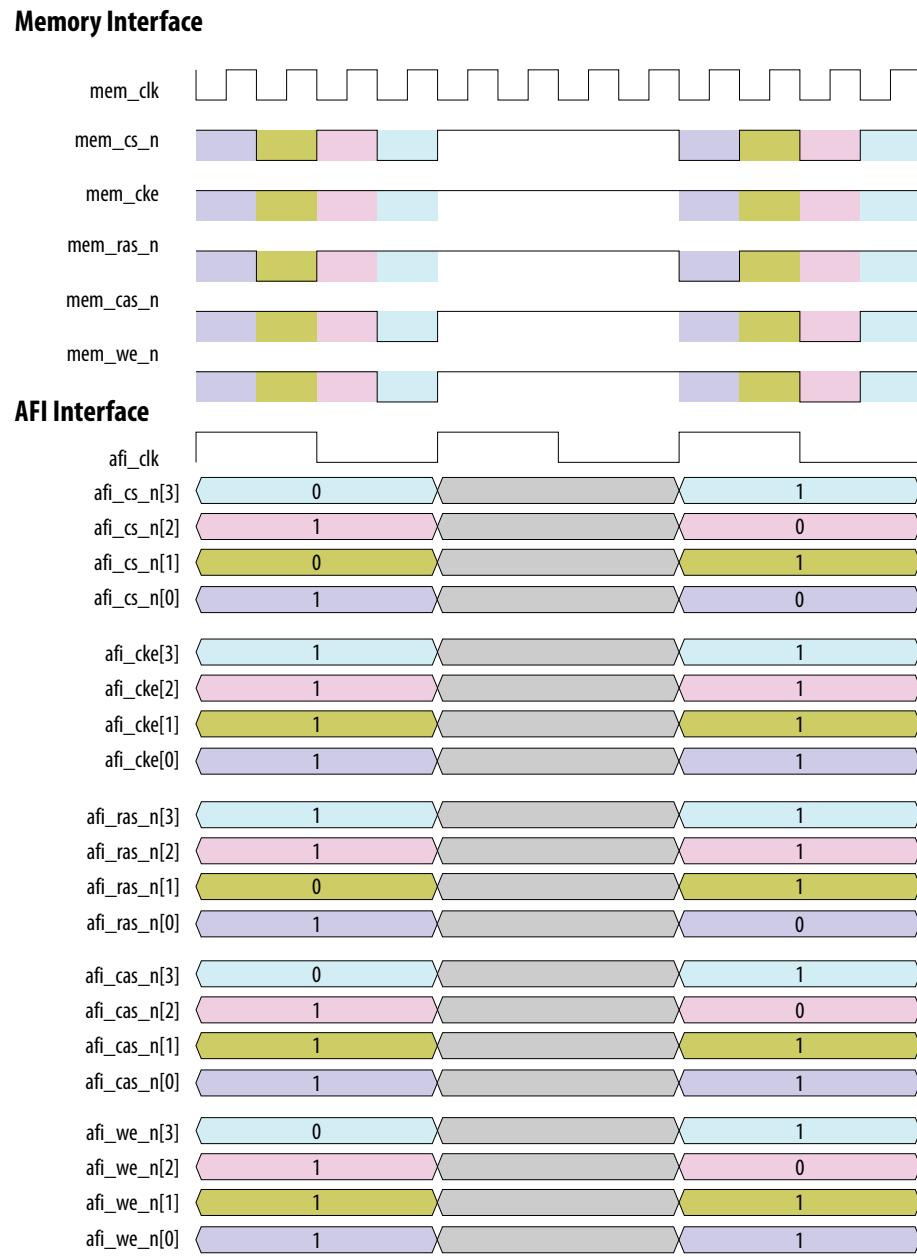


Figure 2-38: AFI Address and Command Quarter-Rate



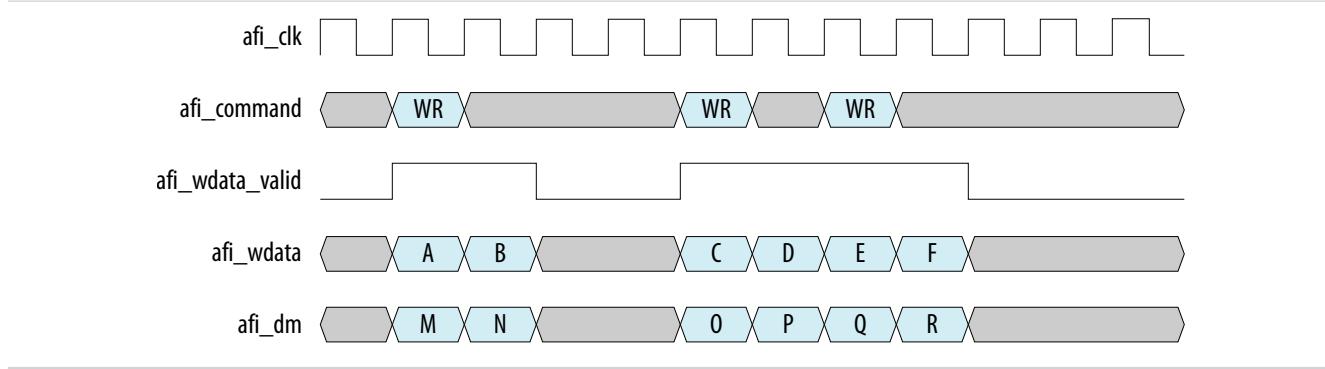
## AFI Write Sequence Timing Diagrams

The following timing diagrams illustrate the relationships between the write command and corresponding write data and write enable signals, in full, half, and quarter rate.

For half rate and quarter rate, when the `write` command is sent on the first memory clock in a PHY clock (for example, `afi_cs_n[0] = 0`), that access is called *aligned access*; otherwise it is called *unaligned access*. You may use either aligned or unaligned access, or you may use both, but you must ensure that the distance between the `write` command and the corresponding write data are constant on the AFI interface. For example, if a command is sent on the second memory clock in a PHY clock, the write data must also start at the second memory clock in a PHY clock.

### Write sequences with wlat=0

Figure 2-39: AFI Write Data Full-Rate, wlat=0



The following diagrams illustrate both aligned and unaligned access. The first three write commands are aligned accesses where they were issued on LSB of afi\_command. The fourth write command is unaligned access where it was issued on a different command slot. AFI signals must be shifted accordingly, based on the command slot.

Figure 2-40: AFI Write Data Half-Rate, wlat=0

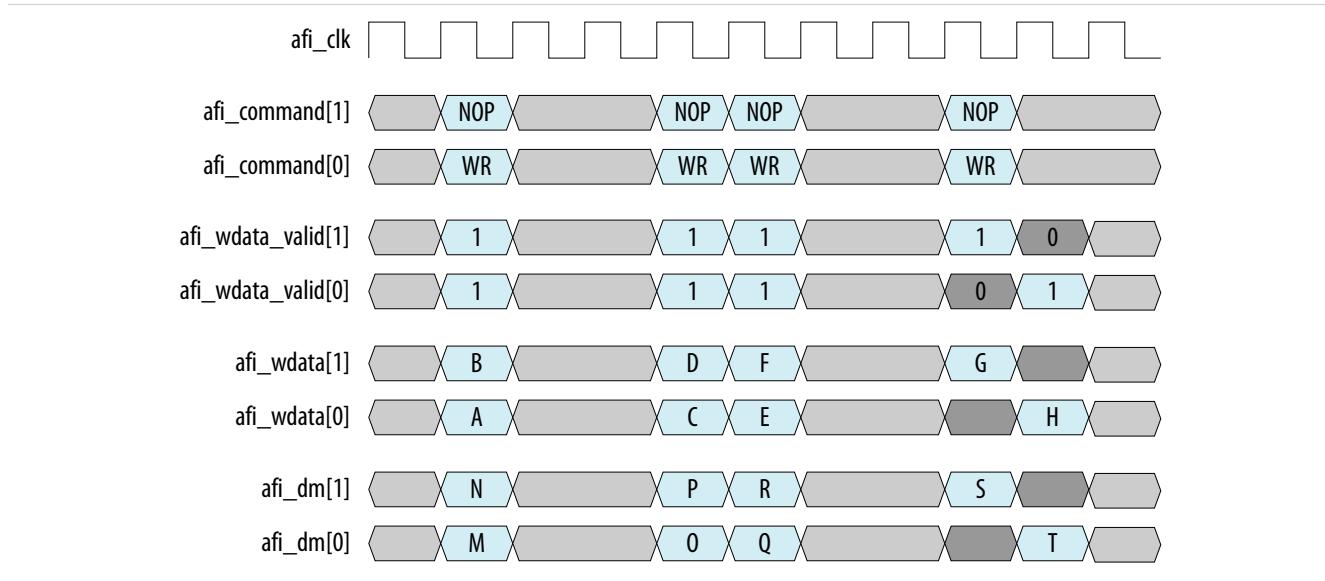
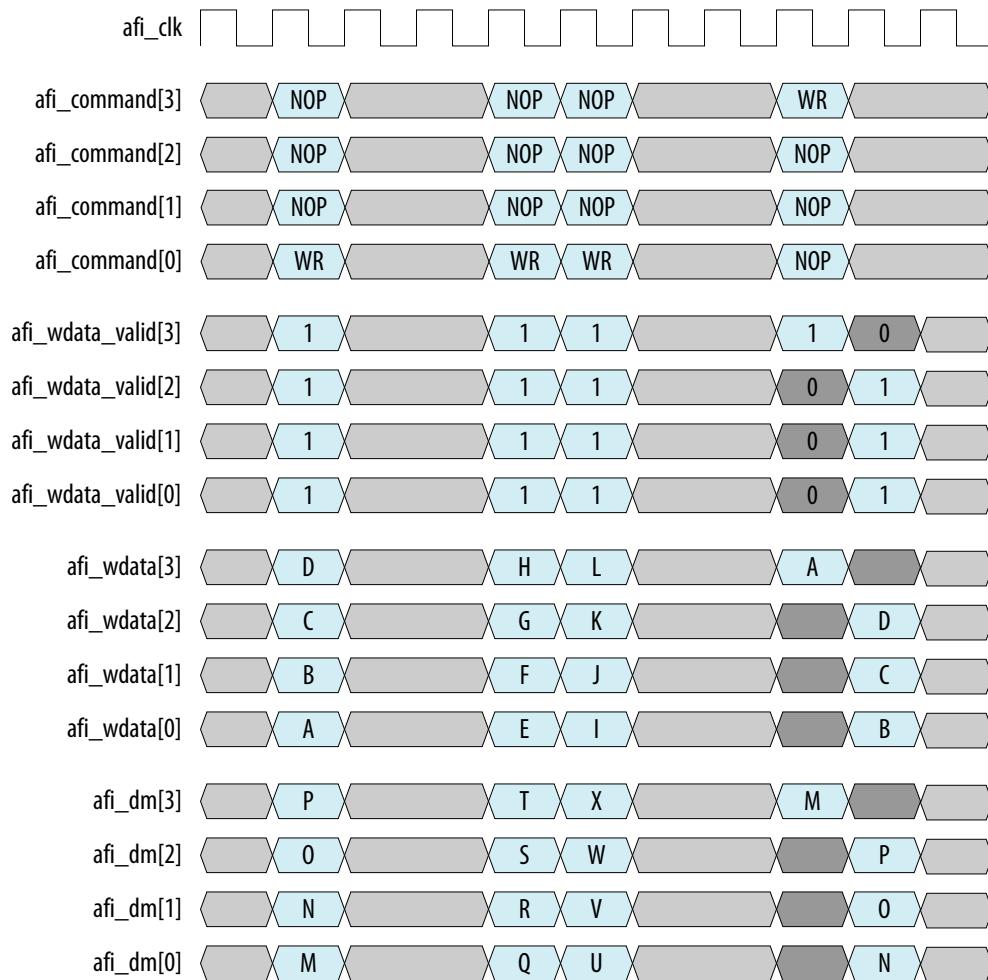


Figure 2-41: AFI Write Data Quarter-Rate, wlat=0



### Write sequences with wlat=non-zero

The **afi\_wlat** is a signal from the PHY. The controller must delay **afi\_dqs\_burst**, **afi\_wdata\_valid**, **afi\_wdata** and **afi\_dm** signals by a number of PHY clock cycles equal to **afi\_wlat**, which is a static value determined by calibration before the PHY asserts **cal\_success** to the controller. The following figures illustrate the cases when **wlat=1**. Note that **wlat** is in the number of PHY clocks and therefore **wlat=1** equals 1, 2, and 4 memory clocks delay, respectively, on full, half and quarter rate.

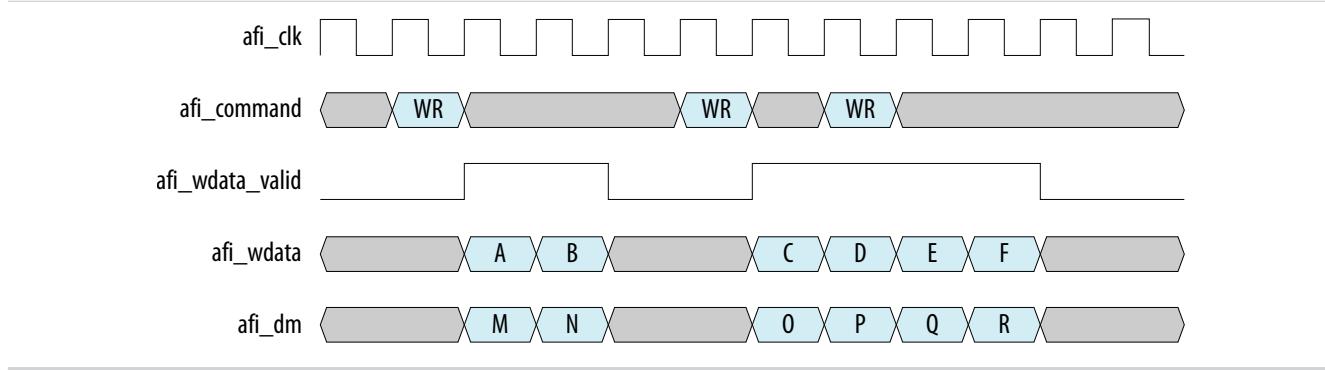
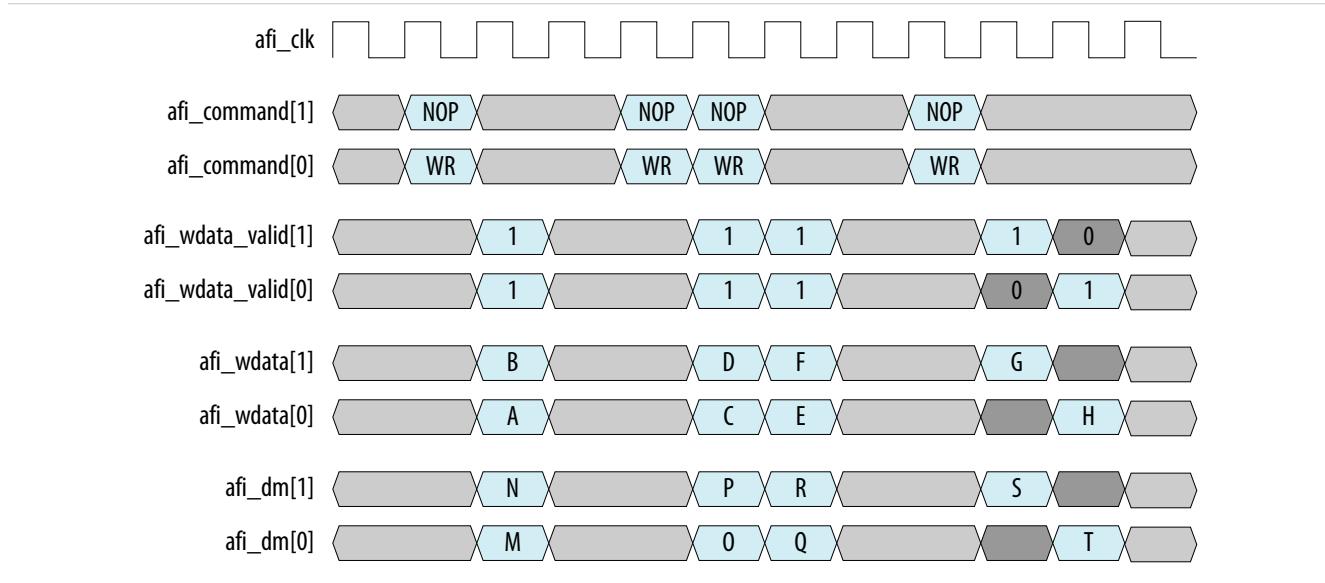
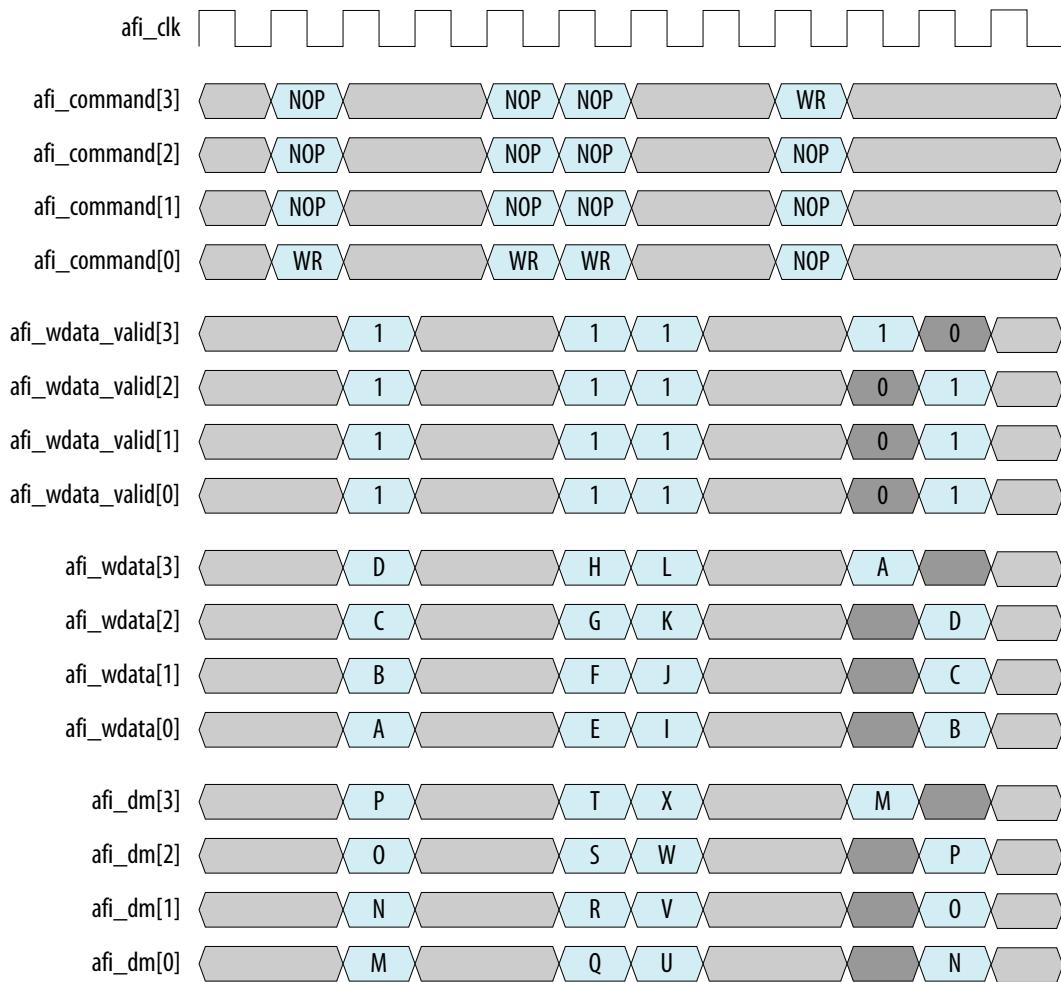
**Figure 2-42: AFI Write Data Full-Rate, wlat=1****Figure 2-43: AFI Write Data Half-Rate, wlat=1**

Figure 2-44: AFI Write Data Quarter-Rate, wlat=1



### DQS burst

The `afi_dqs_burst` signal must be asserted one or two complete memory clock cycles earlier to generate DQS preamble. DQS preamble is equal to one-half and one-quarter AFI clock cycles in half and quarter rate, respectively.

A DQS preamble of two is required in DDR4, when the write preamble is set to two clock cycles.

The following diagrams illustrate how `afi_dqs_burst` must be asserted in full, half, and quarter-rate configurations.

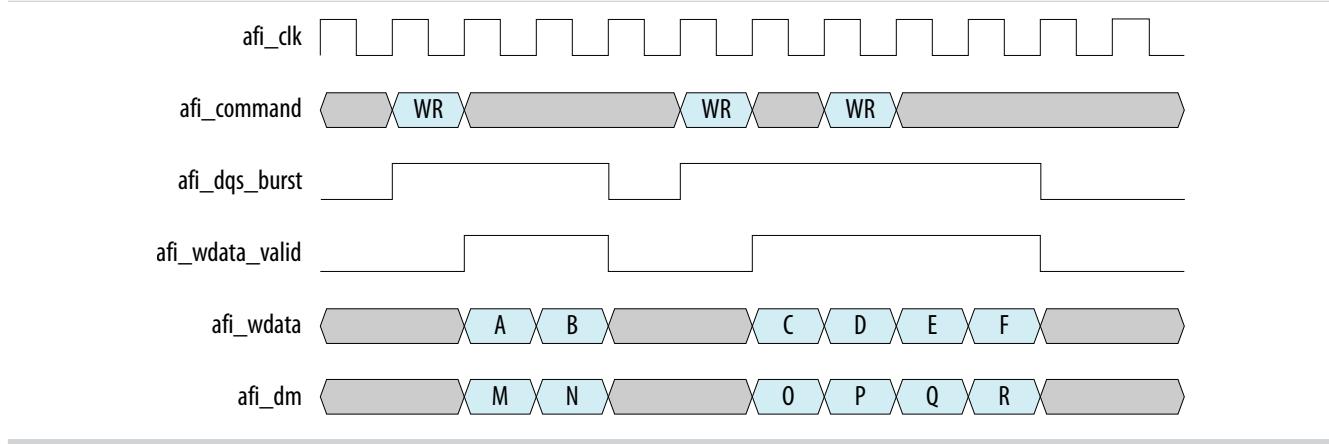
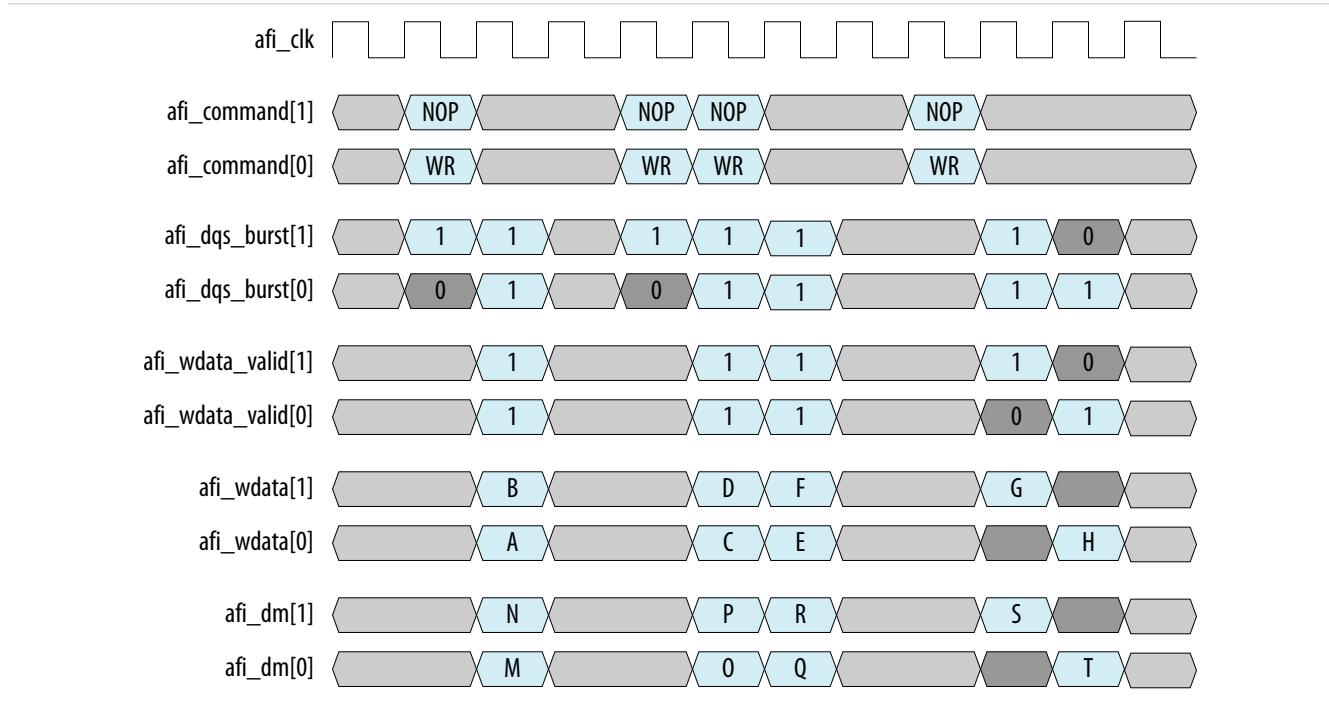
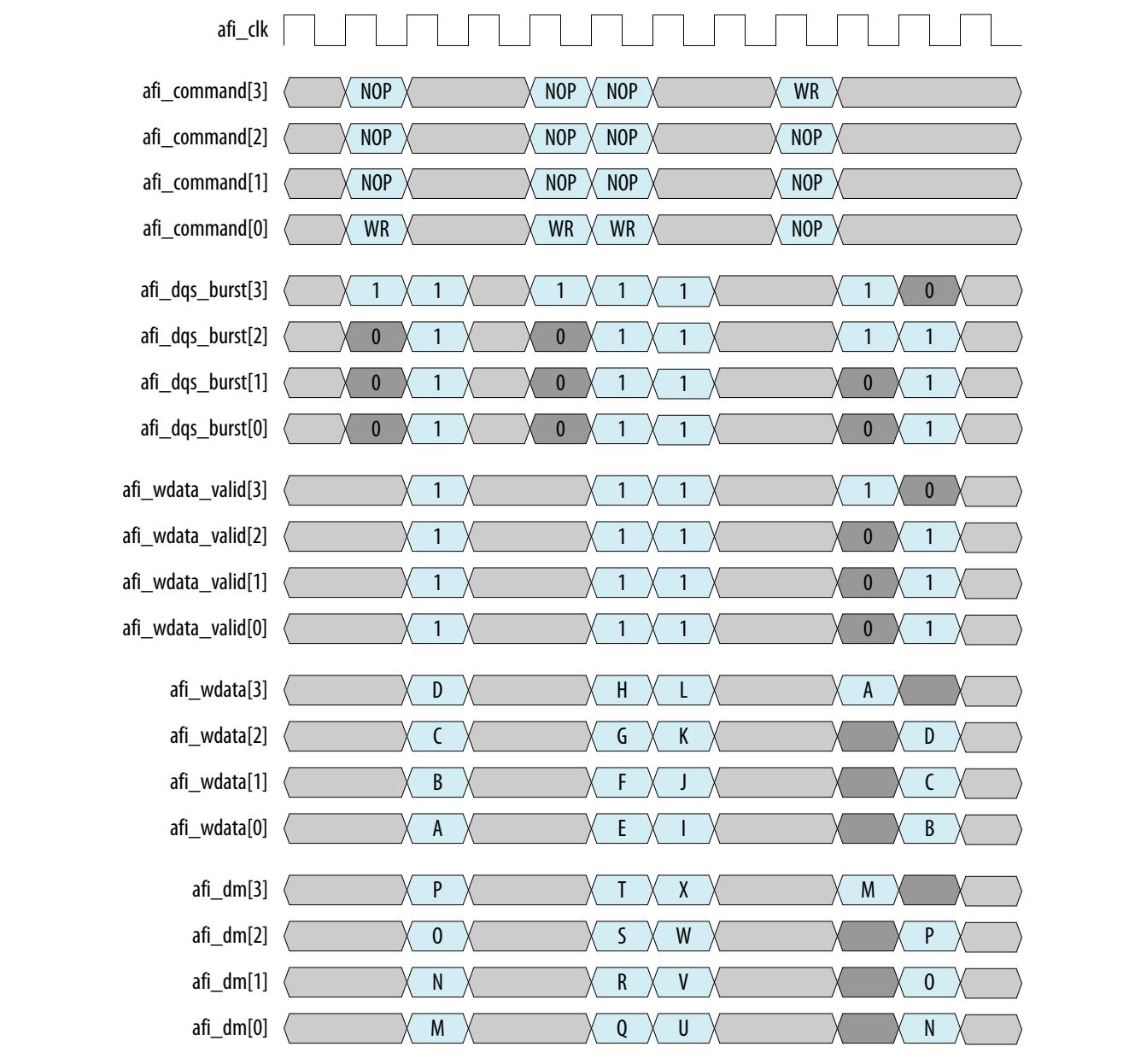
**Figure 2-45: AFI DQS Burst Full-Rate, wlat=1****Figure 2-46: AFI DQS Burst Half-Rate, wlat=1**

Figure 2-47: AFI DQS Burst Quarter-Rate, wlat=1



### Write data sequence with DBI (DDR4 and QDRIV only)

The DDR4 write DBI feature is supported in the PHY, and when it is enabled, the PHY sends and receives the DBI signal without any controller involvement. The sequence is identical to non-DBI scenarios on the AFI interface.

### Write data sequence with CRC (DDR4 only)

When the CRC feature of the PHY is enabled and used, the controller ensures at least one memory clock cycle between `write` commands, during which the PHY inserts the CRC data. Sending back-to-back `write` command would cause functional failure. The following figures show the legal sequences in CRC mode.

Entries marked as 0 and RESERVE must be observed by the controller; no information is allowed on those entries.

Figure 2-48: AFI Write Data with CRC Half-Rate, wlat=2

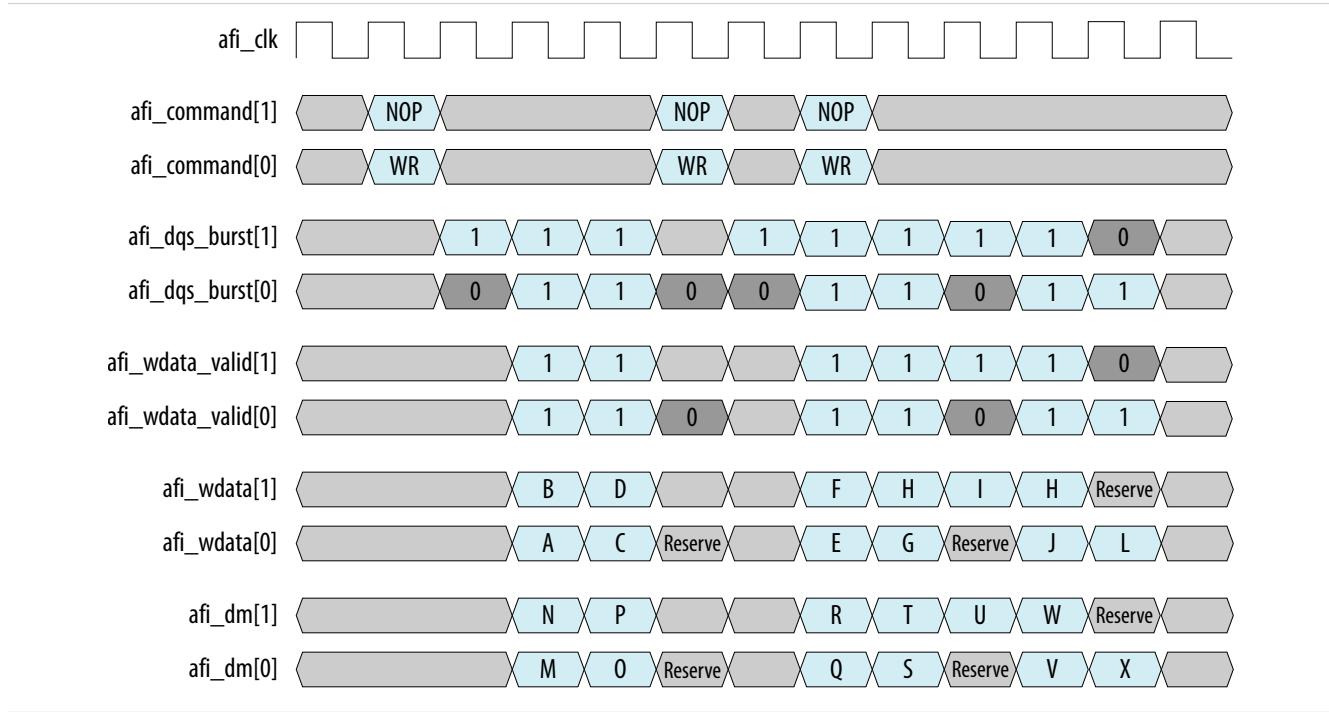
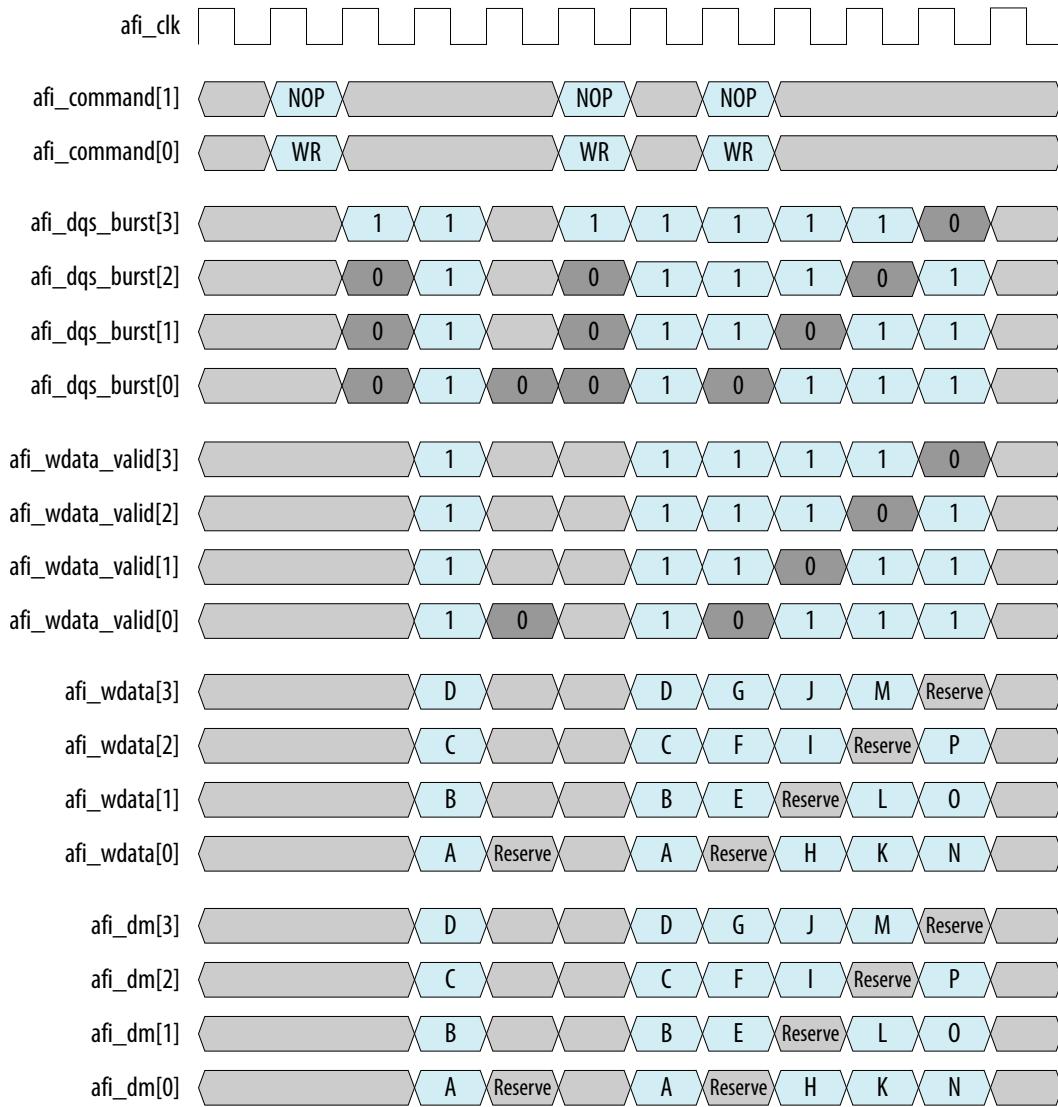


Figure 2-49: AFI Write Data with CRC Quarter-Rate, wlat=2

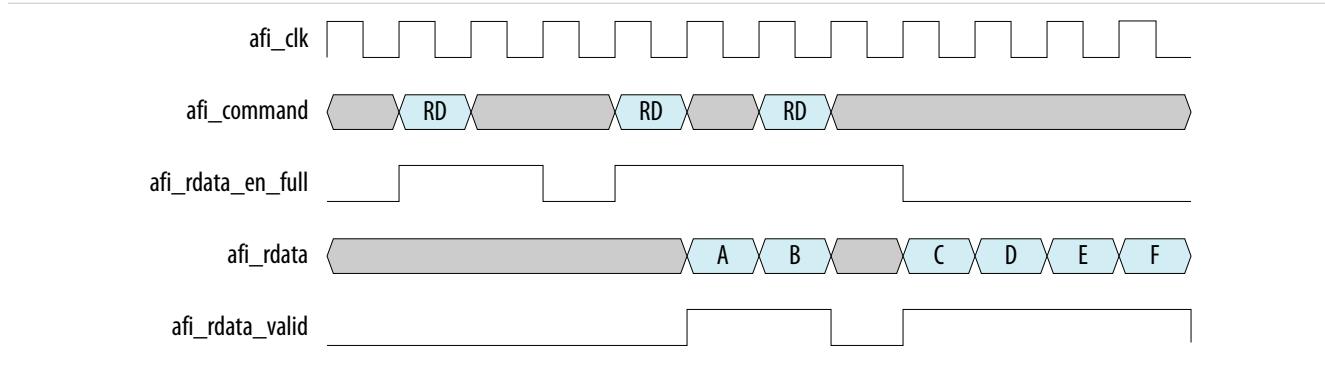


## AFI Read Sequence Timing Diagrams

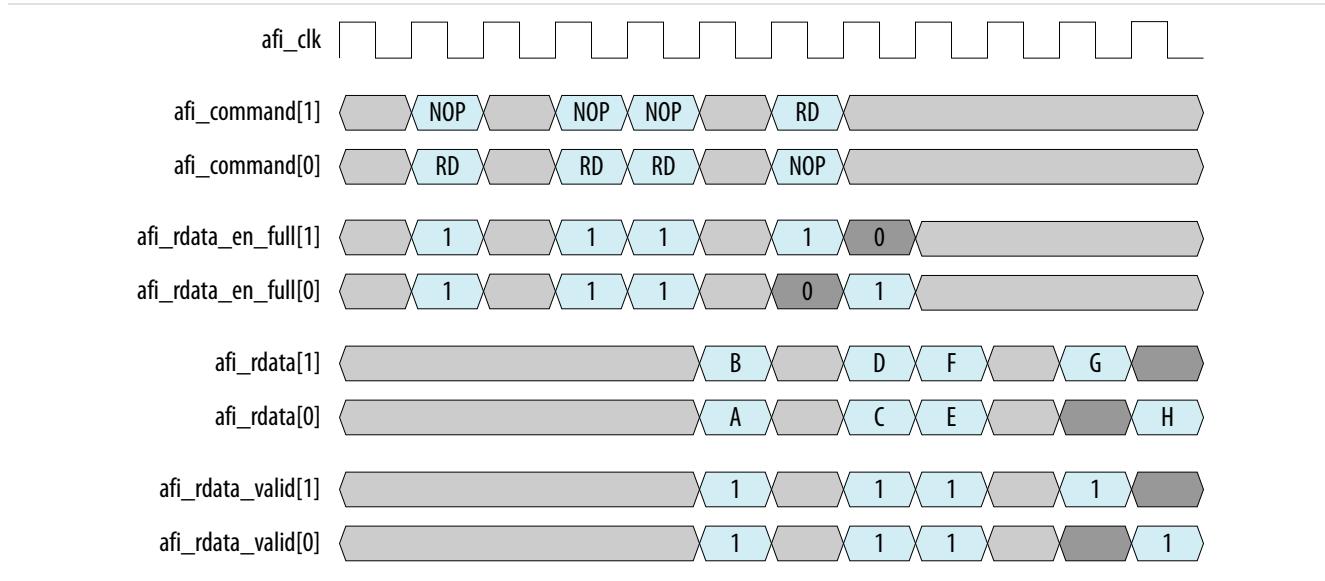
The following waveforms illustrate the AFI write data waveform in full, half, and quarter-rate, respectively.

The `afi_rdata_en_full` signal must be asserted for the entire read burst operation. The `afi_rdata_en` signal need only be asserted for the intended read data.

Aligned and unaligned access for read commands is similar to write commands; however, the `afi_rdata_en_full` signal must be sent on the same memory clock in a PHY clock as the read command. That is, if a read command is sent on the second memory clock in a PHY clock, `afi_rdata_en_full` must also be asserted, starting from the second memory clock in a PHY clock.

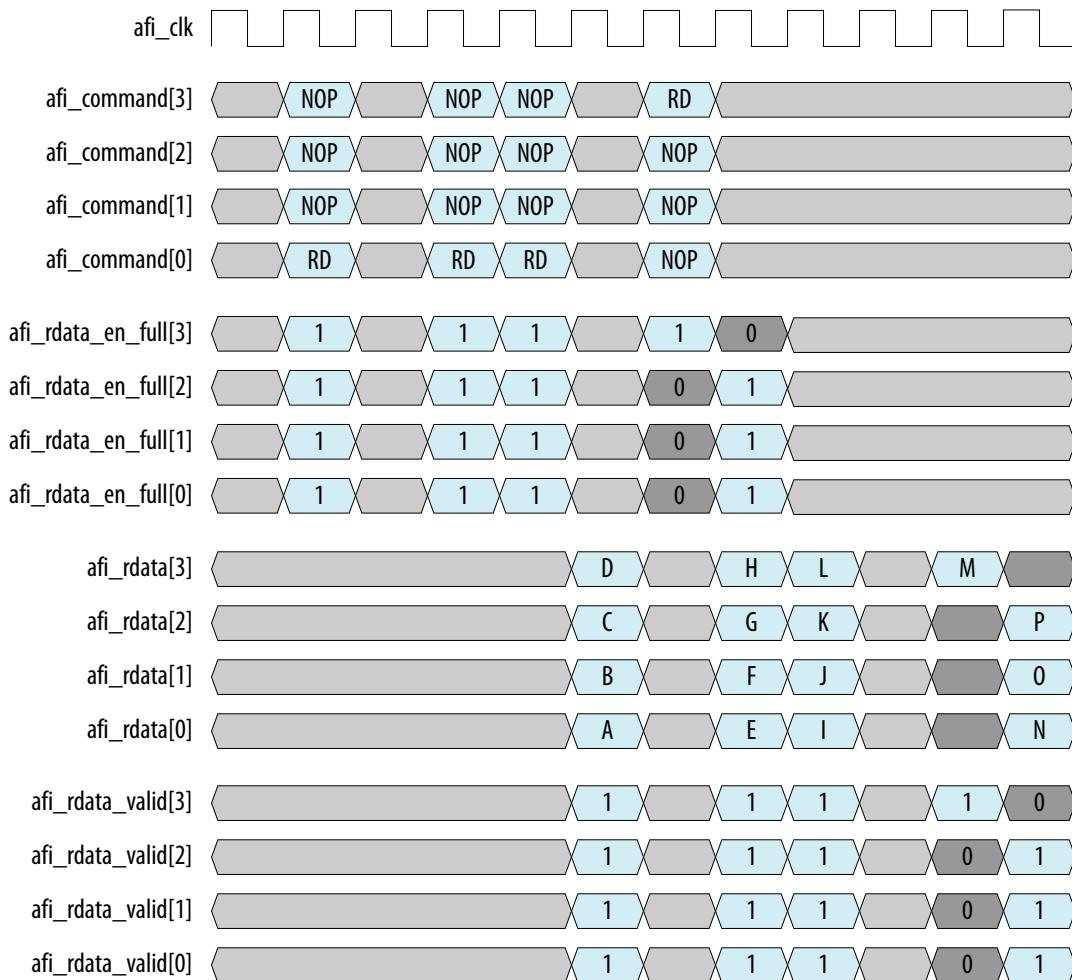
**Figure 2-50: AFI Read Data Full-Rate**

The following figure illustrates that the second and third reads require only the first and second half of data, respectively. The first three `read` commands are aligned accesses where they are issued on the LSB of `afi_command`. The fourth `read` command is unaligned access, where it is issued on a different command slot. AFI signals must be shifted accordingly, based on command slot.

**Figure 2-51: AFI Read Data Half-Rate**

In the following figure, the first three `read` commands are aligned accesses where they are issued on the LSB of `afi_command`. The fourth `read` command is unaligned access, where it is issued on a different command slot. AFI signals must be shifted accordingly, based on command slot.

Figure 2-52: AFI Read Data Quarter-Rate

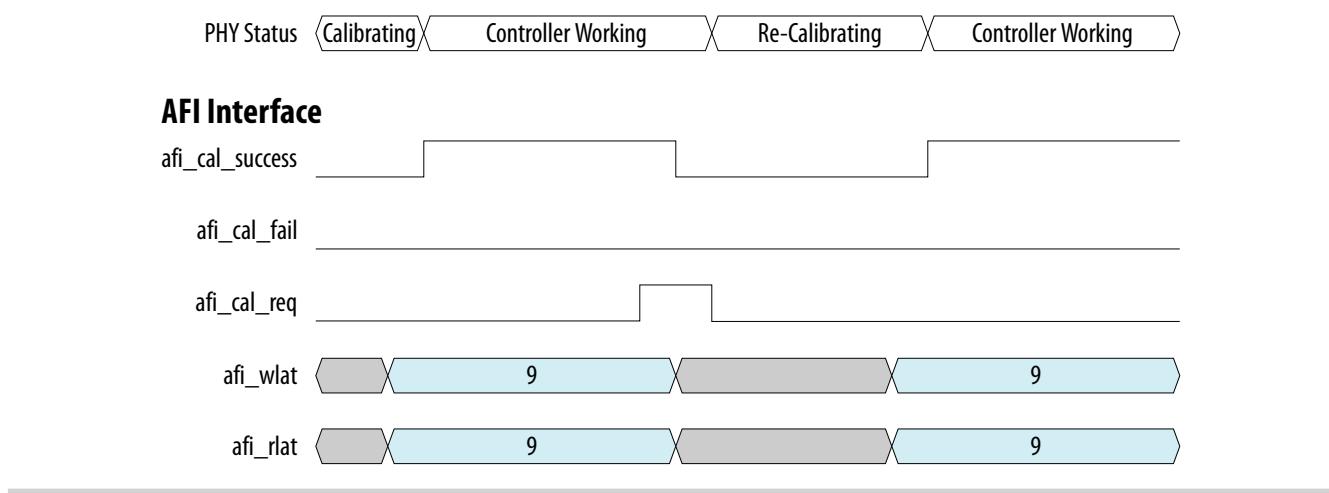


### AFI Calibration Status Timing Diagram

The controller interacts with the PHY during calibration at power-up and at recalibration.

At power-up, the PHY holds `afi_cal_success` and `afi_cal_fail` 0 until calibration is done, when it asserts `afi_cal_success`, indicating to controller that the PHY is ready to use and `afi_wlat` and `afi_rlat` signals have valid values.

At recalibration, the controller asserts `afi_cal_req`, which triggers the same sequence as at power-up, and forces recalibration of the PHY.

**Figure 2-53: Calibration**

## Resource Utilization

The following tables provide resource utilization information for external memory interfaces on Arria 10 devices.

### QDR-IV Resource Utilization in Arria 10 Devices

The following table shows typical resource usage of QDR-IV interfaces with soft controller for Arria 10 devices.

**Table 2-17: QDR-IV Resource Utilization in Arria 10 Devices**

Memory Width (Bits)	Combinational ALUTs	Dedicated Logic Registers	Block Memory Bits	M20Ks	Soft Controller
18	2123	4592	18432	8	1
36	2127	6023	36864	16	1
72	2114	8826	73728	32	1

## Arria 10 EMIF Latency

The following latency data applies to all memory protocols supported by the Arria 10 EMIF IP.

**Table 2-18: Latency in Full-Rate Memory Clock Cycles**

Rate <sup>1</sup>	Controller Address & Command	PHY Address & Command	Memory Read Latency <sup>2</sup>	PHY Read Data Return	Controller Read Data Return	Round Trip	Round Trip Without Memory
Half:Write	12	2	3-23	—	—	—	—
Half:Read	8	2	3-23	6	8	27-47	24
Quarter: Write	14	2	3-23	—	—	—	—
Quarter: Read	10	2	3-23	6	14	35-55	32
Half:Write (ECC)	14	2	3-23	—	—	—	—
Half:Read (ECC)	12	2	3-23	6	8	31-51	28
Quarter: Write (ECC)	14	2	3-23	—	—	—	—
Quarter: Read (ECC)	12	2	3-23	6	14	37-57	34

1. User interface rate; the controller always operates in half rate.
2. Minimum and maximum read latency range for DDR3, DDR4, and LPDDR3.

## Arria 10 EMIF Calibration Times

The time needed for calibration varies, depending on many factors including the interface width, the number of ranks, frequency, board layout, and difficulty of calibration.

The following table lists approximate typical calibration times for various protocols and configurations.

**Table 2-19: Arria 10 EMIF IP Approximate Calibration Times**

Protocol	Rank and Frequency	Typical Calibration Time
DDR3, x64 UDIMM, DQS x8, DM on	1 rank, 933 MHz	102 ms
	1 rank, 800 MHz	106 ms
	2 rank, 933 MHz	198 ms
	2 rank, 800 MHz	206 ms

Protocol	Rank and Frequency	Typical Calibration Time
DDR4, x64 UDIMM, DQS x8, DBI on	1 rank, 1067 MHz	314 ms
	1 rank, 800 MHz	353 ms
	2 rank 1067 MHz	625 ms
	2 rank 800 MHz	727 ms
RLDRAM 3, x36	1200 MHz	2808 ms
	1067 MHz	2825 ms
	1200 MHz, with DM	2818 ms
	1067 MHz, with DM	2833 ms
QDR II, x36, BWS on	333 MHz	616 ms
	633 MHz	833 ms
QDR-IV, x36, BWS on	1067 MHz	1563 ms
	1067 MHz, with DBI	1556 ms

## Integrating a Custom Controller with the Hard PHY

If you want to use your own custom memory controller, you must integrate the controller with the hard PHY to achieve a complete memory solution.

Observe the following general guidelines:

- When you configure your external memory interface IP, ensure that you select **Configuration > Hard PHY Only** on the **General** tab in the parameter editor.
- Consult the AFI 4.0 Specification, for detailed information on the AFI interface to the PHY.

## Memory Mapped Register (MMR) Tables

### Register Summary

Register	Address 32-bit Bus	Bits Register link
dbgcfg0	0	16
dbgcfg1	1	32
dbgcfg2	2	32
dbgcfg3	3	32
dbgcfg4	4	32
dbgcfg5	5	32
dbgcfg6	6	16

Register	Address 32-bit Bus	Bits Register link
sbcfg8	7	16
sbcfg9	8	16
reserve2	9	16
ctrlcfg0	10	32
ctrlcfg1	11	32
ctrlcfg2	12	32
ctrlcfg3	13	32
ctrlcfg4	14	32
ctrlcfg5	15	16
ctrlcfg6	16	16
ctrlcfg7	17	16
ctrlcfg8	18	8
ctrlcfg9	19	8
dramtiming0	20	24
dramodt0	21	32
dramodt1	22	24
sbcfg0	23	32
sbcfg1	24	32
sbcfg2	25	8
sbcfg3	26	24
sbcfg4	27	24
sbcfg5	28	8
sbcfg6	29	32
sbcfg7	30	8
caltiming0	31	32
caltiming1	32	32
caltiming2	33	32
caltiming3	34	32
caltiming4	35	32
caltiming5	36	24
caltiming6	37	32
caltiming7	38	32
caltiming8	39	32
caltiming9	40	8

Register	Address 32-bit Bus	Bits Register link
caltiming10	41	8
dramaddrw	42	24
sideband0	43	8
sideband1	44	8
sideband2	45	8
sideband3	46	8
sideband4	47	8
sideband5	48	8
sideband6	49	8
sideband7	50	8
sideband8	51	8
sideband9	52	8
sideband10	53	8
sideband11	54	8
sideband12	55	8
sideband13	56	32
sideband14	57	16
sideband15	58	8
dramsts	59	8
dbgdone	60	8
dbgsignals	61	32
dbgreset	62	8
dbgmatch	63	32
counter0mask	64	32
counter1mask	65	32
counter0match	66	32
counter1match	67	32
niosreserve0	68	16
niosreserve1	69	16
niosreserve2	70	16
ecc1	128	10
ecc2	129	22
ecc3	130	9

Register	Address 32-bit Bus	Bits Register link
ecc4	144	16

**Note:** Addresses are in decimal format.

## ctrlcfg0: Controller Configuration

address=10(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_mem_type	3	0	Selects memory type. Program this field with one of the following binary values, "0000" for DDR3 SDRAM, "0001" for DDR4 SDRAM, "0010" for LPDDR3 SDRAM and "0011" for RLDRAM3.	Read/Write
cfg_dimm_type	6	4	Selects dimm type. Program this field with one of the following binary values, "3	Read/Write
cfg_ac_pos	8	7	Specify C/A (command/address) pin position.	Read/Write
cfg_ctrl_burst_length	13	9	Configures burst length for control path. Legal values are valid for JEDEC allowed DRAM values for the DRAM selected in cfg_type. For DDR3, DDR4 and LPDDR3, this should be programmed with 8 (binary "01000"), for RLDRAM III it can be programmed with 2 or 4 or 8.	Read/Write
cfg_dbc0_burst_length	18	14	Configures burst length for DBC0. Legal values are valid for JEDEC allowed DRAM values for the DRAM selected in cfg_type. For DDR3, DDR4 and LPDDR3, this should be programmed with 8 (binary "01000"), for RLDRAM III it can be programmed with 2 or 4 or 8.	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_dbc1_burst_length	23	19	Configures burst length for DBC1. Legal values are valid for JEDEC allowed DRAM values for the DRAM selected in cfg_type. For DDR3, DDR4 and LPDDR3, this should be programmed with 8 (binary "01000"), for RLDRAM III it can be programmed with 2 or 4 or 8.	Read/Write
cfg_dbc2_burst_length	28	24	Configures burst length for DBC2. Legal values are valid for JEDEC allowed DRAM values for the DRAM selected in cfg_type. For DDR3, DDR4 and LPDDR3, this should be programmed with 8 (binary "01000"), for RLDRAM III it can be programmed with 2 or 4 or 8.	Read/Write

## ctrlcfg1: Controller Configuration

address=11(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_dbc3_burst_length	4	0	Configures burst length for DBC3. Legal values are valid for JEDEC allowed DRAM values for the DRAM selected in cfg_type. For DDR3, DDR4 and LPDDR3, this should be programmed with 8 (binary "01000"), for RLDRAM III it can be programmed with 2 or 4 or 8.	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_addr_order	6	5	Selects the order for address interleaving. Programming this field with different values gives different mappings between the AXI or Avalon-MM address and the SDRAM address. Program this field with the following binary values to select the ordering. "00" - chip, row, bank(BG, BA), column; "01" - chip, bank(BG, BA), row, column; "10" -row, chip, bank(BG, BA), column.	Read/Write
cfg_ctrl_enable_ecc	7	7	Enable the generation and checking of ECC.	Read/Write
cfg_dbc0_enable_ecc	8	8	Enable the generation and checking of ECC.	Read/Write
cfg_dbc1_enable_ecc	9	9	Enable the generation and checking of ECC.	Read/Write
cfg_dbc2_enable_ecc	10	10	Enable the generation and checking of ECC.	Read/Write
cfg_dbc3_enable_ecc	11	11	Enable the generation and checking of ECC.	Read/Write
cfg_reorder_data	12	12	This bit controls whether the controller can re-order operations to optimize SDRAM bandwidth. It should generally be set to a one.	Read/Write
cfg_ctrl_reorder_rdata	13	13	This bit controls whether the controller need to re-order the read return data.	Read/Write
cfg_dbc0_reorder_rdata	14	14	This bit controls whether the controller need to re-order the read return data.	Read/Write
cfg_dbc1_reorder_rdata	15	15	This bit controls whether the controller need to re-order the read return data.	Read/Write
cfg_dbc2_reorder_rdata	16	16	This bit controls whether the controller need to re-order the read return data.	Read/Write
cfg_dbc3_reorder_rdata	17	17	This bit controls whether the controller need to re-order the read return data.	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_reordered_read	18	18	This bit controls whether the controller can re-order read command to 1.	Read/Write
cfg_starve_limit	24	19	Specifies the number of DRAM burst transactions an individual transaction will allow to reorder ahead of it before its priority is raised in the memory controller.	Read/Write
cfg_dqstrk_en	25	25	Enables DQS tracking in the PHY.	Read/Write
cfg_ctrl_enable_dm	26	26	Set to a one to enable DRAM operation if DM pins are connected.	Read/Write
cfg_dbc0_enable_dm	27	27	Set to a one to enable DRAM operation if DM pins are connected.	Read/Write
cfg_dbc1_enable_dm	28	28	Set to a one to enable DRAM operation if DM pins are connected.	Read/Write
cfg_dbc2_enable_dm	29	29	Set to a one to enable DRAM operation if DM pins are connected.	Read/Write
cfg_dbc3_enable_dm	30	30	Set to a one to enable DRAM operation if DM pins are connected.	Read/Write

## ctrlcfg2: Controller Configuration

address=12(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_ctrl_output_regd	0	0	Set to one to register the HMC command output. Set to 0 to disable it.	Read/Write
cfg_dbc0_output_regd	1	1	Set to one to register the HMC command output. Set to 0 to disable it.	Read/Write
cfg_dbc1_output_regd	2	2	Set to one to register the HMC command output. Set to 0 to disable it.	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_dbc2_output_regd	3	3	Set to one to register the HMC command output. Set to 0 to disable it.	Read/Write
cfg_dbc3_output_regd	4	4	Set to one to register the HMC command output. Set to 0 to disable it.	Read/Write
cfg_ctrl2dbc_switch0	6	5	Select of the MUX ctrl2dbc_switch0. 2	Read/Write
cfg_ctrl2dbc_switch1	8	7	Select of the MUX ctrl2dbc_switch1. 2	Read/Write
cfg_dbc0_ctrl_sel	9	9	DBC0 - control path select. 1	Read/Write
cfg_dbc1_ctrl_sel	10	10	DBC1 - control path select. 1	Read/Write
cfg_dbc2_ctrl_sel	11	11	DBC2 - control path select. 1	Read/Write
cfg_dbc3_ctrl_sel	12	12	DBC3 - control path select. 1	Read/Write
cfg_dbc2ctrl_sel	14	13	Specifies which DBC is driven by the local control path. 2	Read/Write
cfg_dbc0_pipe_lat	17	15	Specifies in number of controller clock cycles the latency of pipelining the signals from control path to DBC0	Read/Write
cfg_dbc1_pipe_lat	20	18	Specifies in number of controller clock cycles the latency of pipelining the signals from control path to DBC1	Read/Write
cfg_dbc2_pipe_lat	23	21	Specifies in number of controller clock cycles the latency of pipelining the signals from control path to DBC2	Read/Write
cfg_dbc3_pipe_lat	26	24	Specifies in number of controller clock cycles the latency of pipelining the signals from control path to DBC3	Read/Write

## ctrlcfg3: Controller Configuration

**address=13(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_ctrl_cmd_rate	2	0	3	Read/Write
cfg_dbc0_cmd_rate	5	3	3	Read/Write
cfg_dbc1_cmd_rate	8	6	3	Read/Write
cfg_dbc2_cmd_rate	11	9	3	Read/Write
cfg_dbc3_cmd_rate	14	12	3	Read/Write
cfg_ctrl_in_protocol	15	15	1	Read/Write
cfg_dbc0_in_protocol	16	16	1	Read/Write
cfg_dbc1_in_protocol	17	17	1	Read/Write
cfg_dbc2_in_protocol	18	18	1	Read/Write
cfg_dbc3_in_protocol	19	19	1	Read/Write
cfg_ctrl_dualport_en	20	20	Enable the second command port for RLDRAM3 only (BL=2 or 4)	Read/Write
cfg_dbc0_dualport_en	21	21	Enable the second data port for RLDRAM3 only (BL=2 or 4)	Read/Write
cfg_dbc1_dualport_en	22	22	Enable the second data port for RLDRAM3 only (BL=2 or 4)	Read/Write
cfg_dbc2_dualport_en	23	23	Enable the second data port for RLDRAM3 only (BL=2 or 4)	Read/Write
cfg_dbc3_dualport_en	24	24	Enable the second data port for RLDRAM3 only (BL=2 or 4)	Read/Write
cfg_arbiter_type	25	25	Indicates controller arbiter operating mode. Set this to: - 1	Read/Write
cfg_open_page_en	26	26	Set to 1 to enable the open page policy when command reordering is disabled (cfg_cmd_reorder = 0). This bit does not matter when cfg_cmd_reorder is 1.	Read/Write
cfg_rld3_multibank_mode	30	28	Multibank setting, specific for RLDRAM3. Set this to: - 3	Read/Write

**Note:** DDR4 gear down mode is not supported.

## ctrlcfg4: Controller Configuration

**address=14(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_tile_id	4	0	Tile ID.	Read/Write
cfg_pingpong_mode	6	5	Ping Pong mode: 2	Read/Write
cfg_ctrl_slot_rotate_en	9	7	Cmd slot rotate enable: bit[0] controls write, 1	Read/Write
cfg_dbc0_slot_rotate_en	12	10	DBC0 slot rotate enable: bit[0] controls write, 1	Read/Write
cfg_dbc1_slot_rotate_en	15	13	DBC1 slot rotate enable: bit[0] controls write, 1	Read/Write
cfg_dbc2_slot_rotate_en	18	16	DBC2 slot rotate enable: bit[0] controls write, 1	Read/Write
cfg_dbc3_slot_rotate_en	21	19	DBC3 slot rotate enable: bit[0] controls write, 1	Read/Write
cfg_ctrl_slot_offset	23	22	Enables afi information to be offset by numbers of FR cycles. Affected afi signal is afi_rdata_en, afi_rdata_en_full, afi_wdata_valid, afi_dqs_burst, afi_mrnk_write and afi_mrnk_read. Set this to: - 2	Read/Write
cfg_dbc0_slot_offset	25	24	Enables afi information to be offset by numbers of FR cycles. Affected afi signal is afi_rdata_en, afi_rdata_en_full, afi_wdata_valid, afi_dqs_burst, afi_mrnk_write and afi_mrnk_read. Set this to: - 2	Read/Write
cfg_dbc1_slot_offset	27	26	Enables afi information to be offset by numbers of FR cycles. Affected afi signal is afi_rdata_en, afi_rdata_en_full, afi_wdata_valid, afi_dqs_burst, afi_mrnk_write and afi_mrnk_read. Set this to: - 2	Read/Write
cfg_dbc2_slot_offset	29	28	Enables afi information to be offset by numbers of FR cycles. Affected afi signal is afi_rdata_en, afi_rdata_en_full, afi_wdata_valid, afi_dqs_burst, afi_mrnk_write and afi_mrnk_read. Set this to: - 2	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_dbc3_slot_offset	31	30	Enables afi information to be offset by numbers of FR cycles. Affected afi signal is afi_rdata_en, afi_rdata_en_full, afi_wdata_valid, afi_dqs_burst, afi_mrnk_write and afi_mrnk_read. Set this to: - 2	Read/Write

## ctrlcfg5: Controller Configuration

address=15(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_col_cmd_slot	3	0	Specify the col cmd slot. One hot encoding.	Read/Write
cfg_row_cmd_slot	7	4	Specify the row cmd slot. One hot encoding.	Read/Write
cfg_ctrl_rc_en	8	8	Set to 1 to enable the rate conversion. It converts QR input from core to HR inside HMC.	Read/Write
cfg_dbc0_rc_en	9	9	Set to 1 to enable the rate conversion. It converts QR input from core to HR inside HMC.	Read/Write
cfg_dbc1_rc_en	10	10	Set to 1 to enable the rate conversion. It converts QR input from core to HR inside HMC.	Read/Write
cfg_dbc2_rc_en	11	11	Set to 1 to enable the rate conversion. It converts QR input from core to HR inside HMC.	Read/Write
cfg_dbc3_rc_en	12	12	Set to 1 to enable the rate conversion. It converts QR input from core to HR inside HMC.	Read/Write

## ctrlcfg6: Controller Configuration

**address=16(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_cs_chip	15	0	Chip select mapping scheme. Mapping separated into 4 sections: [CS3][CS2][CS1] [CS0] Each section consists of 4 bits to indicate which CS_n signal should be active when command goes to current CS. Eg: if we set to 16.	Read/Write

## ctrlcfg7: Controller Configuration

**address=17(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_clkgating_en	0	0	Set to 1 to enable the clock gating. The clock is shut off for the whole HMC.	Read/Write
cfg_rb_reserved_entry	7	1	Specify how many entries are reserved in read buffer before almost full is asserted.	Read/Write
cfg_wb_reserved_entry	14	8	Specify how many entries are reserved in write buffer before almost full is asserted.	Read/Write

## ctrlcfg8: Controller Configuration

**address=18(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_3ds_en	0	0	Setting to 1 to enable #DS support for DDR4.	Read/Write
cfg_ck_inv	1	1	Use to program CK polarity. 1	Read/Write
cfg_addr_mplx_en	2	2	Setting to 1 enables RLD3 address multiplex mode.	Read/Write

## ctrlcfg9: Controller Configuration

**address=19(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_dfx_bypass_en	0	0	Used for dft and timing characterization only. 1	Read/Write

## dramtiming0: Timing Parameters

**address=20(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_tcl	6	0	Memory read latency.	Read/Write
cfg_power_saving_exit_cycles	12	7	The minimum number of cycles to stay in a low power state. This applies to both power down and self-refresh and should be set to the greater of tPD and tCKESR.	Read/Write
cfg_mem_clk_disable_entry_cycles	18	13	Set to a the number of clocks after the execution of an self-refresh to stop the clock. This register is generally set based on PHY design latency and should generally not be changed.	Read/Write

## dramodt0: On-Die Termination Parameters

**address=21(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_write_odt_chip	15	0	ODT scheme setting for write command. Setting separated into 4 sections: [CS3][CS2] [CS1][CS0] Each section consists of 4 bits to indicate which chip should ODT be asserted when write occurs on current CS. Eg: if we set to 16.	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_read_odt_chip	31	16	ODT scheme setting for read command. Setting separated into 4 sections: [CS3][CS2] [CS1][CS0] Each section consists of 4 bits to indicate which chip should ODT be asserted when write occurs on current CS. Eg: if we set to 16.	Read/Write

## dramodt1: On-Die Termination Parameters

address=22(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_wr_odt_on	5	0	Indicates number of memory clock cycle gap between write command and ODT signal rising edge.	Read/Write
cfg_rd_odt_on	11	6	Indicates number of memory clock cycle gap between read command and ODT signal rising edge.	Read/Write
cfg_wr_odt_period	17	12	Indicates number of memory clock cycle write ODT signal should stay asserted after rising edge.	Read/Write
cfg_rd_odt_period	23	18	Indicates number of memory clock cycle read ODT signal should stay asserted after rising edge.	Read/Write

## sbcfg0: Sideband Configuration

address=23(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_rld3_refresh_seq0	15	0	Banks to Refresh for RLD3 in sequence 0. Must not be more than 4 banks.	Read/Write
cfg_rld3_refresh_seq1	31	16	Banks to Refresh for RLD3 in sequence 1. Must not be more than 4 banks.	Read/Write

## sbcfg1: Sideband Configuration

address=24(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_rld3_refresh_seq2	15	0	Banks to Refresh for RLD3 in sequence 2. Must not be more than 4 banks.	Read/Write
cfg_rld3_refresh_seq3	31	16	Banks to Refresh for RLD3 in sequence 3. Must not be more than 4 banks.	Read/Write

## sbcfg2: Sideband Configuration

address=25(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_srf_zqcal_disable	0	0	Set to 1 to disable ZQ Calibration after self refresh.	Read/Write
cfg_mps_zqcal_disable	1	1	Set to 1 to disable ZQ Calibration after Maximum Power Saving exit.	Read/Write
cfg_mps_dqstrk_disable	2	2	Set to 1 to disable DQS Tracking after Maximum Power Saving exit.	Read/Write
cfg_sb_cg_disable	3	3	Set to 1 to disable mem_ck gating during self refresh and deep power down. Clock gating is not supported when the Ping Pong PHY feature is enabled. Do not enable clock gating for Ping Pong PHY interfaces.	Read/Write
cfg_user_rfsh_en	4	4	Setting to 1 to enable user refresh.	Read/Write
cfg_srf_autoexit_en	5	5	Setting to 1 to enable controller to exit Self Refresh when new command is detected.	Read/Write
cfg_srf_entry_exit_block	7	6	Blocking arbiter from issuing cmds for the 4 cases, 2	Read/Write

## sbcfg3: Sideband Configuration

**address=26(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_sb_ddr4_mr3	19	0	This register stores the DDR4 MR3 Content.	Read/Write

## sbcfg4: Sideband Configuration

**address=27(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_sb_ddr4_mr4	19	0	This register stores the DDR4 MR4 Content.	Read/Write

## sbcfg5: Sideband Configuration

**address=28(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_short_dqstrk_ctrl_en	0	0	Set to 1 to enable controller controlled DQS short tracking, Set to 0 to enable sequencer controlled DQS short tracking.	Read/Write
cfg_period_dqstrk_ctrl_en	1	1	Set to 1 to enable controller to issue periodic DQS tracking.	Read/Write

## sbcfg6: Sideband Configuration

**address=29(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_period_dqstrk_interval	15	0	Interval between two controller controlled periodic DQS tracking.	Read/Write
cfg_t_param_dqstrk_to_valid_last	23	16	DQS Tracking Rd to Valid timing for the last Rank.	Read/Write
cfg_t_param_dqstrk_to_valid	31	24	DQS Tracking Rd to Valid timing for Ranks other than the Last.	Read/Write

## sbcfg7: Sideband Configuration

**address=30(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_rfsh_warn_threshold	6	0	Threshold to warn a refresh is needed within the number of controller clock cycles specified by the threshold.	Read/Write

## sbcfg8: Sideband Configuration

**address=7(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_reserve	15	1	General purpose reserved register.	Read/Write
cfg_ddr4_mps_addrmirror	0	0	When asserted, indicates DDR4 Address Mirroring is enabled for MPS.	Read/Write

## sbcfg9: Sideband Configuration

**address=8(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_sb_ddr4_mr5	15	0	DDR4 Mode Register 5.	Read/Write

## caltiming0: Command/Address/Latency Parameters

**address=31(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_t_param_act_to_rdwr	5	0	Activate to Read/write command timing.	Read/Write
cfg_t_param_act_to_pch	11	6	Active to precharge.	Read/Write
cfg_t_param_act_to_act	17	12	Active to activate timing on same bank.	Read/Write
cfg_t_param_act_to_act_diff_bank	23	18	Active to activate timing on different banks, for DDR4 same bank group.	Read/Write
cfg_t_param_act_to_act_diff_bg	29	24	Active to activate timing on different bank groups, DDR4 only.	Read/Write

## caltiming1: Command/Address/Latency Parameters

address=32(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_rd_to_rd	5	0	Read to read command timing on same bank.	Read/Write
cfg_t_param_rd_to_rd_diff_chip	11	6	Read to read command timing on different chips.	Read/Write
cfg_t_param_rd_to_rd_diff_bg	17	12	Read to read command timing on different chips.	Read/Write
cfg_t_param_rd_to_wr	23	18	Write to read command timing on same bank.	Read/Write
cfg_t_param_rd_to_wr_diff_chip	29	24	Read to write command timing on different chips	Read/Write

## caltiming2: Command/Address/Latency Parameters

address=33(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_rd_to_wr_diff_bg	5	0	Read to write command timing on different bank groups.	Read/Write
cfg_t_param_rd_to_pch	11	6	Read to precharge command timing.	Read/Write
cfg_t_param_rd_ap_to_valid	17	12	Read command with autoprecharge to data valid timing.	Read/Write
cfg_t_param_wr_to_wr	23	18	Write to write command timing on same bank.	Read/Write
cfg_t_param_wr_to_wr_diff_chip	29	24	Write to write command timing on different chips.	Read/Write

## caltiming3: Command/Address/Latency Parameters

**address=34(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_t_param_wr_to_wr_diff_bg	5	0	Write to write command timing on different bank groups.	Read/Write
cfg_t_param_wr_to_rd	11	6	Write to read command timing.	Read/Write
cfg_t_param_wr_to_rd_diff_chip	17	12	Write to read command timing on different chips.	Read/Write
cfg_t_param_wr_to_rd_diff_bg	23	18	Write to read command timing on different bank groups.	Read/Write
cfg_t_param_wr_to_pch	29	24	Write to precharge command timing.	Read/Write

**caltiming4: Command/Address/Latency Parameters****address=35(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_t_param_wr_ap_to_valid	5	0	Write with autoprecharge to valid command timing.	Read/Write
cfg_t_param_pch_to_valid	11	6	Precharge to valid command timing.	Read/Write
cfg_t_param_pch_all_to_valid	17	12	Precharge all to banks being ready for bank activation command.	Read/Write
cfg_t_param_arf_to_valid	25	18	Auto Refresh to valid DRAM command window.	Read/Write
cfg_t_param_pdn_to_valid	31	26	Power down to valid bank command window.	Read/Write

**caltiming5: Command/Address/Latency Parameters****address=36(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_t_param_srf_to_valid	9	0	Self-refresh to valid bank command window.	Read/Write
cfg_t_param_srf_to_zq_cal	19	10	Self refresh to ZQ calibration window.	Read/Write

## caltiming6: Command/Address/Latency Parameters

address=37(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_arf_period	12	0	Auto-refresh period.	Read/Write
cfg_t_param_pdn_period	28	13	Clock power down recovery period.	Read/Write

## caltiming7: Command/Address/Latency Parameters

address=38(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_zqcl_to_valid	8	0	Long ZQ calibration to valid.	Read/Write
cfg_t_param_zqcs_to_valid	15	9	Short ZQ calibration to valid.	Read/Write
cfg_t_param_mrs_to_valid	19	16	Mode Register Setting to valid.	Read/Write
cfg_t_param_mps_to_valid	29	20	Timing parameter for Maximum Power Saving to any valid command. tXMP	Read/Write

## caltiming8: Command/Address/Latency Parameters

address=39(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_mrr_to_valid	3	0	Timing parameter for Mode Register Read to any valid command.	Read/Write
cfg_t_param_mpr_to_valid	8	4	Timing parameter for Multi Purpose Register Read to any valid command.	Read/Write
cfg_t_param_mps_exit_cs_to_cke	12	9	Timing parameter for exit Maximum Power Saving. Timing requirement for CS assertion vs CKE de-assertion. tMPX_S	Read/Write
cfg_t_param_mps_exit_cke_to_cs	16	13	Timing parameter for exit Maximum Power Saving. Timing requirement for CKE de-assertion vs CS de-assertion. tMPX_LH	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_t_param_rld3_multibank_ref_delay	19	17	RLD3 Refresh to Refresh Delay for all sequences.	Read/Write
cfg_t_param_mmr_cmd_to_valid	27	20	MMR cmd to valid delay.	Read/Write

## caltiming9: Command/Address/Latency Parameters

address=40(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_4_act_to_act	7	0	The four-activate window timing parameter.	Read/Write

## caltiming10: Command/Address/Latency Parameters

address=41(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_16_act_to_act	7	0	The 16-activate window timing parameter (RLD3).	Read/Write

## dramaddrw: Row/Column/Bank Address Width Configuration

address=42(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_col_addr_width	4	0	The number of column address bits for the memory devices in your memory interface.	Read/Write
cfg_row_addr_width	9	5	The number of row address bits for the memory devices in your memory interface.	Read/Write
cfg_bank_addr_width	13	10	The number of bank address bits for the memory devices in your memory interface.	Read/Write
cfg_bank_group_addr_width	15	14	The number of bank group address bits for the memory devices in your memory interface.	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_cs_addr_width	18	16	The number of chip select address bits for the memory devices in your memory interface.	Read/Write

## sideband0: Sideband

address=43(32 bit)

Field	Bit High	Bit Low	Description	Access
mr_cmd_trigger	0	0	Write to 1 to trigger the execution of the mode register command.	Read/Write

## sideband1: Sideband

address=44(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_refresh_req	3	0	When asserted, indicates Refresh request to the specific rank. Each bit corresponds to each rank.	Read/Write

## sideband2: Sideband

address=45(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_zqcal_long_req	0	0	When asserted, indicates long ZQ cal request. This bit is write clear.	Read/Write

## sideband3: Sideband

address=46(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_zqcal_short_req	0	0	When asserted, indicates short ZQ cal request. This bit is write clear.	Read/Write

## sideband4: Sideband

address=47(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_self_rfsh_req	3	0	When asserted, indicates self refresh request to the specific rank. Each bit corresponds to each rank. These bits are write clear.	Read/Write

## sideband5: Sideband

address=48(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_dpd_mps_req	0	0	When asserted, indicates deep power down or max power saving request. This bit is write clear.	Read/Write

## sideband6: Sideband

address=49(32 bit)

Field	Bit High	Bit Low	Description	Access
mr_cmd_ack	0	0	Acknowledge to mode register command.	Read

## sideband7: Sideband

address=50(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_refresh_ack	0	0	Acknowledge to indicate refresh is in progress.	Read

## sideband8: Sideband

**address=51(32 bit)**

Field	Bit High	Bit Low	Description	Access
mmr_zqcal_ack	0	0	Acknowledge to indicate ZQCAL is progressing.	Read

**sideband9: Sideband****address=52(32 bit)**

Field	Bit High	Bit Low	Description	Access
mmr_self_rfsh_ack	0	0	Acknowledge to indicate self refresh is progressing.	Read

**sideband10: Sideband****address=53(32 bit)**

Field	Bit High	Bit Low	Description	Access
mmr_dpd_mps_ack	0	0	Acknowledge to indicate deep power down/max power saving is in progress.	Read

**sideband11: Sideband****address=54(32 bit)**

Field	Bit High	Bit Low	Description	Access
mmr_auto_pd_ack	0	0	Acknowledge to indicate auto power down is in progress.	Read

**sideband12: Sideband****address=55(32 bit)**

Field	Bit High	Bit Low	Description	Access
mr_cmd_type	2	0	Indicates the type of Mode Register Command.	Read/Write
mr_cmd_rank	6	3	Indicates which rank the mode register command is intended to.	Read/Write

**sideband13: Sideband****address=56(32 bit)**

Field	Bit High	Bit Low	Description	Access
mr_cmd_opcode	31	0	[31:27] reserved. Register Command Opcode Information to be used for Register Command LPDDR3 [26:20] Reserved [19:10] falling edge CA. [9:0] rising edge CA DDR4 [26:24] C2:C0 [23] ACT [22:21] BG1:BG0 [20] Reserved [19:18] BA1:BA0 [17] A17 [16] RAS [15] CAS [14] WE [13:0] A13:A0 DDR3 [26:21] Reserved [20:18] BA2:BA0 [17] A17 [16] RAS [15] CAS [14] WE [13] Reserved [12:0] A12:A0 RLDRAM3 [26] Reserved [25:22] BA3:BA0 [21] REF [20] WE [19:0] A19:A0	Read/Write

**sideband14: Sideband****address=57(32 bit)**

Field	Bit High	Bit Low	Description	Access
mmr_refresh_bank	15	0	User refresh bank information, binary representation of bank address. Enables refresh to that bank address when requested.	Read/Write

**sideband15: Sideband****address=58(32 bit)**

Field	Bit High	Bit Low	Description	Access
mmr_stall_rank	3	0	Setting to 1 to stall the corresponding rank.	Read/Write

**dramsts: Calibration Status**

**address=59(32 bit)**

Field	Bit High	Bit Low	Description	Access
phy_cal_success	0	0	This bit will be set to 1 if the PHY was able to successfully calibrate.	Read
phy_cal_fail	1	1	This bit will be set to 1 if the PHY was unable to calibrate.	Read

## ecc1: ECC General Configuration

**address=128(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_enable_ecc	0	0	A value of 1 enables the ECC encoder/decoder.	Read/Write
cfg_enable_dm	1	1	A value of 1 indicate that this is a design with DM.	Read/Write
cfg_enable_rmw	2	2	A value of 1 enables the RMW feature, including partial/dummy write support.	Read/Write
cfg_data_rate	6	3	Set this value to 2, 4, or 8 for full, half or quarter rate designs.	Read
cfg_ecc_in_protocol	7	7	Set this value to 1 for Avalon-MM or 0 for Avalon-ST input interface. Read-only register.	Read/Write
cfg_enable_auto_corr	8	8	A value of 1 enables the auto correction feature, injecting a dummy write command after a single-bit error is (SBE) detected. This feature must be enabled together with RMW and ECC.	Read/Write
cfg_enable_ecc_code_overwrite	9	9	A value of 1 enables the ECC code overwrite feature. Reuse the original read-back ECC code during RMW if a double-bit error (DBE) is detected.	Read/Write
Reserved	31	10		

## ecc2: Width Configuration

**address=129(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_dram_data_width	7	0	Set this value to the DRAM data width, without taking rate conversion into consideration. For example, for a 64+8=72 bit ECC design, set this register to 'd72.	Read
cfg_local_data_width	15	8	Set this value to the LOCAL data width, without taking rate conversion into consideration. For example, for a 64+8=72 DQ bit ECC design, set this register to 'd64.	Read
cfg_addr_width	21	16	Set this value to the LOCAL address width, which corresponds to the address field width of the cmd field. For example, for a LOCAL address width of 24 bits, set this value to 'd24.	Read
Reserved	31	22		

**ecc3: ECC Error and Interrupt Configuration****address=130(32 bit)**

Field	Bit High	Bit Low	Description	Access
cfg_gen_sbe	0	0	A value of 1 enables the generate SBE feature. Generates a single bit error during the write process.	Read/Write
cfg_gen_dbe	1	1	A value of 1 enables the generate DBE feature. Generates a double bit error during the write process.	Read/Write
cfg_enable_intr	2	2	A value of 1 enables the interrupt feature. The interrupt signal notifies if an error condition occurs. The condition is configurable.	Read/Write
cfg_mask_sbe_intr	3	3	A value of 1 masks the interrupt signal when SBE occurs.	Read/Write

Field	Bit High	Bit Low	Description	Access
cfg_mask_dbe_intr	4	4	A value of 1 masks the interrupt signal when DBE occurs.	Read/Write
cfg_mask_corr_dropped_intr	5	5	A value of 1 masks the interrupt signal when auto correction command can't be scheduled, due to back-pressure (FIFO full).	Read/Write
cfg_mask_hmi_intr	6	6	A value of 1 masks the interrupt signal when the hard memory interface asserts an interrupt signal via hmi_interrupt port.	Read/Write
cfg_clr_intr	7	7	Writing a vale of 1 to this self-clearing bit clears the interrupt signal, error status, and address.	Read/Write
Reserved	31	9		

## ecc4: Status and Error Information

address=144(32 bit)

Field	Bit High	Bit Low	Description	Access
sts_ecc_intr	0	9	Indicates the interrupt status; a value of 1 indicates interrupt occurred.	Read
sts_sbe_error	1	1	Indicates the SBE status; a value of 1 indicates SBE occurred.	Read
sts_dbe_error	2	2	Indicates the DBE status; a value of 1 indicates DBE occurred.	Read
sts_corr_dropped	3	3	Indicates the status of correction command dropped; a value of 1 indicates correction command dropped.	Read
sts_sbe_count	7	4	Indicates the number of times SBE error has occurred. The counter will overflow.	Read
sts_dbe_count	11	8	Indicates the number of times DBE error has occurred. The counter will overflow.	Read

Field	Bit High	Bit Low	Description	Access
sts_corr_dropped_count	15	12	Indicates the number of times correction command has dropped. Counter will overflow.	Read
Reserved	31	16		

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"> <li>Modified text of <i>Hard Memory Controller and Hard PHY</i> and <i>Soft Memory Controller and Hard PHY</i>.</li> <li>Modified content of the <i>Ping Pong PHY Architecture</i> topic.</li> <li>Added section on read-modify-write operations, to <i>ECC in Arria 10 EMIF IP</i> topic.</li> <li>Added <i>AFI 4.0 Timing Diagrams</i> section.</li> <li>Added <i>Arria 10 EMIF Latency</i> section.</li> <li>Added <i>Arria 10 EMIF Calibration Times</i> section.</li> <li>Added <i>Integrating a Custom Controller with the Hard PHY</i> section.</li> </ul>
November 2015	2015.11.02	<ul style="list-style-type: none"> <li>Added LPDDR3 support to <i>AFI Parameters</i>, <i>AFI Address and Command Signals</i>, and <i>AFI Write Data Signals</i>.</li> <li>Revised rendering of address information in the <i>Memory Mapped Register (MMR) Tables</i>.</li> <li>Added <i>ecc1: ECC General Configuration</i>, <i>ecc2: Width Configuration</i>, <i>ecc3: ECC Error and Interrupt Configuration</i>, and <i>ecc4: Status and Error Information</i> in the <i>Memory Mapped Register (MMR) Tables</i>.</li> <li>Removed <i>RZQ Pin Sharing</i> section.</li> <li>Added bank numbers to figure in <i>Restrictions on I/O Bank Usage for Arria 10 EMIF IP with HPS</i> topic.</li> <li>Added <i>Arria 10 EMIF and SmartVID</i> topic.</li> <li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> </ul>
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	<ul style="list-style-type: none"> <li>Added debug-related connections to the <i>Logical Connections</i> table in the <i>Logical Connections</i> topic.</li> <li>Added <i>Arria 10 EMIF Ping Pong PHY</i> section.</li> <li>Added <i>Arria 10 EMIF Debugging Examples</i> section.</li> <li>Added x4 mode support.</li> <li>Added <i>AFI 4.0 Specification</i> section.</li> <li>Added <i>MMR Register Tables</i> section.</li> </ul>

Date	Version	Changes
August 2014	2014.08.15	<ul style="list-style-type: none"> <li>Added PHY-only support for DDR3, DDR4, and RLDRAM 3, and soft controller and hard PHY support for RLDRAM 3 and QDR II/II +/II+ Xtreme to <i>Supported Memory Protocols</i> table.</li> <li>Added <code>afi_conduit_end</code>, <code>afi_clk_conduit_end</code>, <code>afi_half_clk_conduit_end</code>, and <code>afi_reset_n_conduit_end</code> to <i>Logical Connections</i> table.</li> <li>Expanded description of <code>ctrl_amm_avalon_slave</code> in <i>Logical Connections</i> table.</li> <li>Added <i>ECC in Arria 10 EMIF IP</i>.</li> <li>Added <i>Configuring Your EMIF IP for Use with the Debug Toolkit</i>.</li> <li>Added <i>Arria 10 EMIF for Hard Processor Subsystem</i>.</li> </ul>
December 2013	2013.12.16	Initial release.

# Functional Description—MAX 10 EMIF

3

2016.05.02

EMI\_RM



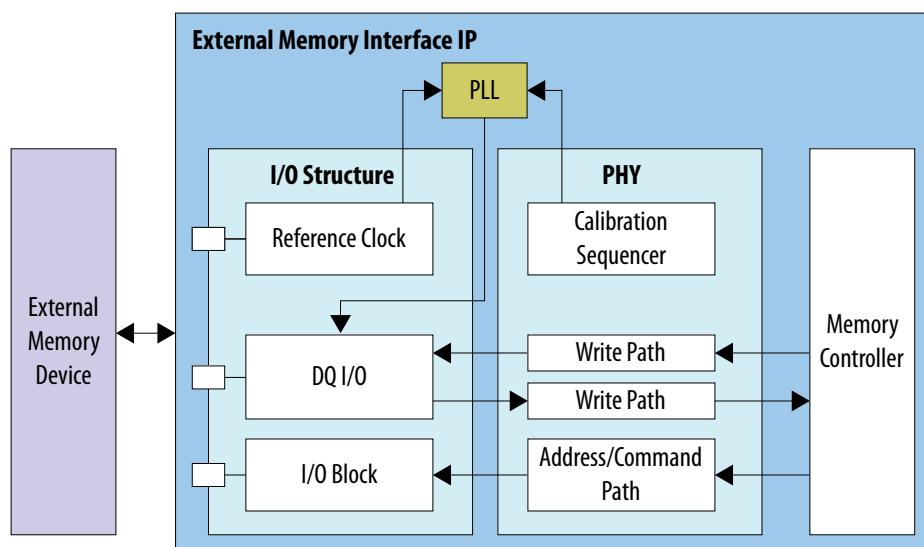
Subscribe



Send Feedback

MAX® 10 FPGAs provide advanced processing capabilities in a low-cost, instant-on, small-form-factor device featuring capabilities such as digital signal processing, analog functionality, Nios II embedded processor support and memory controllers.

Figure 3-1: MAX 10 EMIF Block Diagram

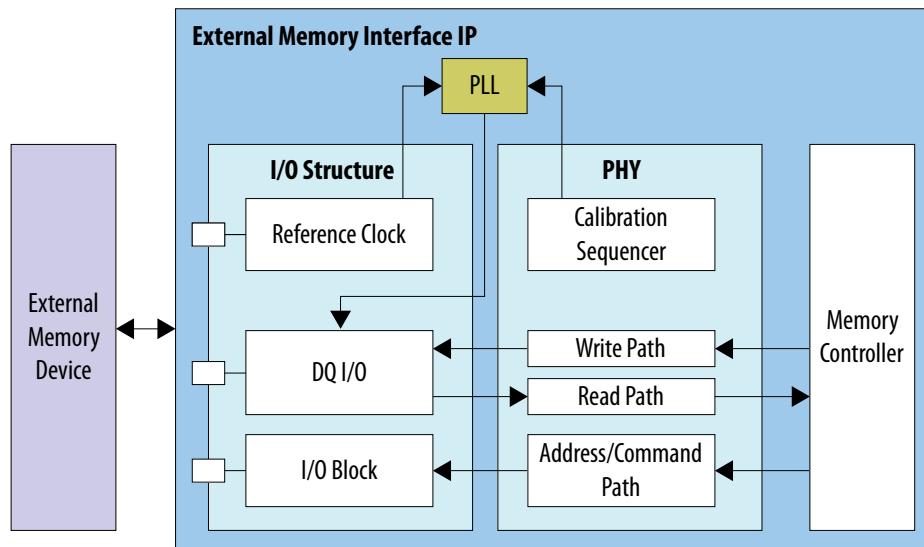


## MAX 10 EMIF Overview

MAX® 10 FPGAs provide advanced processing capabilities in a low-cost, instant-on, small-form-factor device featuring capabilities such as digital signal processing, analog functionality, Nios II embedded processor support and memory controllers.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

**Figure 3-2: MAX 10 EMIF Block Diagram**

## External Memory Protocol Support

MAX 10 FPGAs offer external memory interface support for DDR2, DDR3, DDR3L, and LPDDR2 protocols.

**Table 3-1: Supported External Memory Configurations**

External Memory Protocol	Maximum Frequency	Configuration
DDR3, DDR3L	303 MHz	x16 + ECC + Configuration and Status Register (CSR)
DDR2	200 MHz	x16 + ECC + Configuration and Status Register (CSR)
LPDDR2 <sup>(1)</sup>	200 MHz <sup>(2)</sup>	x16

## MAX 10 Memory Controller

MAX 10 FPGAs use the HPC II external memory controller.

<sup>(1)</sup> MAX® 10 devices support only single-die LPDDR2.

<sup>(2)</sup> To achieve the specified performance, constrain the memory device I/O and core power supply variation to within ±3%. By default, the frequency is 167 MHz.

## MAX 10 Low Power Feature

The MAX 10 low power feature is automatically activated when the self refresh or low power down modes are activated. The low power feature sends the `afi_mem_clk_disable` signal to stop the clock used by the controller.

To conserve power, the MAX 10 UniPHY IP core performs the following functions:

- Tri-states the address and command signals except `CKE` and `RESET_N` signals
- Disables the input buffer of DDR input

**Note:** The MAX 10 low power feature is available from version 15.0 of the Quartus® Prime software. To enable this feature, regenerate your MAX 10 UniPHY IP core using the Quartus Prime software version 15.0 or later.

## MAX 10 Memory PHY

MAX 10 devices employ UniPHY, but without using DLLs for calibration. The physical layer implementation for MAX 10 external memory interfaces provides a calibrated read path and a static write path.

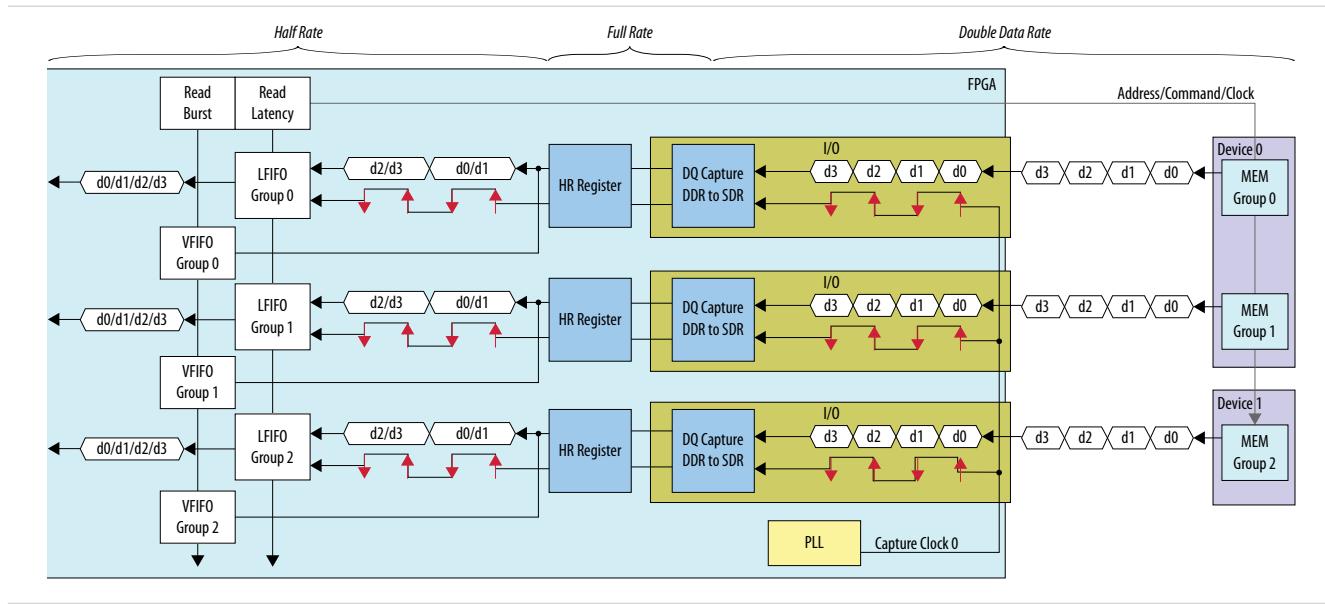
### Supported Topologies

The memory PHY supports DDR2 and DDR3 protocols with up to two discrete memory devices, and the LPDDR2 protocol with one discrete memory device.

### Read Datapath

One PLL output is used to capture data from the memory during read operations. The clock phase is calibrated by the sequencer before the interface is ready for use.

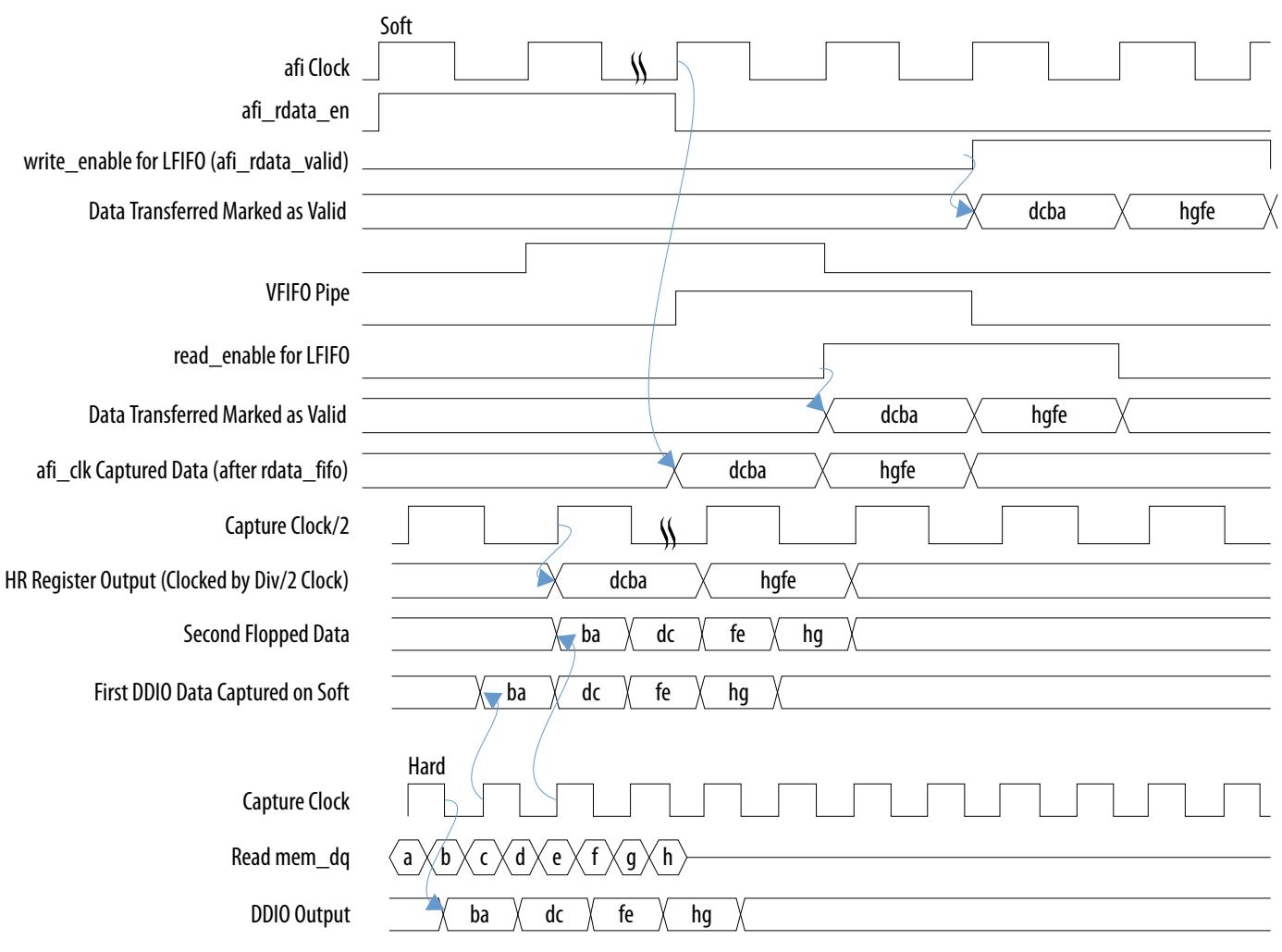
**Figure 3-3: Read Datapath**



For DDR3 interfaces, two PLL outputs capture data from the memory devices during a read. In a 24-bit interface—whether the top 8 bits are used by ECC or not—the supported topology is two discrete DDR3 devices of 16-bit and 8-bit DQ each. Each discrete device has a dedicated capture clock output from the PLL.

For LPDDR2 interfaces, the supported configuration is a single memory device with memory width of 16-bit DQ. The other PLL output is used for DQS tracking purposes, because the tDQSCK drift might cause data capture to fail otherwise. The tracking clock is a shifted capture clock used to sample the DQS signal. By capturing the DQS signal, the system can compensate for DQS signal drift.

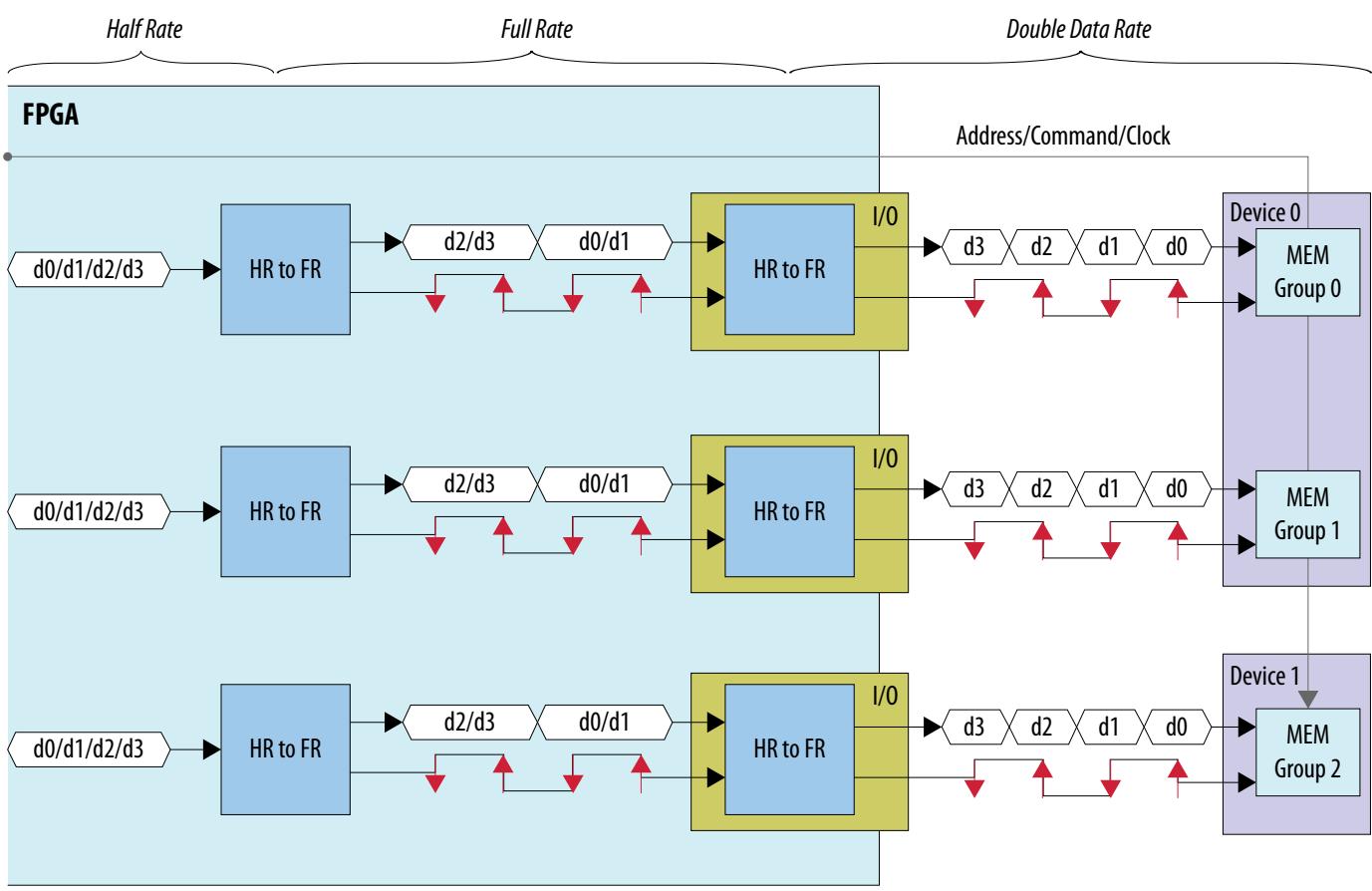
**Figure 3-4: Read Datapath Timing Diagram**



## Write Datapath

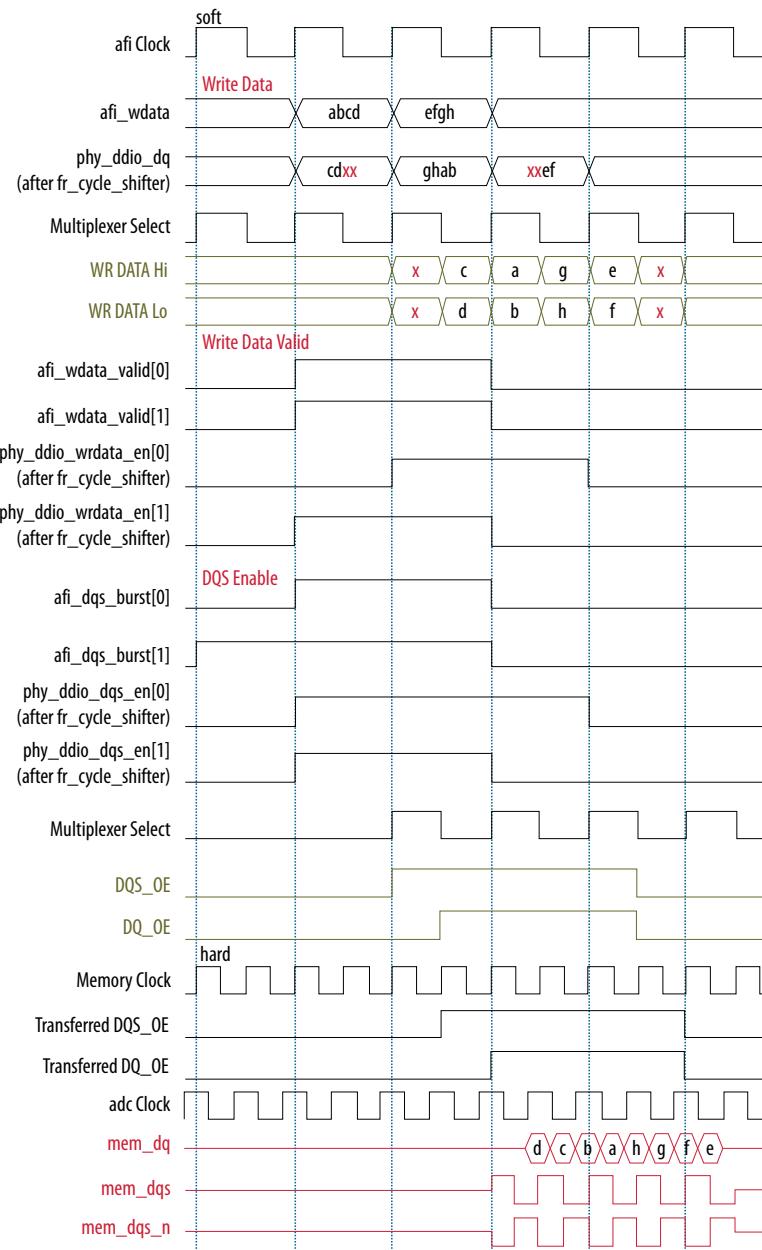
The write datapath is a static path and is timing analyzed to meet timing requirements.

Figure 3-5: Write Datapath



In the PHY write data path, for even write latency the `write_data_valid`, `write_data`, and `dqs` enable pass through one stage of `fr_cycle_shifter` in a flow through path. For odd memory write latency, the output is shifted by a full-rate clock cycle. The full-rate cycle-shifted output feeds into a simple DDIO.

Figure 3-6: Write Datapath Timing Diagram



## Address and Command Datapath

Implementation of the address and command datapath differs between DDR2/DDR3 and LPDDR2, because of the double data-rate command in LPDDR2.

### LPDDR2

For LPDDR2, CA is four bits wide on the AFI interface. The least significant bit of CA is captured by a negative-edge triggered flop, and the most-significant bit of CA is captured by a positive-edge triggered

flop, before multiplexing to provide maximum setup and hold margins for the AFI clock-to-MEM clock data transfer.

## DDR2/DDR3

For DDR2/DDR3, the full-rate register and MUX architecture is similar to LPDDR2, but both phases are driven with the same signal. The DDR3 address and command signal is not clocked by the write/ADC clock as with LPDDR2, but by the inverted MEM clock, for better address and command margin.

## Chip Select

Because the memory controller can drive both phases of `cs_n` in half-rate, the signal is fully exposed to the AFI side.

## Sequencer

The sequencer employs an RTL-based state machine which assumes control of the interface at reset (whether at initial startup or when the IP is reset) and maintains control throughout the calibration process. The sequencer relinquishes control to the memory controller only after successful calibration.

The sequencer consists of a calibration state machine, together with a read-write (RW) manager, PHY Manager, and PLL Manager.

**Figure 3-7: Sequencer Architecture**

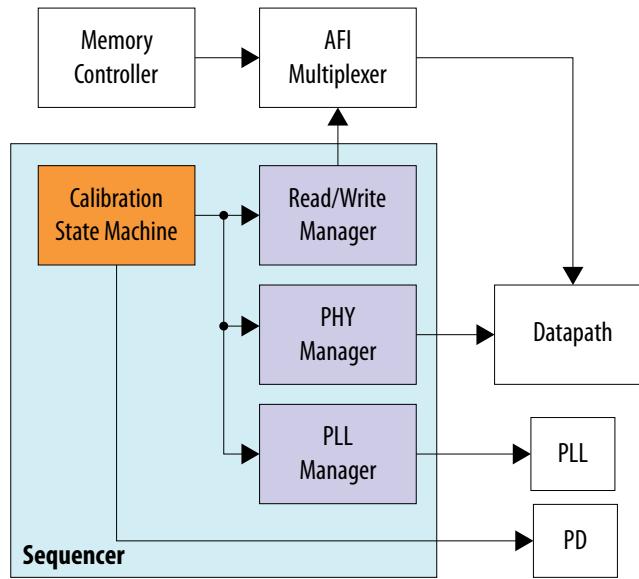
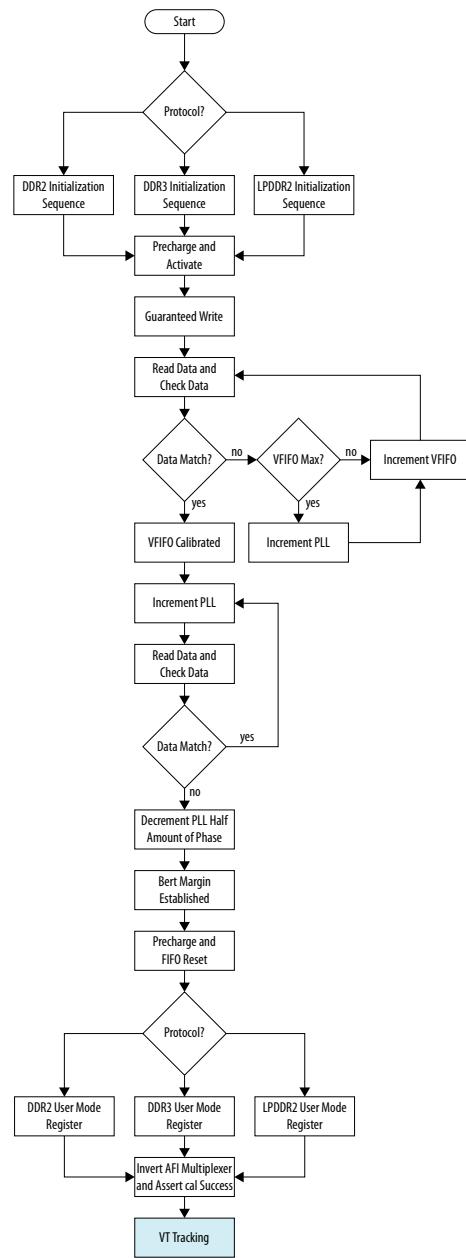


Figure 3-8: Calibration State Machine Stages



## RW Manager

The read-write (RW) manager encapsulates the protocol to read and write to the memory device through the Altera PHY Interface (AFI). It provides a buffer that stores the data to be sent to and read from memory, and provides the following commands:

- Write configuration—configures the memory for use. Sets up burst lengths, read and write latencies, and other device specific parameters.
- Refresh—initiates a refresh operation at the DRAM. The sequencer also provides a register that determines whether the RW manager automatically generates refresh signals.
- Enable or disable multi-purpose register (MPR)—for memory devices with a special register that contains calibration specific patterns that you can read, this command enables or disables access to the register.
- Activate row—for memory devices that have both rows and columns, this command activates a specific row. Subsequent reads and writes operate on this specific row.
- Precharge—closes a row before you can access a new row.
- Write or read burst—writes or reads a burst length of data.
- Write guaranteed—writes with a special mode where the memory holds address and data lines constant. Altera guarantees this type of write to work in the presence of skew, but constrains to write the same data across the entire burst length.
- Write and read back-to-back—performs back-to-back writes or reads to adjacent banks. Most memory devices have strict timing constraints on subsequent accesses to the same bank, thus back-to-back writes and reads have to reference different banks.
- Protocol-specific initialization—a protocol-specific command required by the initialization sequence.

## PHY Manager

The PHY Manager provides access to the PHY for calibration, and passes relevant calibration results to the PHY. For example, the PHYManager sets the VFIFO and LFIFO buffer parameters resulting from calibration, signals the PHY when the memory initialization sequence finishes, and reports the pass/fail status of calibration.

## PLL Manager

The PLL Manager controls the phase of capture clocks during calibration. The output phases of individual PLL outputs can be dynamically adjusted relative to each other and to the reference clock without having to load the scan chain of the PLL. The phase is shifted by 1/8th of the period of the voltage-controlled oscillator (VCO) at a time. The output clocks are active during this dynamic phase-shift process.

A PLL counter increments with every phase increase and decrements with every phase reduction. The PLL Manager records the amount by which the PLL counter has shifted since the last reset, enabling the sequencer and tracking manager to determine whether the phase boundary has been reached.

# Calibration

The purpose of calibration is to exercise the external memory interface to find an optimal window for capture clock. There is no calibration for writing data or for address and command output; these paths are analyzed by TimeQuest to meet timing requirements.

The calibration algorithm assumes that address and command signals can reliably be sent to the external memory device, and that write data is registered with DQS.

## Read Calibration

Read calibration consists of two primary parts: capture clock and VFIFO buffer calibration, and read latency tuning.

A VFIFO buffer is a FIFO buffer that is calibrated to reflect the read latency of the interface. The calibration process selects the correct read pointer of the FIFO. The VFIFO function delays the controller's `afi_rdata_valid` signal to align to data captured internally to the PHY. LFIFO buffers are FIFO buffers which ensure that data from different DQS groups arrives at the user side at the same time. Calibration ensures that data arrives at the user side with minimal latency.

### Capture Clock and VFIFO Calibration

Capture clock and VFIFO calibration performs the following steps:

- Executes a guaranteed write routine in the read-write (RW) manager.
  - Sweeps VFIFO buffer values, beginning at 0, in the PHY Manager.
    - For each adjustment of the VFIFO buffer, sweeps the capture clock phase in the PLL Manager to find the first phase that works. This is accomplished by issuing a read to the RW Manager and performing a bit check. Data is compared for all data bits.
  - Increments the capture clock phase in the PLL Manager until capture clock phase values are exhausted or until the system stops working. If there are no more capture clock phase values to try, calibration increments the VFIFO buffer value in the PHY Manager, and phase sweep again until the system stops working.
- Completion of this step establishes a working range of values.
- As a final step, calibration centers the capture clock phase within the working range.

### Read Latency Tuning

Read latency tuning performs the following steps to achieve the optimal latency value:

- Assigns one LFIFO buffer for each DQ group.
- Aligns read data to the AFI clock.
- Gradually reduces LFIFO latency until reads fail, then increases latency to find the minimum value that yields reliable operation.

## Write Calibration

There is no calibration routine for writing data.

## Sequencer Debug Information

Following calibration, the sequencer loads a set of debug information onto an output port. You can use the SignalTap II Logic Analyzer to access the debug information in the presynthesized design.

You could also bring this port to a register, to latch the value and make it accessible to a host processor.

The output port where the debug information is available is not normally connected to anything, so it could be removed during synthesis.

Signal name: `phy_cal_debug_info`

Module: `<corename>_s0.v`

The signal is 32 bits wide, and is defined in the following table.

**Table 3-2: Sequencer Debug Data**

best_comp_result [23:16]	Best data comparison result on a per-pin basis from the Read-Write Manager. Pin 16 - 23.  Any mismatch on respective pin data comparison produces a high bit.  If calibration fails, the result is a non-zero value. When calibration passes, the result is zero.
best_comp_result [15:8]	Best data comparison result on a per-pin basis from the Read-Write Manager. Pin 8 - 15.  Any mismatch on respective pin data comparison produces a high bit.  If calibration fails, the result is a non-zero value. When calibration passes, the result is zero.
best_comp_result [7:0]	Best data comparison result on a per-pin basis from the Read-Write Manager. Pin 0 - 7.  Any mismatch on respective pin data comparison produces a high bit.  If calibration fails, the result is a non-zero value. When calibration passes, the result is zero.
margin [7:0]	Margin found by the sequencer if calibration passes. Number represents the amount of subsequent PLL phase where valid data is found.  If calibration fails, this number is zero. When calibration passes, this value is non-zero.

**Debug Example and Interpretation**

The following table illustrates possible debug signal values and their interpretations.

	best_comp_result [23:16]	best_comp_result [15:8]	best_comp_result [7:0]	margin [7:0]
Passing result	0000 0000	0000 0000	0000 0000	0001 0000
Interpretation	No failing pins.	No failing pins.	No failing pins.	16 phases of margin for valid window. Ideal case for 300Mhz interface.
Failing result	0010 0000	1111 1111	0000 0000	0000 0000
Interpretation	Pin 21 failing.	All pins failing, pin 8-15.	No failing pins.	No valid window.

## Register Maps

This topic provides register information for MAX 10 EMIF.

**Table 3-3: Controller Register Map**

Address	Description
0x100 - 0x126	Reserved
0x130	ECC control register
0x131	ECC status register
0x132	ECC error address register

UniPHY does not have a configuration and status register. The HPC II controller does have a configuration and status register, when configured for ECC.

### Related Information

[Soft Controller Register Map](#) on page 6-35

## Document Revision History

**Table 3-4: Document Revision History**

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
May 2015	2015.05.04	<ul style="list-style-type: none"> <li>Updated the external memory protocol support.</li> <li>Added topic about the low power feature.</li> </ul>
December 2014	2014.12.15	Initial release.

# Functional Description—Hard Memory Interface

4

2016.05.02

EMI\_RM



Subscribe



Send Feedback

The hard (on-chip) memory interface components are available in the Arria V and Cyclone V device families.

The Arria V device family includes hard memory interface components supporting DDR2 and DDR3 SDRAM memory protocols at speeds of up to 533 MHz. For the Quartus II software version 12.0 and later, the Cyclone V device family supports both hard and soft interface support.

The Arria V device family supports both hard and soft interfaces for DDR3 and DDR2, and soft interfaces for LPDDR2 SDRAM, QDR II SRAM, and RLDRAM II memory protocols. The Cyclone V device family supports both hard and soft interfaces for DDR3, DDR2, and LPDDR2 SDRAM memory protocols.

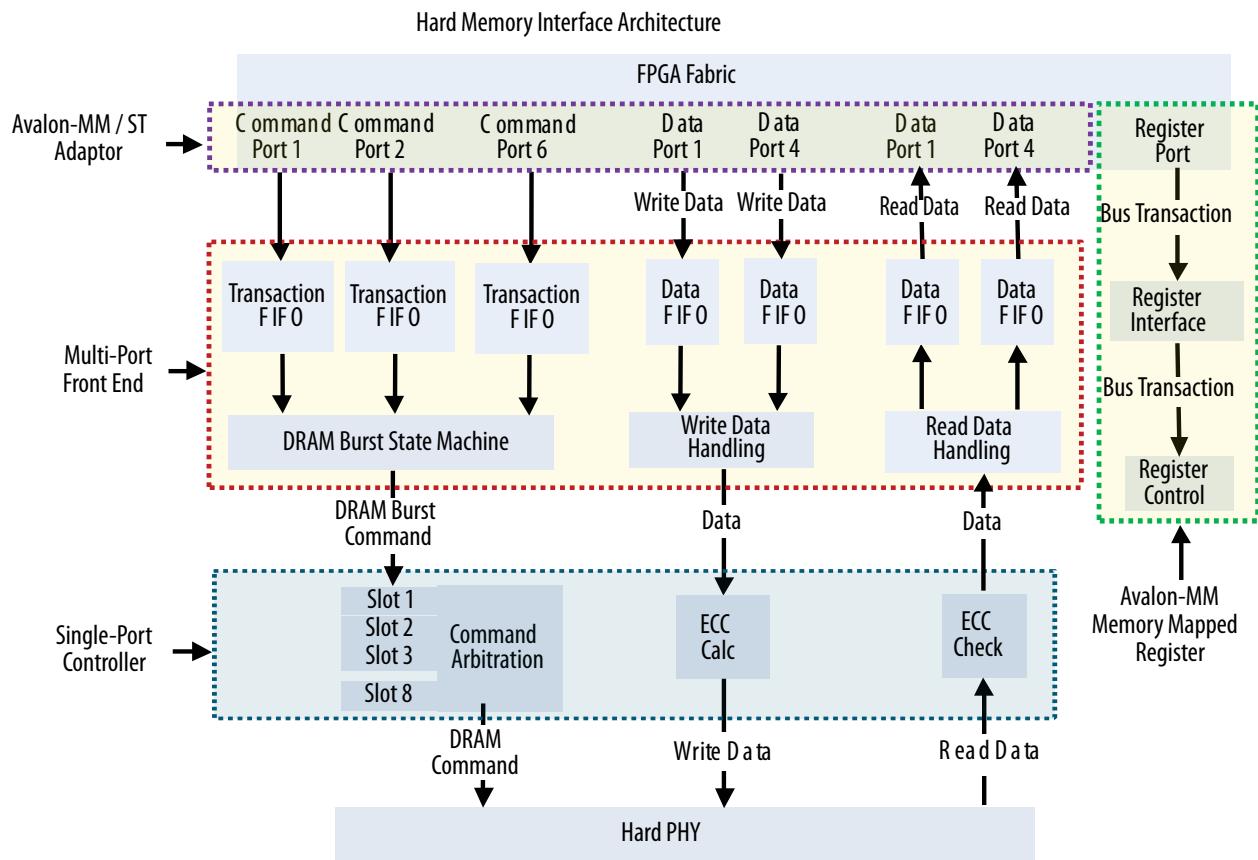
The hard memory interface consists of three main parts, as follows:

- The multi-port front end (MPFE), which allows multiple independent accesses to the hard memory controller.
- The hard memory controller, which initializes, refreshes, manages, and communicates with the external memory device.
- The hard PHY, which provides the physical layer interface to the external memory device.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

Figure 4-1: Hard Memory Interface Architecture



## Multi-Port Front End (MPFE)

The multi-port front end and its associated fabric interface provide up to six command ports, four read-data ports and four write-data ports, through which user logic can access the hard memory controller. Each port can be configured as read only or write only, or read and write ports may be combined to form bidirectional data ports. Ports can be 32, 64, 128, or 256 data bits wide, depending on the number of ports used and the type (unidirectional or bidirectional) of the port.

### Fabric Interface

The fabric interface provides communication between the Avalon-ST-like internal protocol of the hard memory interface and the external Avalon-MM protocol. The fabric interface supports frequencies in the range of 10 MHz to one-half of the memory interface frequency. For example, for an interface running at 533 MHz, the maximum user logic frequency is 267 MHz. The MPFE handles the clock crossing between user logic and the hard memory interface.

The multi-port front end read and write FIFO depths are 8, and the command FIFO depth is 4. The FIFO depths are not configurable.

### Operation Ordering

Requests arriving at a given port are executed in the order in which they are received.

Requests arriving at different ports have no guaranteed order of service, except when a first transaction has completed before the second arrives.

## Multi-port Scheduling

Multi-port scheduling is governed by two considerations: the absolute priority of a request and the weighting of a port. User-configurable priority and weight settings determine the absolute and relative scheduling policy for each port.

### Port Scheduling

The evaluation of absolute priority ensures that ports carrying higher-priority traffic are served ahead of ports carrying lower-priority traffic. The scheduler recognizes eight priority levels, with higher values representing higher priorities. Priority is absolute; for example, any transaction with priority seven will always be scheduled before transactions of priority six or lower.

When ports carry traffic of the same absolute priority, relative priority is determined based on port weighting. Port weighting is a five-bit value, and is determined by a weighted round robin (WRR) algorithm.

The scheduler can alter priority if the latency target for a transaction is exceeded. The scheduler tracks latency on a per-port basis, and counts the cycles that a transaction is pending. Each port has a priority escalation register and a pending counter engagement register. If the number of cycles in the pending counter engagement register elapse without a pending transaction being served, that transaction's priority is escalated.

To ensure that high-priority traffic is served quickly and that long and short bursts are effectively interleaved on ports, bus transactions longer than a single DRAM burst are scheduled as a series of DRAM bursts, with each burst arbitrated separately.

The scheduler uses a form of deficit round robin (DRR) scheduling algorithm which corrects for past over-servicing or under-servicing of a port. Each port has an associated weight which is updated every cycle, with a user-configured weight added to it and the amount of traffic served subtracted from it. The port with the highest weighting is considered the most eligible.

To ensure that lower priority ports do not build up large running weights while higher priority ports monopolize bandwidth, the hard memory controller's DRR weights are updated only when a port matches the scheduled priority. Hence, if three ports have traffic, two being priority 7 and one being priority 4, the weights for both ports at priority 7 are updated but the port with priority 4 remains unchanged.

### DRAM Burst Scheduling

DRAM burst scheduling recognizes addresses that access the same column/row combination—also known as open page accesses. Such operations are always served in the order in which they are received in the single-port controller.

Selection of DRAM operations is a two-stage process; first, each pending transaction must wait for its timers to be eligible for execution, then the transaction arbitrates against other transactions that are also eligible for execution.

The following rules govern transaction arbitration:

- High priority operations take precedence over lower priority operations
- If multiple operations are in arbitration, read operations have precedence over write operations
- If multiple operations still exist, the oldest is served first

A high-priority transaction in the DRAM burst scheduler wins arbitration for that bank immediately if the bank is idle and the high-priority transaction's chip select/row/column address does not match an address already in the single-port controller. If the bank is not idle, other operations to that bank yield until the high-priority operation is finished. If the address matches another chip select/row/column, the high-priority transaction yields until the earlier transaction is completed.

You can force the DRAM burst scheduler to serve transactions in the order that they are received, by setting a bit in the register set.

## DRAM Power Saving Modes

The hard memory controller supports two DRAM power-saving modes: self-refresh, and fast/slow all-bank precharge powerdown exit. Engagement of a DRAM power saving mode can occur due to inactivity, or in response to a user command.

The user command to enter power-down mode forces the DRAM burst-scheduling bank-management logic to close all banks and issue the power-down command. You can program the controller to power down when the DRAM burst-scheduling queue is empty for a specified number of cycles; the DRAM is reactivated when an active DRAM command is received.

## MPFE Signal Descriptions

The following table describes the signals for the multi-port front end.

**Table 4-1: MPFE Signals**

Signal	Direction	Description
avl_<signal_name>_# <sup>(1)</sup>	—	Local interface signals.
mp_cmd_clk_#_clk <sup>(1)</sup>	Input	Clock for the command FIFO buffer. <sup>(3)</sup> Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.
mp_cmd_reset_n_#_reset_n <sup>(1)</sup>	Input	Asynchronous reset signal for command FIFO buffer.
mp_rfifo_clk_#_clk <sup>(2)</sup>	Input	Clock for the read data FIFO buffer. Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.
mp_rfifo_reset_n_#_reset_n <sup>(2)</sup>	Input	Asynchronous reset signal for read data FIFO buffer.

Signal	Direction	Description
mp_wfifo_clk_#_clk <sup>(2)</sup>	Input	Clock for the write data FIFO buffer. Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.
mp_wfifo_reset_n_#_reset_n <sup>(2)</sup>	Input	Asynchronous reset signal for write data FIFO buffer.
bonding_in_1/2/3	Input	Bonding interface input port. Connect second controller bonding output port to this port according to the port sequence.
bonding_out_1/2/3	Output	Bonding interface output port. Connect this port to the second controller bonding intput port according to the port sequence.

Notes to Table:

1. # represents the number of the slave port. Values are 0—5.
2. # represents the number of the slave port. Values are 0—3.
3. The command FIFO buffers have two stages. The first stage is 4 bus transaction deep per port. After port scheduling, the commands are placed in the second stage FIFO buffer, which is 8 DRAM transactions deep. The second stage FIFO buffer is used to optimize memory operations where bank look ahead and data reordering occur. The write data buffer is 32 deep and read buffer is 64 deep.

Every input interface (command, read data, and write data) has its own clock domain. Each command port can be connected to a different clock, but the read data and write data ports associated with a command port must connect to the same clock as that command port. Each input interface uses the same reset signal as its clock.

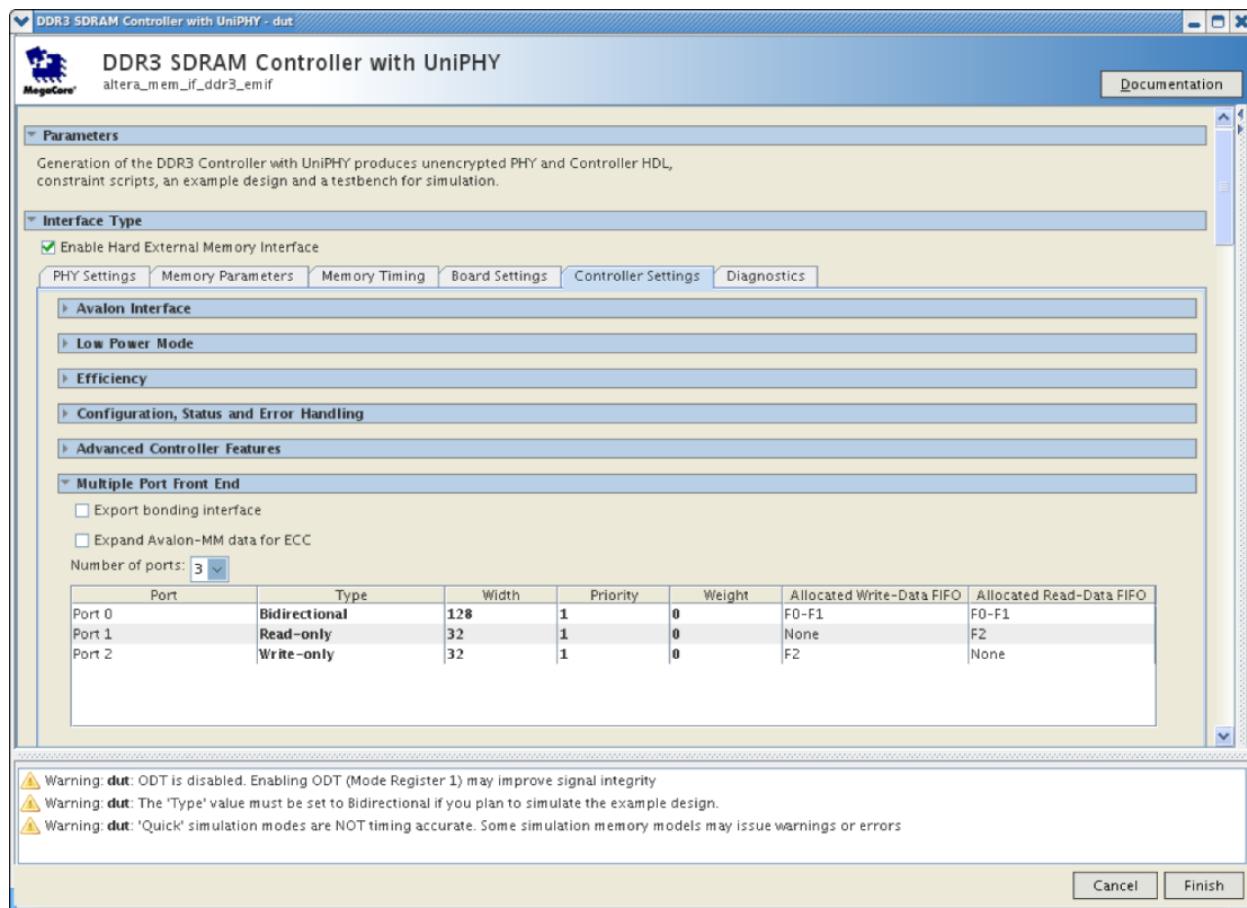
By default, the IP generates all clock signals regardless of the MPFE settings, but all unused ports and FIFO buffers are connected to ground.

The command ports can be used only in unidirectional configurations, with either 4 write and 2 read, 3 write and 3 read, or 2 write and 4 read scenarios. For bidirectional ports, the number of clocks is reduced from 6 to a maximum of 4.

For the scenario depicted in the following figure:

- command port 0 is associated with read and write data FIFO 0 and 1
- command port 1 is associated with read data FIFO 2
- command port 2 is associated with write data FIFO 2

Figure 4-2: Sample MPFE Configuration



Therefore, if port 0 (avl\_0) is clocked by a 100 MHz clock signal, mp\_cmd\_clk\_0, mp\_rfifo\_clk\_0, mp\_rfifo\_clk\_1, mp\_wfifo\_clk\_0, and mp\_wfifo\_clk\_1 must all be connected to the same 100 MHz clock, as illustrated below.

Figure 4-3: Sample Connection Mapping

Use	Connections	Name	Description
<input checked="" type="checkbox"/>		<b>100_MHz</b>	Clock Source
	clk_in	clk_in	Clock Input
	clk_in_reset	clk_in_reset	Reset Input
	clk	clk	Clock Output
	clk_reset	clk_reset	Reset Output
<input checked="" type="checkbox"/>		<b>125_MHz</b>	Clock Source
	clk_in	clk_in	Clock Input
	clk_in_reset	clk_in_reset	Reset Input
	clk	clk	Clock Output
	clk_reset	clk_reset	Reset Output
<input checked="" type="checkbox"/>		<b>200_MHz</b>	Clock Source
	clk_in	clk_in	Clock Input
	clk_in_reset	clk_in_reset	Reset Input
	clk	clk	Clock Output
	clk_reset	clk_reset	Reset Output
<input checked="" type="checkbox"/>		<b>mem_if_ddr3_emif_0</b>	DDR3 SDRAM Controller with UniPHY
	pll_ref_clk	pll_ref_clk	Clock Input
	global_reset	global_reset	Reset Input
	soft_reset	soft_reset	Reset Input
	afi_clk	afi_clk	Clock Output
	afi_half_clk	afi_half_clk	Clock Output
	afi_reset	afi_reset	Reset Output
	memory	memory	Conduit
	avl_0	avl_0	Avalon Memory Mapped Slave
	avl_1	avl_1	Avalon Memory Mapped Slave
	avl_2	avl_2	Avalon Memory Mapped Slave
	mp_cmd_clk_0	mp_cmd_clk_0	Clock Input
	mp_cmd_reset_n_0	mp_cmd_reset_n_0	Reset Input
	mp_cmd_clk_1	mp_cmd_clk_1	Clock Input
	mp_cmd_reset_n_1	mp_cmd_reset_n_1	Reset Input
	mp_cmd_clk_2	mp_cmd_clk_2	Clock Input
	mp_cmd_reset_n_2	mp_cmd_reset_n_2	Reset Input
	mp_rfifo_clk_0	mp_rfifo_clk_0	Clock Input
	mp_rfifo_reset_n_0	mp_rfifo_reset_n_0	Reset Input
	mp_wfifo_clk_0	mp_wfifo_clk_0	Clock Input
	mp_wfifo_reset_n_0	mp_wfifo_reset_n_0	Reset Input
	mp_rfifo_clk_1	mp_rfifo_clk_1	Clock Input
	mp_rfifo_reset_n_1	mp_rfifo_reset_n_1	Reset Input
	mp_rfifo_clk_2	mp_rfifo_clk_2	Clock Input
	mp_rfifo_reset_n_2	mp_rfifo_reset_n_2	Reset Input
	mp_wfifo_clk_2	mp_wfifo_clk_2	Clock Input
	mp_wfifo_reset_n_2	mp_wfifo_reset_n_2	Reset Input
	status	status	Conduit
	oct	oct	Conduit

## Hard Memory Controller

The hard memory controller initializes, refreshes, manages, and communicates with the external memory device.

**Note:** The hard memory controller is functionally similar to the High Performance Controller II (HPC II). For information on signals, refer to the *Functional Description—HPC II Controller* chapter.

**Related Information**

[Functional Description—HPC II Controller](#) on page 6-1

## Clocking

The ports on the MPFE can be clocked at different frequencies, and synchronization is maintained by cross-domain clocking logic in the MPFE.

Command ports can connect to different clocks, but the data ports associated with a given command port must be attached to the same clock as that command port. For example, a bidirectional command port that performs a 64-bit read/write function has its read port and write port connected to the same clock as the command port. Note that these clocks are separate from the EMIF core generated clocks.

## Reset

The ports of the MPFE must connect to the same reset signal.

When the reset signal is asserted, it resets the command and data FIFO buffer in the MPFE without resetting the hard memory controller.

**Note:** The `global_reset_n` and `soft_reset_n` signals are asynchronous.

For easiest management of reset signals, Altera recommends the following sequence at power-up:

1. Initially `global_reset_n`, `soft_reset_n`, and the MPFE reset signals are all asserted.
2. `global_reset_n` is deasserted.
3. Wait for `pll_locked` to transition high.
4. `soft_reset_n` is deasserted.
5. (Optional) If you encounter difficulties, wait for the controller signal `local_cal_success` to go high, indicating that the external memory interface has successfully completed calibration, before deasserting the MPFE FIFO reset signals. This will ensure that read/write activity cannot occur until the interface is successfully calibrated.

## DRAM Interface

The DRAM interface is 40 bits wide, and can accommodate 8-bit, 16-bit, 16-bit plus ECC, 32-bit, or 32-bit plus ECC configurations. Any unused I/Os in the DRAM interface can be reused as user I/Os.

The DRAM interface supports DDR2 and DDR3 memory protocols, and LPDDR2 for Cyclone V only. Fast and medium speed grade devices are supported to 533 MHz for Arria V and 400 MHz for Cyclone V.

## ECC

The hard controller supports both error-correcting code (ECC) calculated by the controller and by the user. Controller ECC code employs standard Hamming logic which can detect and correct single-bit errors and detect double-bit errors. The controller ECC is available for 16-bit and 32-bit widths, each requiring an additional 8 bits of memory, resulting in an actual memory width of 24-bits and 40-bits, respectively.

In user ECC mode, all bits are treated as data bits, and are written to and read from memory. User ECC can implement nonstandard memory widths such as 24-bit or 40-bit, where ECC is not required.

## Controller ECC

Controller ECC provides the following features:

**Byte Writes**—The memory controller performs a read/modify/write operation to keep ECC valid when a subset of the bits of a word is being written. If an entire word is being written (but less than a full burst) and the DM pins are connected, no read is necessary and only that word is updated. If controller ECC is disabled, byte-writes have no performance impact.

**ECC Write Backs**—When a read operation detects a correctable error, the memory location is scheduled for a read/modify/write operation to correct the single-bit error.

**User ECC**—User ECC is 24-bits or 40-bits wide; with user ECC, the controller performs no ECC checking. The controller employs memory word addressing with byte enables, and can handle arbitrary memory widths. User ECC does not disable byte writes; hence, you must ensure that any byte writes do not result in corrupted ECC.

## Bonding of Memory Controllers

Bonding is a feature that allows data to be split between two memory controllers, providing the ability to service bandwidth streams similar to a single 64-bit controller. Bonding works by dividing data buses in proportion to the memory widths, and always sending a transaction to both controllers. When signals are returned, bonding ensures that both sets of signals are returned identically.

Bonding can be applied to asymmetric controllers, and allows controllers to have different memory clocks. Bonding does not attempt to synchronize the controllers. Bonding supports only one port. The Avalon port width can be varied from 64-bit to 256-bit; 32-bit port width is not supported.

The following signals require bonding circuitry:

**Read data return**—This bonding allows read data from the two controllers to return with effectively one ready signal to the bus master that initiated the bus transaction.

**Write ready**—For Avalon-MM, this is effectively bonding on the `waitrequest` signal.

**Write acknowledge**—Synchronization on returning the write completed signal.

For each of the above implementations, data is returned in order, hence the circuitry must match up for each valid cycle.

Bonded FIFO buffers must have identical FIFO numbers; that is, read FIFO 1 on controller 1 must be paired with Read FIFO 1 on controller 2.

### Data Return Bonding

Long loop times can lead to communications problems when using bonded controllers. The following effects are possible when using bonded controllers:

- If one memory controller completes its transaction and receives new data before the other controller, then the second controller can send data as soon as it arrives, and before the first controller acknowledges that the second controller has data.
- If the first controller has a single word in its FIFO buffer and the second controller receives single-word transactions, the second controller must determine whether the second word is a valid signal or not.

To accommodate the above effects, the hard controller maintains two counters for each bonded pair of FIFO buffers and implements logic that monitors those counters to ensure that the bonded controllers receive the same data on the same cycle, and that they send the data out on the same cycle.

## FIFO Ready

FIFO ready bonding is used for write command and write data buses. The implementation is similar to the data return bonding.

## Bonding Latency Impact

Bonding has no latency impact on ports that are not bonded.

## Bonding Controller Usage

Arria V and Cyclone V devices employ three shared bonding controllers to manage the read data return bonding, write acknowledge bonding, and command/write data ready bonding.

The three bonding controllers require three pairs of bonding I/Os, each based on a six port count; this means that a bonded hard memory controller requires 21 input signals and 21 output signals for its connection to the fabric, and another 21 input signals and 21 output signals to the paired hard memory controller.

**Note:** The hard processor system (HPS) hard memory controller cannot be bonded with another hard memory controller on the FPGA portion of the device.

## Bonding Configurations and Parameter Requirements

Altera has verified hard memory controller bonding between two interfaces with the following configuration:

- Same clock source
- Same memory clock frequency
- Same memory parameters and timings (except interface width)
- Same controller settings.
- Same port width in MPFE settings

Bonding supports only one port. The Avalon port width can be varied from 64-bits to 256-bits; a 32-bit port width is not supported.

## Hard PHY

A physical layer interface (PHY) is embedded in the periphery of the Arria V device, and can run at the same high speed as the hard controller and hard sequencer. This hard PHY is located next to the hard controller. Differing device configurations have different numbers and sizes of hard controller and hard PHY pairs.

The hard PHY implements logic that connects the hard controller to the I/O ports. Because the hard controller and AFI interface support high frequencies, a portion of the sequencer is implemented as hard logic. The Nios II processor, the instruction/data RAM, and the Avalon fabric of the sequencer are implemented as core soft logic. The read/write manager and PHY manager components of the sequencer, which must operate at full rate, are implemented as hard logic in the hard PHY.

## Interconnections

The hard PHY resides on the device between the hard controller and the I/O register blocks. The hard PHY is instantiated or bypassed entirely, depending on the parameterization that you specify.

The hard PHY connects to the hard memory controller and the core, enabling the use of either the hard memory controller or a software-based controller. (You can have the hard controller and hard PHY, or

the soft controller and soft PHY; however, the combination of soft controller with hard PHY is not supported.) The hard PHY also connects to the I/O register blocks and the DQS logic. The path between the hard PHY and the I/O register blocks can be bypassed, but not reconfigured—in other words, if you use the hard PHY datapath, the pins to which it connects are predefined and specified by the device pin table.

## Clock Domains

The hard PHY contains circuitry that uses the following clock domains:

**AFI clock domain (pll\_afi\_clk)** —The main full-rate clock signal that synchronizes most of the circuit logic.

**Avalon clock domain (pll\_avl\_clk)** —Synchronizes data on the internal Avalon bus, namely the Read/Write Manager, PHY Manager, and Data Manager data. The data is then transferred to the AFI clock domain. To ensure reliable data transfer between clock domains, the Avalon clock period must be an integer multiple of the AFI clock period, and the phases of the two clocks must be aligned.

**Address and Command clock domain (pll\_addr\_cmd\_clk)** —Synchronizes the global asynchronous reset signal, used by the I/Os in this clock domain.

## Hard Sequencer

The sequencer initializes the memory device and calibrates the I/Os, with the objective of maximizing timing margins and achieving the highest possible performance.

When the hard memory controller is in use, a portion of the sequencer must run at full rate; for this reason, the Read/Write Manager, PHY Manager, and Data Manager are implemented as hard components within the hard PHY. The hard sequencer communicates with the soft-logic sequencer components (including the Nios II processor) via an Avalon bus.

## MPFE Setup Guidelines

The following instructions provide information on configuring the multi-port front end of the hard memory interface.

1. To enable the hard memory interface, turn on **Enable Hard External Memory Interface** in the **Interface Type** tab in the parameter editor.
2. To export bonding interface ports to the top level, turn on **Export bonding interface** in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

**Note:** The system exports three bonding-in ports and three bonding-out ports. You must generate two controllers and connect the bonding ports manually.

3. To expand the interface data width from a maximum of 32 bits to a maximum of 40 bits, turn on **Enable Avalon-MM data for ECC** in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

**Note:** The controller does not perform ECC checking when this option is turned on.

4. Select the required **Number of ports** for the multi-port front end in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

**Note:** The maximum number of ports is 6, depending on the port type and width. The maximum port width is 256 bits, which is the maximum data width of the read data FIFO and write data FIFO buffers.

5. The table in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor lists the ports that are created. The columns in the table describe each port, as follows:

- **Port:** Indicates the port number.
- **Type:** Indicates whether the port is read only, write only, or bidirectional.
- **Width:** To achieve optimum MPFE throughput, Altera recommends setting the MPFE data port width according to the following calculation:

$2 \times (\text{frequency ratio of HMC to user logic}) \times (\text{interface data width})$

For example, if the frequency of your user logic is one-half the frequency of the hard memory controller, you should set the port width to be 4x the interface data width. If the frequency ratio of the hard memory controller to user logic is a fractional value, you should use a larger value; for example, if the ratio is 1.5, you can use 2.

- **Priority:** The priority setting specifies the priority of the slave port, with higher values representing higher priority. The slave port with highest priority is served first.
- **Weight:** The weight setting has a range of values of 0–31, and specifies the relative priority of a slave port, with higher weight representing higher priority. The weight value can determine relative bandwidth allocations for slave ports with the same priority values. For example, if two ports have the same priority value, and weight values of 4 and 6, respectively, the port with a weight of 4 will receive 40% of the bus bandwidth, while the port with a weight of 6 will receive 60% of the bus bandwidth—assuming 100% total available bus bandwidth.

## Soft Memory Interface to Hard Memory Interface Migration Guidelines

The following instructions provide information on mapping your soft memory interface to a hard memory interface.

### Pin Connections

1. The hard and soft memory controllers have compatible pinouts. Assign interface pins to the hard memory interface according to the pin table.
2. Ensure that your soft memory interface pins can fit into the hard memory interface. The hard memory interface can support a maximum of a 40-bit interface with user ECC, or a maximum of 80-bits with same-side bonding. The soft memory interface does not support bonding.
3. Follow the recommended board layout guidelines for the hard memory interface.

### Software Interface Preparation

Observe the following points in preparing your soft memory interface for migration to a hard memory interface:

- You cannot use the hard PHY without also using the hard memory controller.
- The hard memory interface supports only full-rate controller mode.
- Ensure that the MPFE data port width is set according to the soft memory interface half-rate mode Avalon data width.
- The hard memory interface uses a different Avalon port signal naming convention than the software memory interface. Ensure that you change the `avl_<signal_name>` signals in the soft memory interface to `.avl_<signal_name>.o` signals in the hard memory interface.
- The hard memory controller MPFE includes an additional three clocks and three reset ports (CMD port, RFIFO port, and WFIFO port) that do not exist with the soft memory controller. You should connect the user logic clock signal to the MPFE clock port, and the user logic reset signal to the MPFE reset port.
- In the soft memory interface, the half-rate `afi_clk` is a user logic clock. In the hard memory interface, `afi_clk` is a full-rate clock, because the core fabric might not be able to achieve full-rate speed. When you migrate your soft memory interface to a hard memory interface, you need to supply an additional slower rate clock. The maximum clock rate supported by core logic is one-half of the maximum interface frequency.

### Latency

Overall, you should expect to see slightly more latency when using the hard memory controller and multi-port front end, than when using the soft memory controller.

The hard memory controller typically exhibits lower latency than the soft memory controller; however, the multi-port front end does introduce additional latency cycles due to FIFO buffer stages used for synchronization. The MPFE cannot be bypassed, even if only one port is needed.

## Bonding Interface Guidelines

Bonding allows a single data stream to be split between two memory controllers, providing the ability to expand the interface data width similar to a single 64-bit controller. This section provides some guidelines for setting up the bonding interface.

1. Bonding interface ports are exported to the top level in your design. You should connect each `bonding_in*` port in one hard memory controller to the corresponding `bonding_out_*`port in the other hard memory controller, and vice versa.
2. You should modify the Avalon signal connections to drive the bonding interface with a single user logic/master, as follows:
  - a. AND both `avl_ready` signals from both hard memory controllers before the signals enter the user logic.
  - b. AND both `avl_rdata_valid` signals from both hard memory controllers before the signals enter the user logic. (The `avl_rdata_valid` signals should be identical for both hard memory controllers.)
  - c. Branch the following signals from the user logic to both hard memory controllers:
    - `avl_burstbegin`
    - `avl_addr`
    - `avl_read_req`
    - `avl_write_req`
    - `avl_size`
  - d. Split the following signals according to each multi-port front end data port width:
    - `avl_rdata`
    - `avl_wdata`
    - `avl_be`

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.

Date	Version	Changes
August 2014	2014.08.15	<ul style="list-style-type: none"> <li>Updated descriptions of <code>mp_cmd_reset_n</code>, <code>#_reset_n</code>, <code>mp_rfifo_reset_n</code>, <code>#_reset_n</code>, and <code>mp_wfifo_reset_n</code>, <code>#_reset_n</code> in the <i>MPFE Signals</i> table.</li> <li>Added <i>Reset</i> section to <i>Hard Memory Controller</i> section.</li> </ul>
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>Added footnote about command FIFOs to MPFE Signals table.</li> <li>Added information about FIFO depth for the MPFE.</li> <li>Added information about hard memory controller bonding.</li> <li>Reworded protocol-support information for Arria V and Cyclone V devices.</li> </ul>
November 2012	2.1	<ul style="list-style-type: none"> <li>Added <i>Hard Memory Interface Implementation Guidelines</i>.</li> <li>Moved content of EMI-Related HPS Features in SoC Devices section to chapter 4, <i>Functional Description—HPS Memory Controller</i>.</li> </ul>
June 2012	2.0	<ul style="list-style-type: none"> <li>Added EMI-Related HPS Features in SoC Devices.</li> <li>Added LPDDR2 support.</li> <li>Added Feedback icon.</li> </ul>
November 2011	1.0	Initial release.

# Functional Description—HPS Memory Controller

5

2016.05.02

EMI\_RM



Subscribe



Send Feedback

The hard processor system (HPS) SDRAM controller subsystem provides efficient access to external SDRAM for the ARM® Cortex®-A9 microprocessor unit (MPU) subsystem, the level 3 (L3) interconnect, and the FPGA fabric. The SDRAM controller provides an interface between the FPGA fabric and HPS. The interface accepts Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI™) and Avalon® Memory-Mapped (Avalon-MM) transactions, converts those commands to the correct commands for the SDRAM, and manages the details of the SDRAM access.

## Features of the SDRAM Controller Subsystem

The SDRAM controller subsystem offers programming flexibility, port and bus configurability, error correction, and power management for external memories up to 4 GB.

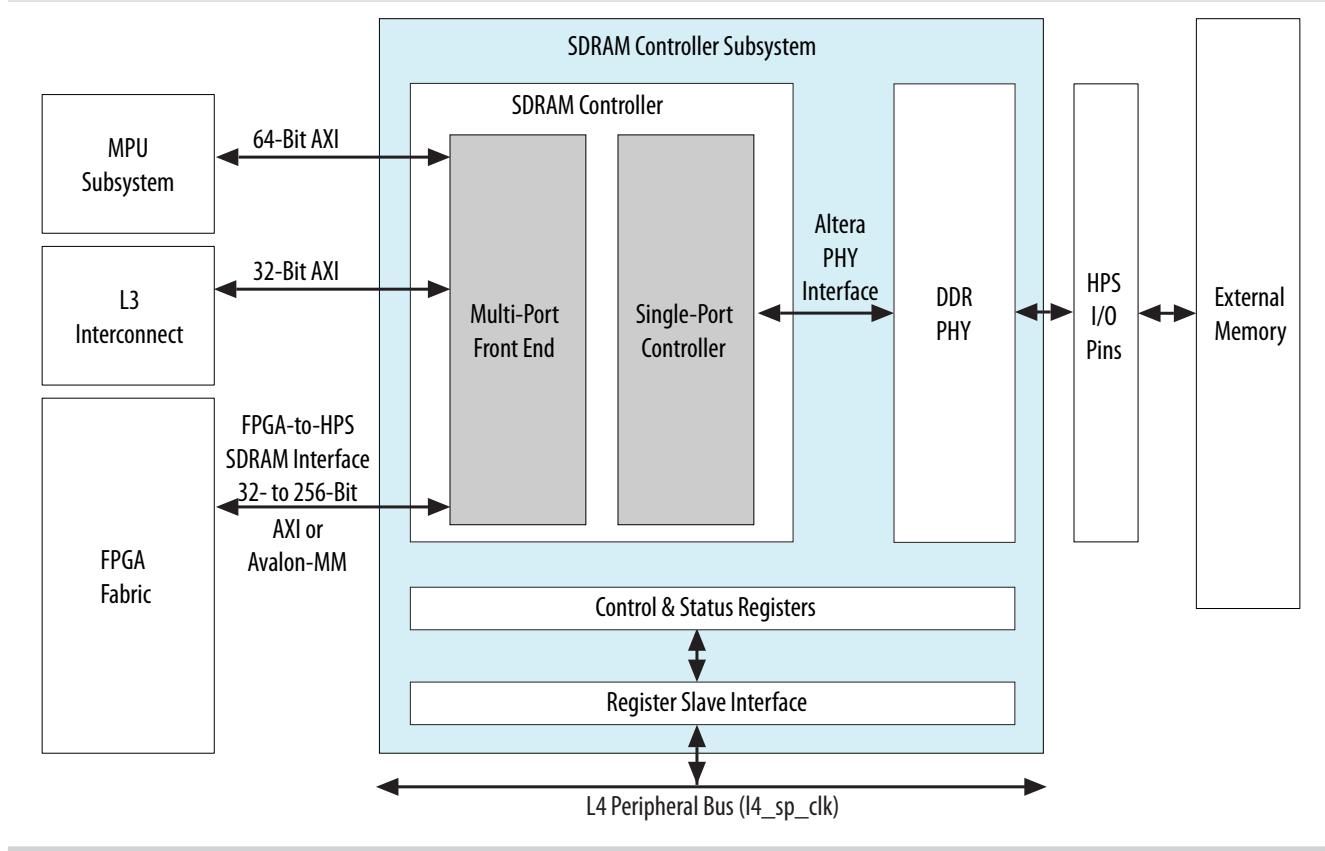
- Support for double data rate 2 (DDR2), DDR3, and low-power DDR2 (LPDDR2) SDRAM
- Flexible row and column addressing with the ability to support up to 4 GB of memory in various interface configurations
- Optional 8-bit integrated error correction code (ECC) for 16- and 32-bit data widths<sup>(3)</sup>
- User-configurable memory width of 8, 16, 16+ECC, 32, 32+ECC
- User-configurable timing parameters
- Two chip selects (DDR2 and DDR3)
- Command reordering (look-ahead bank management)
- Data reordering (out of order transactions)
- User-controllable bank policy on a per port basis for either closed page or conditional open page accesses
- User-configurable priority support with both absolute and weighted round-robin scheduling
- Flexible FPGA fabric interface with up to 6 ports that can be combined for a data width up to 256 bits using Avalon-MM and AXI interfaces
- Power management supporting self refresh, partial array self refresh (PASR), power down, and LPDDR2 deep power down

<sup>(3)</sup> The level of ECC support is package dependent.

## SDRAM Controller Subsystem Block Diagram

The SDRAM controller subsystem connects to the MPU subsystem, the L3 interconnect, and the FPGA fabric. The memory interface consists of the SDRAM controller, the physical layer (PHY), control and status registers (CSRs), and their associated interfaces.

**Figure 5-1: SDRAM Controller Subsystem High-Level Block Diagram**



### SDRAM Controller

The SDRAM controller provides high performance data access and run-time programmability. The controller reorders data to reduce row conflicts and bus turn-around time by grouping read and write transactions together, allowing for efficient traffic patterns and reduced latency.

The SDRAM controller consists of a multiport front end (MPFE) and a single-port controller. The MPFE provides multiple independent interfaces to the single-port controller. The single-port controller communicates with and manages each external memory device.

The MPFE FPGA-to-HPS SDRAM interface port has an asynchronous FIFO buffer followed by a synchronous FIFO buffer. Both the asynchronous and synchronous FIFO buffers have a read and write data FIFO depth of 8, and a command FIFO depth of 4. The MPU subsystem 64-bit AXI port and L3 interconnect 32-bit AXI port have asynchronous FIFO buffers with read and write data FIFO depth of 8, and command FIFO depth of 4.

## DDR PHY

The DDR PHY provides a physical layer interface for read and write memory operations between the memory controller and memory devices. The DDR PHY has dataflow components, control components, and calibration logic that handle the calibration for the SDRAM interface timing.

### Related Information

[Memory Controller Architecture](#) on page 5-6

## SDRAM Controller Memory Options

Bank selects, and row and column address lines can be configured to work with SDRAMs of various technology and density combinations.

**Table 5-1: SDRAM Controller Interface Memory Options**

Memory Type <sup>(4)</sup>	Mbits	Column Address Bit Width	Bank Select Bit Width	Row Address Bit Width	Page Size	MBytes
DDR2	256	10	2	13	1024	32
	512	10	2	14	1024	64
	1024 (1 Gb)	10	3	14	1024	128
	2048 (2 Gb)	10	3	15	1024	256
	4096 (4 Gb)	10	3	16	1024	512
DDR3	512	10	3	13	1024	64
	1024 (1 Gb)	10	3	14	1024	128
	2048 (2 Gb)	10	3	15	1024	256
	4096 (4 Gb)	10	3	16	1024	512

<sup>(4)</sup> For all memory types shown in this table, the DQ width is 8.

Memory Type <sup>(4)</sup>	Mbits	Column Address Bit Width	Bank Select Bit Width	Row Address Bit Width	Page Size	MBytes
LPDDR2	64	9	2	12	512	8
	128	10	2	12	1024	16
	256	10	2	13	1024	32
	512	11	2	13	2048	64
	1024 (1 Gb) - S2 <sup>(5)</sup>	11	2	14	2048	128
	1024 (1 Gb) - S4 <sup>(6)</sup>	11	3	13	2048	128
	2048 (2 Gb) - S2 <sup>(5)</sup>	11	2	15	2048	256
	2048 (2 Gb) - S4 <sup>(6)</sup>	11	3	14	2048	256
	4096 (4 Gb)	12	3	14	4096	512

## SDRAM Controller Subsystem Interfaces

### MPU Subsystem Interface

The SDRAM controller connects to the MPU subsystem with a dedicated 64-bit AXI interface, operating on the `mpu_12_ram_clk` clock domain.

### L3 Interconnect Interface

The SDRAM controller interfaces to the L3 interconnect with a dedicated 32-bit AXI interface, operating on the `l3_main_clk` clock domain.

#### Related Information

#### [System Interconnect](#)

<sup>(4)</sup> For all memory types shown in this table, the DQ width is 8.

<sup>(5)</sup> S2 signifies a 2n prefetch size

<sup>(6)</sup> S4 signifies a 4n prefetch size

## CSR Interface

The CSR interface connects to the level 4 (L4) bus and operates on the `14_sp_clk` clock domain. The MPU subsystem uses the CSR interface to configure the controller and PHY, for example setting the memory timing parameter values or placing the memory in a low power state. The CSR interface also provides access to the status registers in the controller and PHY.

## FPGA-to-HPS SDRAM Interface

The FPGA-to-HPS SDRAM interface provides masters implemented in the FPGA fabric access to the SDRAM controller subsystem in the HPS. The interface has three port types that are used to construct the following AXI or Avalon-MM interfaces:

- Command ports—issue read and/or write commands, and for receive write acknowledge responses
- 64-bit read data ports—receive data returned from a memory read
- 64-bit write data ports—transmit write data

The FPGA-to-HPS SDRAM interface supports six command ports, allowing up to six Avalon-MM interfaces or three AXI interfaces. Each command port can be used to implement either a read or write command port for AXI, or be used as part of an Avalon-MM interface. The AXI and Avalon-MM interfaces can be configured to support 32-, 64-, 128-, and 256-bit data.

**Table 5-2: FPGA-to-HPS SDRAM Controller Port Types**

Port Type	Available Number of Ports
Command	6
64-bit read data	4
64-bit write data	4

The FPGA-to-HPS SDRAM controller interface can be configured with the following characteristics:

- Avalon-MM interfaces and AXI interfaces can be mixed and matched as required by the fabric logic, within the bounds of the number of ports provided to the fabric.
- Because the AXI protocol allows simultaneous read and write commands to be issued, two SDRAM control ports are required to form an AXI interface.
- Because the data ports are natively 64-bit, they must be combined if wider data paths are required for the interface.
- Each Avalon-MM or AXI interface of the FPGA-to-HPS SDRAM interface operates on an independent clock domain.
- The FPGA-to-HPS SDRAM interfaces are configured during FPGA configuration.

The following table shows the number of ports needed to configure different bus protocols, based on type and data width.

**Table 5-3: FPGA-to-HPS SDRAM Port Utilization**

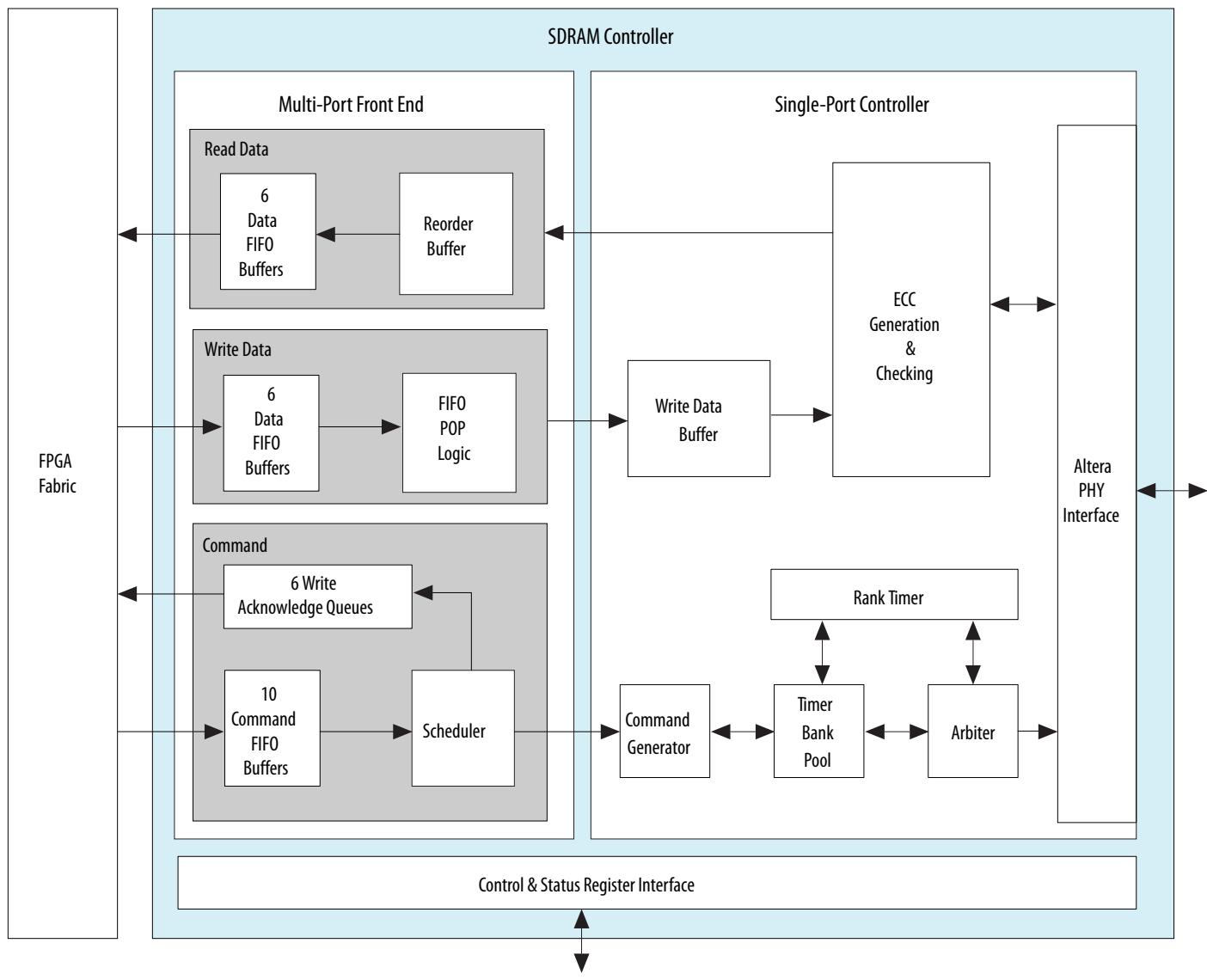
Bus Protocol	Command Ports	Read Data Ports	Write Data Ports
32- or 64-bit AXI	2	1	1

Bus Protocol	Command Ports	Read Data Ports	Write Data Ports
128-bit AXI	2	2	2
256-bit AXI	2	4	4
32- or 64-bit Avalon-MM	1	1	1
128-bit Avalon-MM	1	2	2
256-bit Avalon-MM	1	4	4
32- or 64-bit Avalon-MM write-only	1	0	1
128-bit Avalon-MM write-only	1	0	2
256-bit Avalon-MM write-only	1	0	4
32- or 64-bit Avalon-MM read-only	1	1	0
128-bit Avalon-MM read-only	1	2	0
256-bit Avalon-MM read-only	1	4	0

## Memory Controller Architecture

The SDRAM controller consists of an MPFE, a single-port controller, and an interface to the CSRs.

Figure 5-2: SDRAM Controller Block Diagram



## Multi-Port Front End

The Multi-Port Front End (MPFE) is responsible for scheduling pending transactions from the configured interfaces and sending the scheduled memory transactions to the single-port controller. The MPFE handles all functions related to individual ports.

The MPFE consists of three primary sub-blocks.

### Command Block

The command block accepts read and write transactions from the FPGA fabric and the HPS. When the command FIFO buffer is full, the command block applies backpressure by deasserting the ready signal. For each pending transaction, the command block calculates the next SDRAM burst needed to progress on that transaction. The command block schedules pending SDRAM burst commands based on the user-supplied configuration, available write data, and unallocated read data space.

### Write Data Block

The write data block transmits data to the single-port controller. The write data block maintains write data FIFO buffers and clock boundary crossing for the write data. The write data block informs the command block of the amount of pending write data for each transaction so that the command block can calculate eligibility for the next SDRAM write burst.

### Read Data Block

The read data block receives data from the single-port controller. Depending on the port state, the read data block either buffers the data in its internal buffer or passes the data straight to the clock boundary crossing FIFO buffer. The read data block reorders out-of-order data for Avalon-MM ports.

In order to prevent the read FIFO buffer from overflowing, the read data block informs the command block of the available buffer area so the command block can pace read transaction dispatch.

## Single-Port Controller

The single-port logic is responsible for following actions:

- Queuing the pending SDRAM bursts
- Choosing the most efficient burst to send next
- Keeping the SDRAM pipeline full
- Ensuring all SDRAM timing parameters are met

Transactions passed to the single-port logic for a single page in SDRAM are guaranteed to be executed in order, but transactions can be reordered between pages. Each SDRAM burst read or write is converted to the appropriate Altera PHY interface (AFI) command to open a bank on the correct row for the transaction (if required), execute the read or write command, and precharge the bank (if required).

The single-port logic implements command reordering (looking ahead at the command sequence to see which banks can be put into the correct state to allow a read or write command to be executed) and data reordering (allowing data transactions to be dispatched even if the data transactions are executed in an order different than they were received from the multi-port logic).

The single-port controller consists of eight sub-modules.

### Command Generator

The command generator accepts commands from the MPFE and from the internal ECC logic, and provides those commands to the timer bank pool.

#### Related Information

[Memory Controller Architecture](#) on page 5-6

For more information, refer to the SDRAM Controller Block diagram.

### Timer Bank Pool

The timer bank pool is a parallel queue that operates with the arbiter to enable data reordering. The timer bank pool tracks incoming requests, ensures that all timing requirements are met, and, on receiving write-data-ready notifications from the write data buffer, passes the requests to the arbiter.

#### Related Information

[Memory Controller Architecture](#) on page 5-6

For more information, refer to the SDRAM Controller Block diagram.

## Arbiter

The arbiter determines the order in which requests are passed to the memory device. When the arbiter receives a single request, that request is passed immediately. When multiple requests are received, the arbiter uses arbitration rules to determine the order to pass requests to the memory device.

### Related Information

[Memory Controller Architecture](#) on page 5-6

For more information, refer to the SDRAM Controller Block diagram.

## Rank Timer

The rank timer performs the following functions:

- Maintains rank-specific timing information
- Ensures that only four activates occur within a specified timing window
- Manages the read-to-write and write-to-read bus turnaround time
- Manages the time-to-activate delay between different banks

### Related Information

[Memory Controller Architecture](#) on page 5-6

For more information, refer to the SDRAM Controller Block diagram.

## Write Data Buffer

The write data buffer receives write data from the MPFE and passes the data to the PHY, on approval of the write request.

### Related Information

[Memory Controller Architecture](#) on page 5-6

For more information, refer to the SDRAM Controller Block diagram.

## ECC Block

The ECC block consists of an encoder and a decoder-corrector, which can detect and correct single-bit errors, and detect double-bit errors. The ECC block can correct single-bit errors and detect double-bit errors resulting from noise or other impairments during data transmission.

**Note:** The level of ECC support is package dependent.

### Related Information

[Memory Controller Architecture](#) on page 5-6

For more information, refer to the SDRAM Controller Block diagram.

## AFI Interface

The AFI interface provides communication between the controller and the PHY.

### Related Information

[Memory Controller Architecture](#) on page 5-6

For more information, refer to the SDRAM Controller Block diagram.

## CSR Interface

The CSR interface is accessible from the L4 bus. The interface allows code in the HPS MPU or soft IP cores in the FPGA fabric to configure and monitor the SDRAM controller.

### Related Information

[Memory Controller Architecture](#) on page 5-6

For more information, refer to the SDRAM Controller Block diagram.

# Functional Description of the SDRAM Controller Subsystem

## MPFE Operation Ordering

Operation ordering is defined and enforced within a port, but not between ports. All transactions received on a single port for overlapping addresses execute in order. Requests arriving at different ports have no guaranteed order of service, except when a first transaction has completed before the second arrives.

Avalon-MM does not support write acknowledgement. When a port is configured to support Avalon-MM, you should read from the location that was previously written to ensure that the write operation has completed. When a port is configured to support AXI, the master accessing the port can safely issue a read operation to the same address as a write operation as soon as the write has been acknowledged. To keep write latency low, writes are acknowledged as soon as the transaction order is guaranteed—meaning that any operations received on any port to the same address as the write operation are executed after the write operation.

To reduce read latency, the single-port logic can return read data out of order to the multi-port logic. The returned data is rearranged to its initial order on a per port basis by the multi-port logic and no traffic reordering occurs between individual ports.

### Read Data Handling

The MPFE contains a read buffer shared by all ports. If a port is capable of receiving returned data then the read buffer is bypassed. If the size of a read transaction is smaller than twice the memory interface width, the buffer RAM cannot be bypassed. The lowest memory latency occurs when the port is ready to receive data and the full width of the interface is utilized.

## MPFE Multi-Port Arbitration

The HPS SDRAM controller multi-port front end (MPFE) contains a programmable arbiter. The arbiter decides which MPFE port gains access to the single-port memory controller.

The SDRAM transaction size that is arbitrated is a burst of two beats. This burst size ensures that the arbiter does not favor one port over another when the incoming transaction size is a large burst.

The arbiter makes decisions based on two criteria: priority and weight. The priority is an absolute arbitration priority where the higher priority ports always win arbitration over the lower priority ports. Because multiple ports can be set to the same priority, the weight value refines the port choice by implementing a round-robin arbitration among ports set to the same priority. This programmable weight allows you to assign a higher arbitration value to a port in comparison to others such that the highest weighted port receives more transaction bandwidth than the lower weighted ports of the same priority.

Before arbitration is performed, the MPFE buffers are checked for any incoming transactions. The priority of each port that has buffered transactions is compared and the highest priority wins. If multiple ports are of the same highest priority value, the port weight is applied to determine which port wins. Because the arbiter only allows SDRAM-sized bursts into the single-port memory controller, large

transactions may need to be serviced multiple times before the read or write command is fully accepted to the single-port memory controller. The MPFE supports dynamic tuning of the priority and weight settings for each port, with changes committed into the SDRAM controller at fixed intervals of time.

Arbitration settings are applied to each port of the MPFE. The memory controller supports a mix of Avalon-MM and AXI protocols. As defined in the "Port Mappings" section, the Avalon-MM ports consume a single command port while the AXI ports consume a pair of command ports to support simultaneous read and write transactions. In total, there are ten command ports for the MPFE to arbitrate. The following table illustrates the command port mapping within the HPS as well as the ports exposed to the FPGA fabric.

**Table 5-4: HPS SDRAM MPFE Command Port Mapping**

Command Port	Allowed Functions	Data Size
0, 2, 4	FPGA fabric AXI read command ports FPGA fabric Avalon-MM read or write command ports	
1, 3, 5	FPGA fabric AXI write command ports FPGA fabric Avalon-MM read or write command ports	32-bit to 256-bit data
6	L3 AXI read command port	32-bit data
7	MPU AXI read command port	64-bit data
8	L3 AXI write command port	32-bit data
9	MPU AXI write command port	64-bit data

When the FPGA ports are configured for AXI, the command ports are always assigned in groups of two starting with even number ports 0, 2, or 4 assigned to the read command channel. For example, if you configure the first FPGA-to-SDRAM port as AXI and the second port as Avalon-MM, you can expect the following mapping:

- Command port 0 = AXI read
- Command port 1 = AXI write
- Command port 2 = Avalon-MM read and write

### Setting the MPFE Priority

The priority of each of the ten command ports is configured through the `userpriority` field of the `mppriority` register. This 30-bit register uses 3 bits per port to configure the priority. The lowest priority is 0x0 and the highest priority is 0x7. The bits are mapped in ascending order with bits [2:0] assigned to command port 0 and bits [29:27] assigned to command port 9.

## Setting the MPFE Static Weights

The static weight settings used in the round-robin command port priority scheme are programmed in a 128-bit field distributed among four 32-bit registers:

- mpweight\_0\_4
- mpweight\_1\_4
- mpweight\_2\_4
- mpweight\_3\_4

Each port is assigned a 5-bit value within the 128-bit field, such that port 0 is assigned to bits [4:0] of the mpweight\_0\_4 register, port 1 is assigned to bits [9:5] of the mpweight\_0\_4 register up to port 9, which is assigned to bits[49:45] contained in the mpweight\_1\_4 register. The valid weight range for each port is 0x0 to 0x1F, with larger static weights representing a larger arbitration share.

Bits[113:50] in the mpweight\_1\_4, mpweight\_2\_4 and mpweight\_3\_4 registers, hold the sum of weights for each priority. This 64-bit field is divided into eight fields of 8-bits, each representing the sum of static weights. Bits[113:50] are mapped in ascending order with bits [57:50] holding the sum of static weights for all ports with priority setting 0x0, and bits [113:106] holding the sum of static weights for all ports with priority setting 0x7.

## Example Using MPFE Priority and Weights

In this example, the following settings apply:

- FPGA MPFE port 0 is assigned to AXI read commands and port 1 is assigned to AXI write commands.
- FPGA MPFE port 2 is assigned to Avalon-MM read and write commands.
- L3 master ports (ports 6 and 8) and the MPU ports (port 7 and 9) are given the lowest priority but the MPU ports are configured with more arbitration static weight than the L3 master ports.
- The FPGA MPFE command ports are given the highest priority; however, AXI ports 0 and 1 are given a larger static weight because they carry the highest priority traffic in the entire system. Assigning a high priority and larger static weight ensures ports 0 and 1 will receive the highest quality-of-service (QoS).

The table below details the port weights and sum of weights.

**Table 5-5: SDRAM MPFE Port Priority, Weights and Sum of Weights**

Priority	Weights										Sum of Weights
-	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7	Port 8	Port 9	-
1	10	10	5	0	0	0	0	0	0	0	25
0	0	0	0	0	0	0	1	4	1	4	10

If the FPGA-to-SDRAM ports are configured according to the table and if both ports are accessed concurrently, you can expect the AXI port to receive 80% of the total service. This value is determined by taking the sum of port 0 and 1 weights divided by the total weight for all ports of priority 1. The remaining 20% of bandwidth is allocated to the Avalon-MM port. With these port settings, any FPGA transaction buffered by the MPFE for either slave port blocks the MPU and L3 masters from having their

buffered transactions serviced. To avoid transaction starvation, you should assign ports the same priority level and adjust the bandwidth allocated to each port by adjusting the static weights.

## MPFE Weight Calculation

The MPFE applies a deficit round-robin arbitration scheme to determine which port is serviced. The larger the port weight, the more often it is serviced. Ports are serviced only when they have buffered transactions and are set to the highest priority of all the ports that also have buffered transactions. The arbiter determines which port to service by examining the running weight of all the same ports at the same priority level and the largest running weight is serviced.

Each time a port is drained of all transactions, its running weight is set to 0x80. Each time a port is serviced, the static weight is added and the sum of weights is subtracted from the running weight for that port. Each time a port is not serviced (same priority as another port but has a lower running weight), the static weight for the port is added to the running weight of the port for that particular priority level. The running weight additions and subtractions are only applied to one priority level, so any time ports of a different priority level are being serviced, the running weight of a lower priority port is not modified.

## MPFE Multi-Port Arbitration Considerations for Use

When using MPFE multi-port arbitration, the following considerations apply:

- To ensure that the dynamic weight value does not roll over when a port is serviced, the following equation should always be true:

$$(\text{sum of weights} - \text{static weight}) < 128$$

If the running weight remains less than 128, arbitration for that port remains functional.

- The memory controller commits the priority and weight registers into the MPFE arbiter once every 10 SDRAM clock cycles. As a result, when the `mppriority` register and `mpweight_*_4` registers are configured at run time, the update interval can occur while the registers are still being written and ports can have different priority or weights than intended for a brief period. Because the `mppriority` and `mpweight_*_4` registers can be updated in a single 32-bit transaction, Altera recommends updating first to ensure that transactions that need to be serviced have the appropriate priority after the next update. Because the static weights are divided among four 32-bit registers
- In addition to the `mppriority` register and `mpweight_*_4` registers, the `remappriority` register adds another level of priority to the port scheduling. By programming bit N in the `priorityremap` field of the `remappriority` register, any port with an absolute priority N is sent to the front of the single port command queue and is serviced before any other transaction. Please refer to the `remappriority` register for more details.

The scheduler is work-conserving. Write operations can only be scheduled when enough data for the SDRAM burst has been received. Read operations can only be scheduled when sufficient internal memory is free and the port is not occupying too much of the read buffer.

## MPFE SDRAM Burst Scheduling

SDRAM burst scheduling recognizes addresses that access the same row/bank combination, known as open page accesses. Operations to a page are served in the order in which they are received by the single-port controller. Selection of SDRAM operations is a two-stage process. First, each pending transaction must wait for its timers to be eligible for execution. Next, the transaction arbitrates against other transactions that are also eligible for execution.

The following rules govern transaction arbitration:

- High-priority operations take precedence over lower-priority operations
- If multiple operations are in arbitration, read operations have precedence over write operations
- If multiple operations still exist, the oldest is served first

A high-priority transaction in the SDRAM burst scheduler wins arbitration for that bank immediately if the bank is idle and the high-priority transaction's chip select, row, or column fields of the address do not match an address already in the single-port controller. If the bank is not idle, other operations to that bank yield until the high-priority operation is finished. If the chip select, row, and column fields match an earlier transaction, the high-priority transaction yields until the earlier transaction is completed.

### Clocking

The FPGA fabric ports of the MPFE can be clocked at different frequencies. Synchronization is maintained by clock-domain crossing logic in the MPFE. Command ports can operate on different clock domains, but the data ports associated with a given command port must be attached to the same clock as that command port.

**Note:** A command port paired with a read and write port to form an Avalon-MM interface must operate at the same clock frequency as the data ports associated with it.

## Single-Port Controller Operation

The single-port controller increases the performance of memory transactions through command and data re-ordering, enforcing bank policies, combining write operations and allowing burst transfers. Correction of single-bit errors and detection of double-bit errors is handled in the ECC module of the single-port Controller.

### SDRAM Interface

The SDRAM interface is up to 40 bits wide and can accommodate 8-bit, 16-bit, 16-bit plus ECC, 32-bit, or 32-bit plus ECC configurations, depending on the device package. The SDRAM interface supports LPDDR2, DDR2, and DDR3 memory protocols.

### Command and Data Reordering

The heart of the SDRAM controller is a command and data reordering engine. Command reordering allows banks for future transactions to be opened before the current transaction finishes.

Data reordering allows transactions to be serviced in a different order than they were received when that new order allows for improved utilization of the SDRAM bandwidth. Operations to the same bank and row are performed in order to ensure that operations which impact the same address preserve the data integrity.

The following figure shows the relative timing for a write/read/write/read command sequence performed in order and then the same command sequence performed with data reordering. Data reordering allows the write and read operations to occur in bursts, without bus turnaround timing delay or bank reassignment.

**Figure 5-3: Data Reordering Effect**

The SDRAM controller schedules among all pending row and column commands every clock cycle.

## Bank Policy

The bank policy of the SDRAM controller allows users to request that a transaction's bank remain open after an operation has finished so that future accesses do not delay in activating the same bank and row combination. The controller supports only eight simultaneously-opened banks, so an open bank might get closed if the bank resource is needed for other operations.

Open bank resources are allocated dynamically as SDRAM burst transactions are scheduled. Bank allocation is requested automatically by the controller when an incoming transaction spans multiple SDRAM bursts or by the extended command interface. When a bank must be reallocated, the least-recently-used open bank is used as the replacement.

If the controller determines that the next pending command will cause the bank request to not be honored, the bank might be held open or closed depending on the pending operation. A request to close a bank with a pending operation in the timer bank pool to the same row address causes the bank to remain open. A request to leave a bank open with a pending command to the same bank but a different row address causes a precharge operation to occur.

## Write Combining

The SDRAM controller combines write operations from successive bursts on a port where the starting address of the second burst is one greater than the ending address of the first burst and the resulting burst length does not overflow the 11-bit burst-length counters.

Write combining does not occur if the previous bus command has finished execution before the new command has been received.

## Burst Length Support

The controller supports burst lengths of 2, 4, 8, and 16. Data widths of 8, 16, and 32 bits are supported for non-ECC operation and data widths of 24 and 40 bits are supported for operations with ECC enabled. The following table shows the type of SDRAM for each burst length.

**Table 5-6: SDRAM Burst Lengths**

Burst Length	SDRAM
4	LPDDR2, DDR2
8	DDR2, DDR3, LPDDR2

Burst Length	SDRAM
16	LPDDR2

### Width Matching

The SDRAM controller automatically performs data width conversion.

### ECC

The single-port controller supports memory ECC calculated by the controller.

The controller ECC employs standard Hamming logic to detect and correct single-bit errors and detect double-bit errors. The controller ECC is available for 16-bit and 32-bit widths, each requiring an additional 8 bits of memory, resulting in an actual memory width of 24-bits and 40-bits, respectively.

**Note:** The level of ECC support is package dependent.

Functions the controller ECC provides are:

- Byte Writes
- ECC Write Backs
- Notification of ECC Errors

### Byte Writes

The memory controller performs a read-modify-write operation to ensure that the ECC data remains valid when a subset of the bits of a word is being written.

Byte writes with ECC enabled are executed as a read-modify-write. Typical operations only use a single entry in the timer bank pool. Controller ECC enabled sub-word writes use two entries. The first operation is a read and the second operation is a write. These two operations are transferred to the timer bank pool with an address dependency so that the write cannot be performed until the read data has returned. This approach ensures that any subsequent operations to the same address (from the same port) are executed after the write operation, because they are ordered on the row list after the write operation.

If an entire word is being written (but less than a full burst) and the DM pins are connected, no read is necessary and only that word is updated. If controller ECC is disabled, byte-writes have no performance impact.

### ECC Write Backs

If the controller ECC is enabled and a read operation results in a correctable ECC error, the controller corrects the location in memory, if write backs are enabled. The correction results in scheduling a new read-modify-write.

A new read is performed at the location to ensure that a write operation modifying the location is not overwritten. The actual ECC correction operation is performed as a read-modify-write operation. ECC write backs are enabled and disabled through the `cfg_enable_ecc_code_overwrites` field in the `ctrlcfg` register.

**Note:** Double-bit errors do not generate read-modify-write commands. Instead, double-bit error address and count are reported through the `erraddr` and `dbecount` registers, respectively. In addition, a double-bit error interrupt can be enabled through the `dramintr` register.

## User Notification of ECC Errors

When an ECC error occurs, an interrupt signal notifies the MPU subsystem, and the ECC error information is stored in the status registers. The memory controller provides interrupts for single-bit and double-bit errors.

The status of interrupts and errors are recorded in status registers, as follows:

- The `dramsts` register records interrupt status.
- The `dramintr` register records interrupt masks.
- The `sbecount` register records the single-bit error count.
- The `dbecount` register records the double-bit error count.
- The `erraddr` register records the address of the most recent error.

For a 32-bit interface, ECC is calculated across a span of 8 bytes, meaning the error address is a multiple of 8 bytes (4-bytes\*2 burst length). To find the byte address of the word that contains the error, you must multiply the value in the `erraddr` register by 8.

### Related Information

#### [Cortex-A9 Microprocessor Unit Subsystem](#)

Information about ECC error interrupts

## Interleaving Options

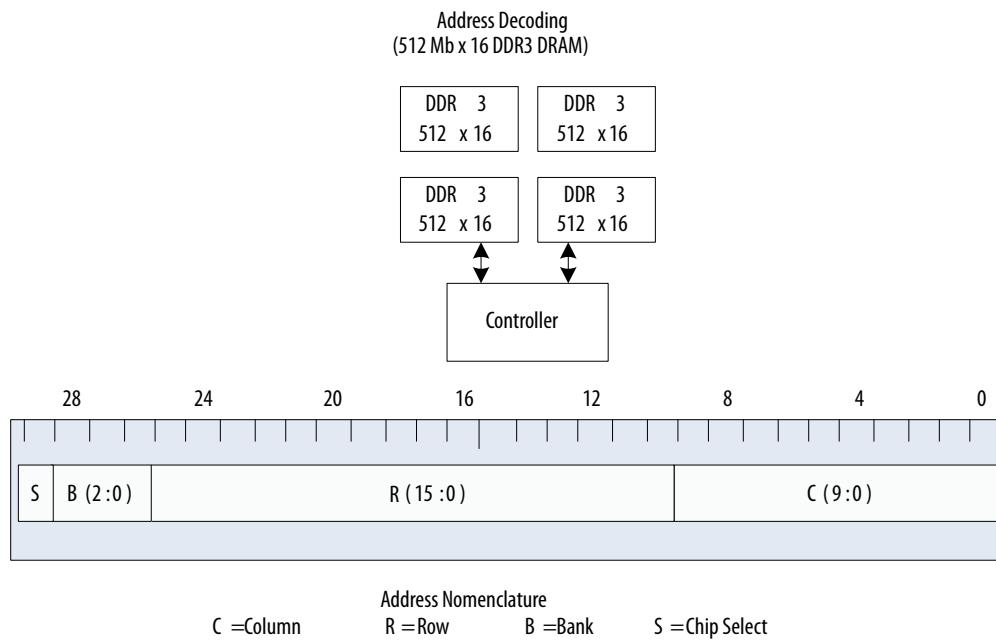
The controller supports the following address-interleaving options:

- Non-interleaved
- Bank interleave without chip select interleave
- Bank interleave with chip select interleave

The following interleaving examples use 512 megabits (Mb) x 16 DDR3 chips and are documented as byte addresses. For RAMs with smaller address fields, the order of the fields stays the same but the widths may change.

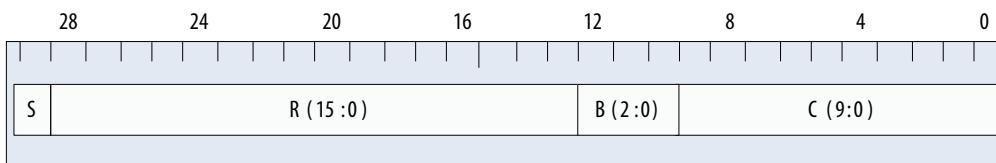
### Non-interleaved

RAM mapping is non-interleaved.

**Figure 5-4: Non-interleaved Address Decoding**

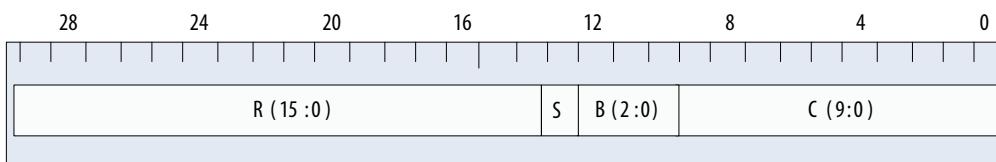
### Bank Interleave Without Chip Select Interleave

Bank interleave without chip select interleave swaps row and bank from the non-interleaved address mapping. This interleaving allows smaller data structures to spread across all banks in a chip.

**Figure 5-5: Bank Interleave Without Chip Select Interleave Address Decoding**

### Bank Interleave with Chip Select Interleave

Bank interleave with chip select interleave moves the row address to the top, followed by chip select, then bank, and finally column address. This interleaving allows smaller data structures to spread across multiple banks and chips (giving access to 16 total banks for multithreaded access to blocks of memory). Memory timing is degraded when switching between chips.

**Figure 5-6: Bank Interleave With Chip Select Interleave Address Decoding**

## AXI-Exclusive Support

The single-port controller supports AXI-exclusive operations. The controller implements a table shared across all masters, which can store up to 16 pending writes. Table entries are allocated on an exclusive read and table entries are deallocated on a successful write to the same address by any master.

Any exclusive write operation that is not present in the table returns an exclusive fail as acknowledgement to the operation. If the table is full when the exclusive read is performed, the table replaces a random entry.

**Note:** When using AXI-exclusive operations, accessing the same location from Avalon-MM interfaces can result in unpredictable results.

## Memory Protection

The single-port controller has address protection to allow the software to configure basic protection of memory from all masters in the system. If the system has been designed exclusively with AMBA masters, TrustZone® is supported. Ports that use Avalon-MM can be configured for port level protection.

Memory protection is based on physical addresses in memory. The single-port controller can configure up to 20 rules to allow or prevent masters from accessing a range of memory based on their AXIDs, level of security and the memory region being accessed. If no rules are matched in an access, then default settings take effect.

The rules are stored in an internal protection table and can be accessed through indirect addressing offsets in the `protruledwr` register in the CSR. To read a specific rule, set the `readrule` bit and write the appropriate offset in the `ruleoffset` field of the `protruledwr` register.

To write a new rule, three registers in the CSR must be configured:

1. The `protportdefault` register is programmed to control the default behavior of memory accesses when no rules match. When a bit is clear, all default accesses from that port pass. When a bit is set, all default accesses from that port fails. The bits are assigned as follows:

**Table 5-7: `protportdefault` register**

Bits	Description
31:10	reserved
9	When this bit is set to 1, deny CPU writes during a default transaction. When this bit is clear, allow CPU writes during a default transaction.
8	When this bit is set to 1, deny L3 writes during a default transaction. When this bit is clear, allow L3 writes during a default transaction.
7	When this bit is set to 1, deny CPU reads during a default transaction. When this bit is clear, allow CPU reads during a default transaction.
6	When this bit is set to 1, deny L3 reads during a default transaction. When this bit is clear, allow L3 reads during a default transaction.

Bits	Description
5 : 0	When this bit is set to 1, deny accesses from FPGA-to-SDRAM ports 0 through 5 during a default transaction. When this bit is clear, allow accesses from FPGA-to-SDRAM ports 0 through 5 during a default transaction.

2. The `protruleid` register gives the bounds of the AxID value that allows an access
3. The `protruledata` register configures the specific security characteristics for a rule.

Once the registers are configured, they can be committed to the internal protection table by programming the `ruleoffset` field and setting the `writerule` bit in the `protruledwr` register.

Secure and non-secure regions are specified by rules containing a starting address and ending address with 1 MB boundaries for both addresses. You can override the port defaults and allow or disallow all transactions.

The following table lists the fields that you can specify for each rule.

**Table 5-8: Fields for Rules in Memory Protection Table**

Field	Width	Description
Valid	1	Set to 1 to activate the rule. Set to 0 to deactivate the rule.
Port Mask <sup>(7)</sup>	10	Specifies the set of ports to which the rule applies, with one bit representing each port, as follows: bits 0 to 5 correspond to FPGA fabric ports 0 to 5, bit 6 corresponds to AXI L3 interconnect read, bit 7 is the CPU read, bit 8 is L3 interconnect write, and bit 9 is the CPU write.
AxID_low <sup>(7)</sup>	12	Low transfer AxID of the rules to which this rule applies. Incoming transactions match if they are greater than or equal to this value. Ports with smaller AxIDs have the AxID shifted to the lower bits and zero padded at the top.
AxID_high <sup>(7)</sup>	12	High transfer AxID of the rules to which this rule applies. Incoming transactions match if they are less than or equal to this value.
Address_low	12	Points to a 1MB block and is the lower address. Incoming addresses match if they are greater than or equal to this value.
Address_high	12	Upper limit of address. Incoming addresses match if they are less than or equal to this value.

<sup>(7)</sup> Although AxID and Port Mask could be redundant, including both in the table allows possible compression of rules. If masters connected to a port do not have contiguous AxIDs, a port-based rule might be more efficient than an AxID-based rule, in terms of the number of rules needed.

Field	Width	Description
Protection	2	A value of 0x0 indicates that the rule applies to secure transactions; a value of 0x1 indicates the rule applies to non-secure transactions. Values 0x2 and 0x3 set the region to shared, meaning both secure and non-secure accesses are valid.
Fail/allow	1	Set this value to 1 to force the operation to fail or succeed.

Each port has a default access status of either allow or fail. Rules with the opposite allow/fail value can override the default. The system evaluates each transaction against every rule in the memory protection table. If a transaction arrives at a port that defaults to access allowed, it fails only if a rule with the fail bit matches the transaction. Conversely, if a transaction arrives at a port that has the default rule set to access denied, it allows access only if there is a matching rule that forces accessed allowed. Transactions that fail the protection rules return a slave error (SLVERR).

The recommended sequence for writing a rule is:

1. Write the `protruledwr` register fields as follows:

- `ruleoffset` = offset selected by user that points to indirect offset in an internal protection table..
- `writerule` = 0
- `readrule` = 0

2. Write the `proruleaddr`, `proruleid`, and `proruledata` registers so you configure the rule you would like to enforce.

3. Write the `protruledwr` register fields as follows:

- `ruleoffset` = offset of the rule that needs to be written
- `writerule` = 1
- `readrule` = 0

Similarly, the recommended sequence for reading a rule is:

1. Write the `protruledwr` register fields as follows:

- `ruleoffset` = offset of the rule that needs to be written
- `writerule` = 0
- `readrule` = 0

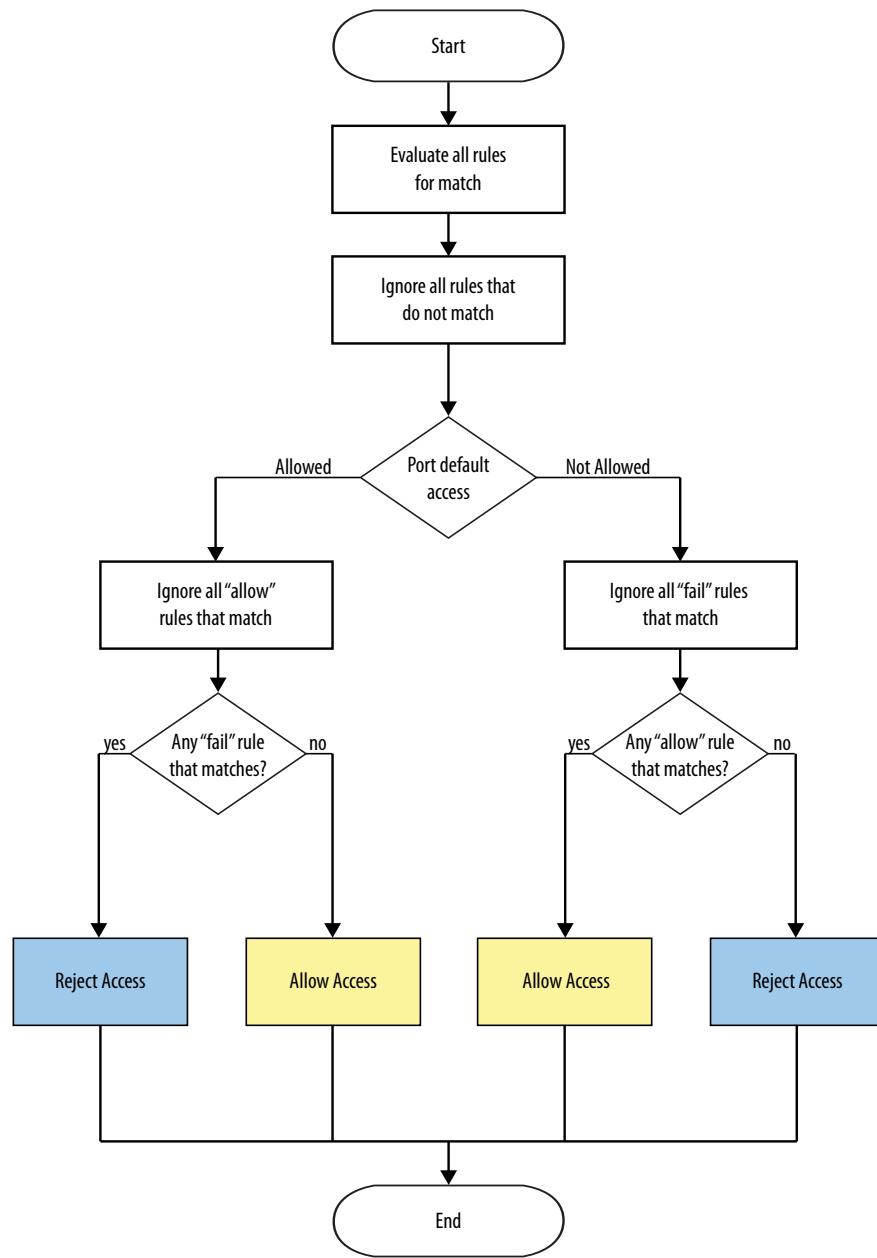
2. Write the `protruledwr` register fields as follows:

- `ruleoffset` = offset of the rule that needs to be read
- `writerule` = 0
- `readrule` = 1

3. Read the values of the `proruleaddr`, `proruleid`, and `proruledata` registers to determine the rule parameters.

The following figure represents an overview of how the protection rules are applied. There is no priority among the 20 rules. All rules are always evaluated in parallel.

Figure 5-7: SDRAM Protection Access Flow Diagram



Exclusive transactions are security checked on the read operation only. A write operation can occur only if a valid read is marked in the internal exclusive table. Consequently, a master performing an exclusive read followed by a write, can write to memory only if the exclusive read was successful.

#### Related Information

##### ARM TrustZone®

For more information about TrustZone® refer to the ARM web page.

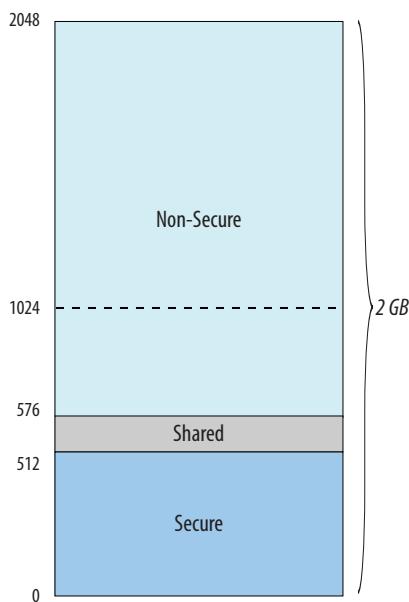
## Example of Configuration for TrustZone

For a TrustZone® configuration, memory is TrustZone divided into a range of memory accessible by secure masters and a range of memory accessible by non-secure masters. The two memory address ranges may have a range of memory that overlaps.

This example implements the following memory configuration:

- 2 GB total RAM size
- 0—512 MB dedicated secure area
- 513—576 MB shared area
- 577—2048 MB dedicated non-secure area

**Figure 5-8: Example Memory Configuration**



In this example, each port is configured by default to disallow all accesses. The following table shows the two rules programmed into the memory protection table.

**Table 5-9: Rules in Memory Protection Table for Example Configuration**

Rule #	Port Mask	AxID Low	AxID High	Address Low	Address High	protruledata.security	Fail/Allow
1	0x3FF (1023)	0x000	0xFFFF (4095)	0	576	0x1	Allow
2	0x3FF (1023)	0x000	0xFFFF (4095)	512	2047	0x0	Allow

The port mask value, AxID Low, and AxID High, apply to all ports and all transfers within those ports. Each access request is evaluated against the memory protection table, and will fail unless there is a rule match allowing a transaction to complete successfully.

**Table 5-10: Result for a Sample Set of Transactions**

Operation	Source	Address Accesses	Security Access Type	Result	Comments
Read	CPU	4096	secure	Allow	Matches rule 1.
Write	CPU	536, 870, 912	secure	Allow	Matches rule 1.
Write	L3 attached masters	605, 028, 350	secure	Fail	Does not match rule 1 (out of range of the address field), does not match rule 2 (protection bit incorrect).
Read	L3 attached masters	4096	non-secure	Fail	Does not match rule 1 (AxPROT signal value wrong), does not match rule 2 (not in address range).
Write	CPU	536, 870, 912	non-secure	Allow	Matches rule 2.
Write	L3 attached masters	605, 028, 350	non-secure	Allow	Matches rule 2.

**Note:** If a master is using the Accelerator Coherency Port (ACP) to maintain cache coherency with the Cortex-A9 MPCore processor, then the address ranges in the rules of the memory protection table should be made mutually exclusive, such that the secure and non-secure regions do not overlap and any area that is shared is part of the non-secure region. This configuration prevents coherency issues from occurring.

## SDRAM Power Management

The SDRAM controller subsystem supports the following power saving features in the SDRAM:

- Partial array self-refresh (PASR)
- Power down
- Deep power down for LPDDR2

To enable self-refresh for the memories of one or both chip selects, program the `selfshreq` bit and the `sefrfshmask` bit in the `lowpwreq` register.

Power-saving mode initiates either due to a user command or from inactivity. The number of idle clock cycles after which a memory can be put into power-down mode is programmed through the `autopdycycles` field of the `lowpwrtiming` register.

Power-down mode forces the SDRAM burst-scheduling bank-management logic to close all banks and issue the power down command. The SDRAM automatically reactivates when an active SDRAM command is received.

To enable deep power down request for the LPDDR2 memories of one or both chip selects, program the `deepwrdnreq` bit and the `deepwrdnmask` field of the `lowpwreq` register.

Other power-down modes are performed only under user control.

## DDR PHY

The DDR PHY connects the memory controller and external memory devices in the speed critical command path.

The DDR PHY implements the following functions:

- Calibration—the DDR PHY supports the JEDEC-specified steps to synchronize the memory timing between the controller and the SDRAM chips. The calibration algorithm is implemented in software.
- Memory device initialization—the DDR PHY performs the mode register write operations to initialize the devices. The DDR PHY handles re-initialization after a deep power down.
- Single-data-rate to double-data-rate conversion.

## Clocks

All clocks are assumed to be asynchronous with respect to the `ddr_dqs_clk` memory clock. All transactions are synchronized to memory clock domain.

**Table 5-11: SDRAM Controller Subsystem Clock Domains**

Clock Name	Description
<code>ddr_dq_clk</code>	Clock for PHY
<code>ddr_dqs_clk</code>	Clock for MPFE, single-port controller, CSR access, and PHY
<code>ddr_2x_dqs_clk</code>	Clock for PHY that provides up to 2 times <code>ddr_dq_clk</code> frequency
<code>14_sp_clk</code>	Clock for CSR interface
<code>mpu_12_ram_clk</code>	Clock for MPU interface
<code>13_main_clk</code>	Clock for L3 interface
<code>f2h_sdram_clk[5:0]</code>	Six separate clocks used for the FPGA-to-HPS SDRAM ports to the FPGA fabric

In terms of clock relationships, the FPGA fabric connects the appropriate clocks to write data, read data, and command ports for the constructed ports.

### Related Information

#### [Clock Manager](#)

## Resets

The SDRAM controller subsystem supports a full reset (cold reset) and a warm reset. The SDRAM controller can be configured to preserve memory contents during a warm reset.

To preserve memory contents, the reset manager can request that the single-port controller place the SDRAM in self-refresh mode prior to issuing the warm reset. If self-refresh mode is enabled before the warm reset to preserve memory contents, the PHY and the memory timing logic is not reset, but the rest of the controller is reset.

### Related Information

#### [Reset Manager](#)

## Taking the SDRAM Controller Subsystem Out of Reset

When a cold or warm reset is issued in the HPS, the Reset Manager resets this module and holds it in reset until software releases it.

After the MPU boots up, it can deassert the reset signal by clearing the appropriate bits in the Reset Manager's corresponding reset trigger.

## Port Mappings

The memory interface controller has a set of command, read data, and write data ports that support AXI3, AXI4 and Avalon-MM. Tables are provided to identify port assignments and functions.

**Table 5-12: Command Port Assignments**

Command Port	Allowed Functions
0, 2, 4	FPGA fabric AXI read command ports FPGA fabric Avalon-MM read or write command ports
1, 3, 5	FPGA fabric AXI write command ports FPGA fabric Avalon-MM read or write command ports
6	L3 AXI read command port
7	MPU AXI read command port
8	L3 AXI write command port
9	MPU AXI write command port

**Table 5-13: Read Port Assignments**

Read Port	Allowed Functions
0, 1, 2, 3	64-bit read data from the FPGA fabric. When 128-bit data read ports are created, then read data ports 0 and 1 get paired as well as 2 and 3.
4	32-bit L3 read data port
5	64-bit MPU read data port

**Table 5-14: Write Port Assignments**

Write Port	Allowed Functions
0, 1, 2, 3	64-bit write data from the FPGA fabric. When 128-bit data write ports are created, then write data ports 0 and 1 get paired as well as 2 and 3.
4	32-bit L3 write data port
5	64-bit MPU write data port

## Initialization

The SDRAM controller subsystem has control and status registers (CSRs) which control the operation of the controller including DRAM type, DRAM timing parameters and relative port priorities. It also has a small set of bits which depend on the FPGA fabric to configure ports between the memory controller and the FPGA fabric; these bits are set for you when you configure your implementation using the HPS GUI in Qsys.

The CSRs are configured using a dedicated slave interface, which provides access to the registers. This region controls all SDRAM operation, MPFE scheduler configuration, and PHY calibration.

The FPGA fabric interface configuration is programmed into the FPGA fabric and the values of these register bits can be read by software. The ports can be configured without software developers needing to know how the FPGA-to-HPS SDRAM interface has been configured.

## FPGA-to-SDRAM Protocol Details

The following topics summarize signals for the Avalon-MM Bidirectional port, Avalon-MM Write Port, Avalon-MM Read Port, and AXI port.

**Note:** If your device has multiple FPGA hardware images, then the same FPGA-to-SDRAM port configuration should be used across all designs.

### Avalon-MM Bidirectional Port

The Avalon-MM bidirectional ports are standard Avalon-MM ports used to dispatch read and write operations.

Each configured Avalon-MM bidirectional port consists of the signals listed in the following table.

**Table 5-15: Avalon-MM Bidirectional Port Signals**

Name	Bit Width	Input/Output Direction	Function
clk	1	In	Clock for the Avalon-MM interface
read	1	In	Indicates read transaction <sup>(8)</sup>
write	1	In	Indicates write transaction <sup>(8)</sup>
address	32	In	Address of the transaction
readdata	32, 64, 128, or 256	Out	Read data return
readdatavalid	1	Out	Indicates the readdata signal contains valid data in response to a previous read request.
writedata	32, 64, 128, or 256	In	Write data for a transaction
byteenable	4, 8, 16, 32	In	Byte enables for each write byte lane
waitrequest	1	Out	Indicates need for additional cycles to complete a transaction
burstcount	11	In	Transaction burst length. The value of the maximum burstcount parameter must be a power of 2.

The read and write interfaces are configured to the same size. The byte-enable size scales with the data bus size.

#### Related Information

##### [Avalon Interface Specifications](#)

Information about the Avalon-MM protocol

#### Avalon-MM Write-Only Port

The Avalon-MM write-only ports are standard Avalon-MM ports used to dispatch write operations. Each configured Avalon-MM write port consists of the signals listed in the following table.

<sup>(8)</sup> The Avalon-MM protocol does not allow read and write transactions to be posted concurrently.

**Table 5-16: Avalon-MM Write-Only Port Signals**

Name	Bits	Direction	Function
reset	1	In	Reset
clk	1	In	Clock
write	1	In	Indicates write transaction
address	32	In	Address of the transaction
writedata	32, 64, 128, or 256	In	Write data for a transaction
byteenable	4, 8, 16, 32	In	Byte enables for each write byte
waitrequest	1	Out	Indicates need for additional cycles to complete a transaction
burstcount	11	In	Transaction burst length

**Related Information****Avalon Interface Specifications**

Information about the Avalon-MM protocol

**Avalon-MM Read Port**

The Avalon-MM read ports are standard Avalon-MM ports used only to dispatch read operations. Each configured Avalon-MM read port consists of the signals listed in the following table.

**Table 5-17: Avalon-MM Read Port Signals**

Name	Bits	Direction	Function
reset	1	In	Reset
clk	1	In	Clock
read	1	In	Indicates read transaction
address	32	In	Address of the transaction
readdata	32, 64, 128, or 256	Out	Read data return

Name	Bits	Direction	Function
readdatavalid	1	Out	Flags valid cycles for read data return
waitrequest	1	Out	Indicates the need for additional cycles to complete a transaction. Needed for read operations when delay is needed to accept the read command.
burstcount	11	In	Transaction burst length

### Related Information

#### Avalon Interface Specifications

Information about the Avalon-MM protocol

### AXI Port

The AXI port uses an AXI-3 interface. Each configured AXI port consists of the signals listed in the following table. Because the AXI protocol allows simultaneous read and write commands to be issued, two SDRAM control ports are required to form an AXI interface.

**Table 5-18: AXI Port Signals**

Name	Bits	Direction	Channel	Function
ARESETn	1	In	n/a	Reset
ACLK	1	In	n/a	Clock
AWID	4	In	Write address	Write identification tag
AWADDR	32	In	Write address	Write address
AWLEN	4	In	Write address	Write burst length
AWSIZE	3	In	Write address	Width of the transfer size
AWBURST	2	In	Write address	Burst type
AWLOCK	2	In	Write address	Lock type signal which indicates if the access is exclusive; valid values are 0x0 (normal access) and 0x1 (exclusive access)
AWCACHE	4	In	Write address	Cache policy type
AWPROT	3	In	Write address	Protection-type signal used to indicate whether a transaction is secure or non-secure
AWREADY	1	Out	Write address	Indicates ready for a write command
AWVALID	1	In	Write address	Indicates valid write command.
WID	4	In	Write data	Write data transfer ID
WDATA	32, 64, 128 or 256	In	Write data	Write data

Name	Bits	Direction	Channel	Function
WSTRB	4, 8, 16, 32	In	Write data	Byte-based write data strobe. Each bit width corresponds to 8 bit wide transfer for 32-bit wide to 256-bit wide transfer.
WLAST	1	In	Write data	Last transfer in a burst
WVALID	1	In	Write data	Indicates write data and strobes are valid
WREADY	1	Out	Write data	Indicates ready for write data and strobes
BID	4	Out	Write response	Write response transfer ID
BRESP	2	Out	Write response	Write response status
BVALID	1	Out	Write response	Write response valid signal
BREADY	1	In	Write response	Write response ready signal
ARID	4	In	Read address	Read identification tag
ARADDR	32	In	Read address	Read address
ARLEN	4	In	Read address	Read burst length
ARSIZE	3	In	Read address	Width of the transfer size
ARBURST	2	In	Read address	Burst type
ARLOCK	2	In	Read address	Lock type signal which indicates if the access is exclusive; valid values are 0x0 (normal access) and 0x1 (exclusive access)
ARCACHE	4	In	Read address	Lock type signal which indicates if the access is exclusive; valid values are 0x0 (normal access) and 0x1 (exclusive access)
ARPROT	3	In	Read address	Protection-type signal used to indicate whether a transaction is secure or non-secure
ARREADY	1	Out	Read address	Indicates ready for a read command
ARVALID	1	In	Read address	Indicates valid read command
RID	4	Out	Read data	Read data transfer ID
RDATA	32, 64, 128 or 256	Out	Read data	Read data
RRESP	2	Out	Read data	Read response status
RLAST	1	Out	Read data	Last transfer in a burst
RVALID	1	Out	Read data	Indicates read data is valid
RREADY	1	In	Read data	Read data channel ready signal

**Related Information****ARM AMBA Open Specification**

AMBA Open Specifications, including information about the AXI-3 interface

# SDRAM Controller Subsystem Programming Model

SDRAM controller configuration occurs through software programming of the configuration registers using the CSR interface.

## HPS Memory Interface Architecture

The configuration and initialization of the memory interface by the ARM processor is a significant difference compared to the FPGA memory interfaces, and results in several key differences in the way the HPS memory interface is defined and configured.

Boot-up configuration of the HPS memory interface is handled by the initial software boot code, not by the FPGA programmer, as is the case for the FPGA memory interfaces. The software is involved in defining the configuration of I/O ports which is used by the boot-up code, as well as timing analysis of the memory interface. Therefore, the memory interface must be configured with the correct PHY-level timing information. Although configuration of the memory interface in Qsys is still necessary, it is limited to PHY- and board-level settings.

## HPS Memory Interface Configuration

To configure the external memory interface components of the HPS, open the HPS interface by selecting the Arria V/Cyclone V Hard Processor System component in Qsys. Within the HPS interface, select the EMIF tab to open the EMIF parameter editor.

The EMIF parameter editor contains four additional tabs: PHY Settings, Memory Parameters, Memory Timing, and Board Settings. The parameters available on these tabs are similar to those available in the parameter editors for non-SoC device families.

There are significant differences between the EMIF parameter editor for the Hard Processor System and the parameter editors for non-SoC devices, as follows:

- Because the HPS memory controller is not configurable through the software, the Controller and Diagnostic tabs, which exist for non-SoC devices, are not present in the EMIF parameter editor for the hard processor system.
- Unlike the protocol-specific parameter editors for non-SoC devices, the EMIF parameter editor for the Hard Processor System supports multiple protocols, therefore there is an SDRAM Protocol parameter, where you can specify your external memory interface protocol. By default, the EMIF parameter editor assumes the DDR3 protocol, and other parameters are automatically populated with DDR3-appropriate values. If you select a protocol other than DDR3, change other associated parameter values appropriately.
- Unlike the memory interface clocks in the FPGA, the memory interface clocks for the HPS are initialized by the boot-up code using values provided by the configuration process. You may accept the values provided by UniPHY, or you may use your own PLL settings. If you choose to specify your own PLL settings, you must indicate that the clock frequency that UniPHY should use is the requested clock frequency, and not the achieved clock frequency calculated by UniPHY.

**Note:** The HPS does not support EMIF synthesis generation, compilation, or timing analysis. The HPS hard memory controller cannot be bonded with another hard memory controller on the FPGA portion of the device.

## HPS Memory Interface Simulation

Qsys provides a complete simulation model of the HPS memory interface controller and PHY, providing cycle-level accuracy, comparable to the simulation models for the FPGA memory interface.

The simulation model supports only the skip-cal simulation mode; quick-cal and full-cal are not supported. An example design is not provided. However, you can create a test design by adding the traffic generator component to your design using Qsys. Also, the HPS simulation model does not use external memory pins to connect to the DDR memory model; instead, the memory model is incorporated directly into the HPS SDRAM interface simulation modules. The memory instance incorporated into the HPS model is in the simulation model hierarchy at: **hps\_0/fpga\_interfaces/f2sdram/hps\_sdram\_inst/mem/**

Simulation of the FPGA-to-SDRAM interfaces requires that you first bring the interfaces out of reset, otherwise transactions cannot occur. Connect the H2F reset to the F2S port resets and add a stage to your testbench to assert and deassert the H2F reset in the HPS. Appropriate Verilog code is shown below:

```
initial
begin
    // Assert reset
    <base name>.hps.fpga_interfaces.h2f_reset_inst.reset_assert();
    // Delay
    #1
    // Deassert reset
    <base name>.hps.fpga_interfaces.h2f_reset_inst.reset_deassert();
end
```

## Generating a Preloader Image for HPS with EMIF

To generate a Preloader image for an HPS-based external memory interface, you must complete the following tasks:

- Create a Qsys project.
- Create a top-level file and add constraints.
- Create a Preloader BSP file.
- Create a Preloader image.

### Creating a Qsys Project

Before you can generate a preloader image, you must create a Qsys project, as follows:

1. On the **Tools** menu in the software, click **Qsys**.
2. Under **Component library**, expand **Embedded Processor System**, select **Hard Processor System** and click **Add**.
3. Specify parameters for the **FPGA Interfaces**, **Peripheral Pin Multiplexing**, and **HPS Clocks**, based on your design requirements.
4. On the **SDRAM** tab, select the SDRAM protocol for your interface.
5. Populate the necessary parameter fields on the **PHY Settings**, **Memory Parameters**, **Memory Timing**, and **Board Settings** tabs.
6. Add other Qsys components in your Qsys design and make the appropriate bus connections.
7. Save the Qsys project.
8. Click **Generate** on the **Generation** tab, to generate the Qsys design.

### Creating a Top-Level File and Adding Constraints

This topic describes adding your Qsys system to your top-level design and adding constraints to your design.

1. Add your Qsys system to your top-level design.
2. Add the IP files (**.qip**) generated in step 2, to your project.
3. Perform analysis and synthesis on your design.
4. Constrain your EMIF design by running the **<variation\_name>\_p0\_pin\_assignments.tcl** pin constraints script.
5. Add other necessary constraints—such as timing constraints, location assignments, and pin I/O standard assignments—for your design.
6. Compile your design to generate an SRAM object file (**.sof**) and the hardware handoff files necessary for creating a preloader image.

**Note:** You must regenerate the hardware handoff files whenever the HPS configuration changes; for example, due to changes in Peripheral Pin Multiplexing or I/O standard for HPS pins.

#### Related Information

##### [Altera SoC Embedded Design Suite User's Guide](#)

For more information on how to create a preloader BSP file and image.

## Debugging HPS SDRAM in the Preloader

To assist in debugging your design, tools are available at the preloader stage.

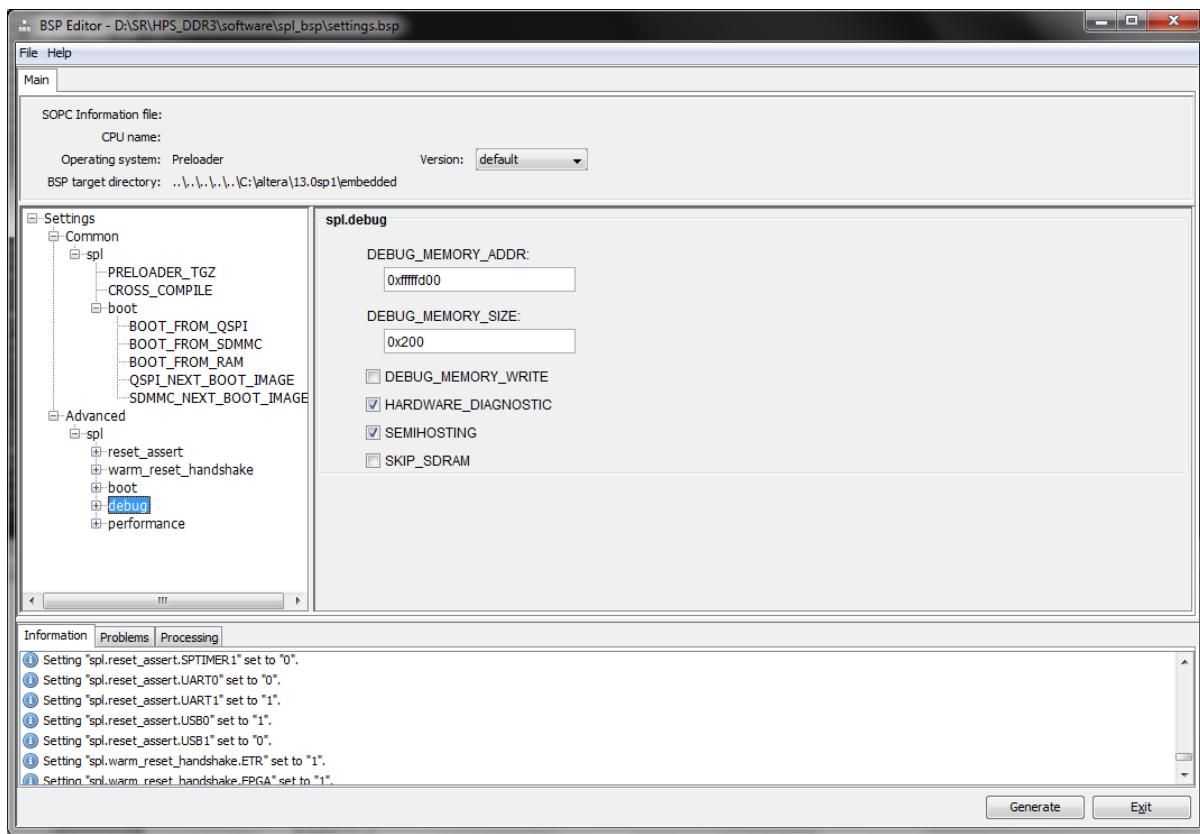
- UART or semihosting printout
- Simple memory test
- Debug report
- Predefined data patterns

The following topics provide procedures for implementing each of the above tools.

## Enabling UART or Semihosting Printout

UART printout is enabled by default. If UART is not available on your system, you can use semihosting together with the debugger tool. To enable semihosting in the Preloader, follow these steps:

1. When you create the **.bsp** file in the BSP Editor, select **SEMIHOSTING** in the **spl.debug** window.



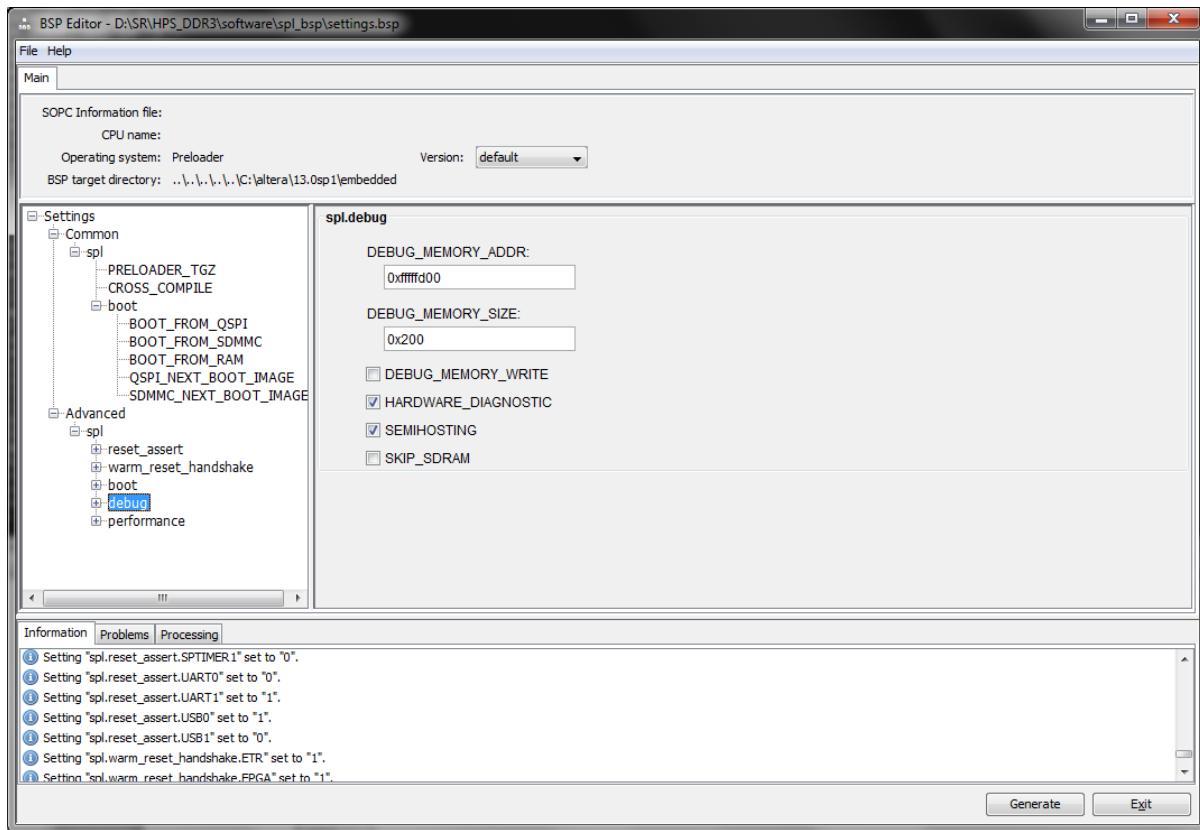
2. Enable semihosting in the debugger, by typing `set semihosting enabled true` at the command line in the debugger.

```
cd "C:\Users\kigtan\Documents\DS-5 Workspace_GHSD"
Working directory "C:\Users\kigtan\Documents\DS-5 Workspace_GHSD"
reset reset.system
Target has been reset
Execution stopped due to a breakpoint or watchpoint: S:0x00000000
S:0x00000000    LDR      pc,[pc,#24] ; [0x20] = 0xA8
loadfile "D:\SR\HPS_DDR3\software\spl_bsp\uboot-socfpga\splu-boot-spl"
Loaded section .text: S:0xFFFFF0000 ~ S:0xFFFFF8807 (size 0x8808)
Loaded section .rodata: S:0xFFFFF8808 ~ S:0xFFFFFA501 (size 0x1CFA)
Loaded section .data: S:0xFFFFFA508 ~ S:0xFFFFFB4E9F (size 0xFE8)
Loaded section .bss: S:0xFFFFB4A0 ~ S:0xFFFFFB593 (size 0xA4)
Loaded section .malloc: S:0xFFFFFB594 ~ S:0xFFFFCC993 (size 0x1400)
Loaded section .stack: S:0xFFFFFC994 ~ S:0xFFFFFD997 (size 0x1004)
Loaded section .spl_irq_stack: S:0xFFFFFD998 ~ S:0xFFFFE99F (size 0x1008)
Entry point S:0xFFFFF0000
set semihosting enabled true
Semihosting server socket created at port 8000
Command: set semihosting enabled true
```

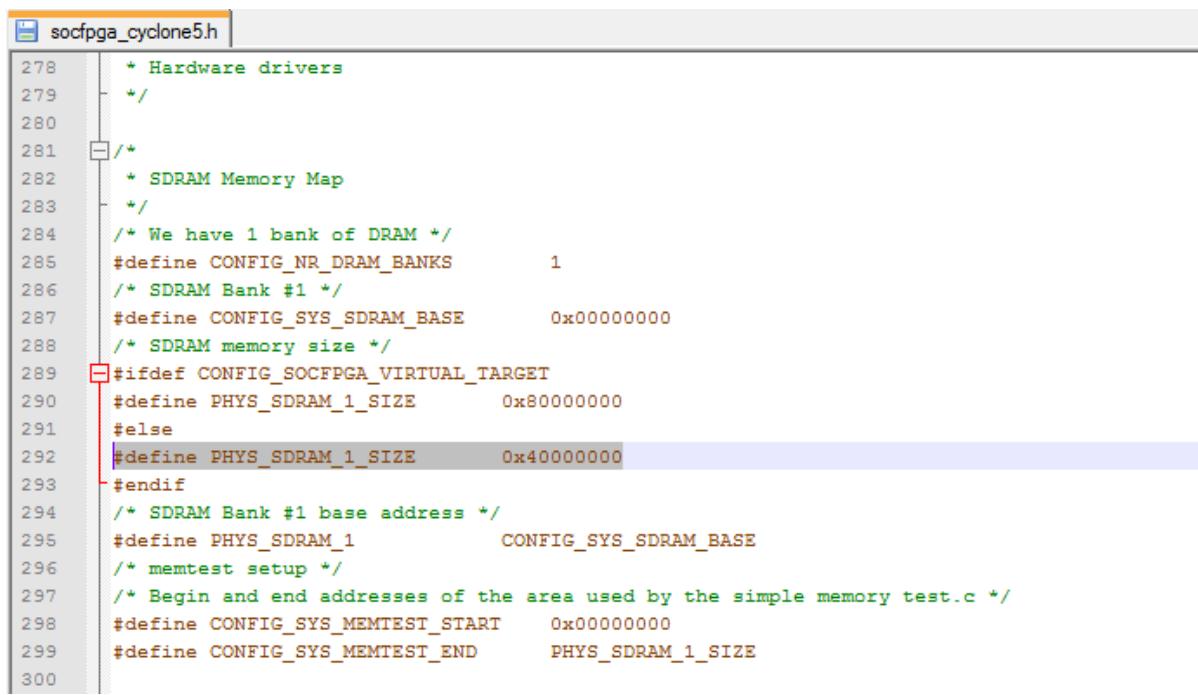
## Enabling Simple Memory Test

After the SDRAM is successfully calibrated, a simple memory test may be performed using the debugger.

1. When you create the **.bsp** file in the BSP Editor, select **HARDWARE\_DIAGNOSTIC** in the **spl.debug** window..



2. The simple memory test assumes SDRAM with a memory size of 1 GB. If your board contains a different SDRAM memory size, open the file <**design folder**>\spl\_bsp\uboot-socfpga\include\configs\socfpga\_cyclone5.h in a text editor, and change the PHYS\_SDRAM\_1\_SIZE parameter at line 292 to specify your actual memory size in bytes.



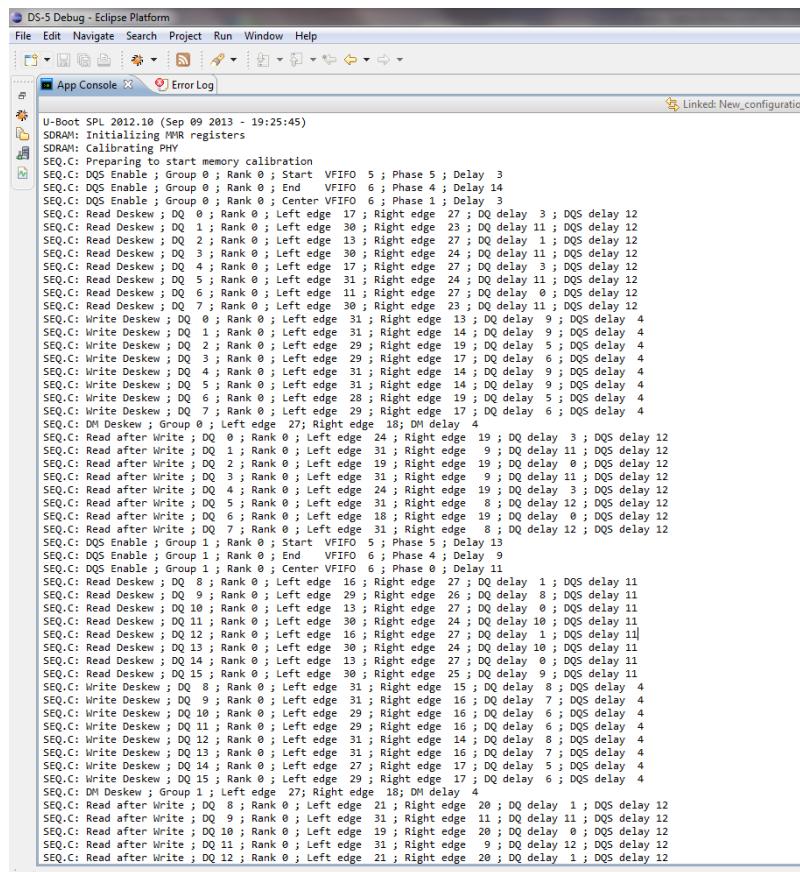
```
278 * Hardware drivers
279 */
280
281 /*
282 * SDRAM Memory Map
283 */
284 /* We have 1 bank of DRAM */
285 #define CONFIG_NR_DRAM_BANKS 1
286 /* SDRAM Bank #1 */
287 #define CONFIG_SYS_SDRAM_BASE 0x00000000
288 /* SDRAM memory size */
289 #ifdef CONFIG_SOCFPGA_VIRTUAL_TARGET
290 #define PHYS_SDRAM_1_SIZE 0x80000000
291 #else
292 #define PHYS_SDRAM_1_SIZE 0x40000000
293 #endif
294 /* SDRAM Bank #1 base address */
295 #define PHYS_SDRAM_1 CONFIG_SYS_SDRAM_BASE
296 /* memtest setup */
297 /* Begin and end addresses of the area used by the simple memory test.c */
298 #define CONFIG_SYS_MEMTEST_START 0x00000000
299 #define CONFIG_SYS_MEMTEST_END PHYS_SDRAM_1_SIZE
300
```

## Enabling the Debug Report

You can enable the SDRAM calibration sequencer to produce a debug report on the UART printout or semihosting output. To enable the debug report, follow these steps:

1. After you have enabled the UART or semihosting, open the file **<project directory>\hps\_isw\_handoff\sequencerDefines.hin** in a text editor.
2. Locate the line `#define RUNTIME_CAL_REPORT 0` and change it to `#define RUNTIME_CAL_REPORT 1`.

Figure 5-9: Semihosting Printout With Debug Support Enabled



```

DS-5 Debug - Eclipse Platform
File Edit Navigate Search Project Run Window Help
App Console Error Log
Linked: New_configuration

U-Boot SPL_2012.10 (Sep 09 2013 - 19:25:45)
SDRAM: Initializing MMR registers
SDRAM: Calibrating PHV
SEQ.C: Preparing to start memory calibration
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; Start VFIFO 5 ; Phase 5 ; Delay 3
SEQ.C: DQS Enable ; Group 0 ; End VFIFO 6 ; Phase 4 ; Delay 14
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; Center VFIFO 6 ; Phase 1 ; Delay 3
SEQ.C: Read Deskew ; DQ 0 ; Rank 0 ; Left edge 17 ; Right edge 27 ; DQ delay 3 ; DQS delay 12
SEQ.C: Read Deskew ; DQ 1 ; Rank 0 ; Left edge 30 ; Right edge 23 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read Deskew ; DQ 2 ; Rank 0 ; Left edge 13 ; Right edge 27 ; DQ delay 1 ; DQS delay 12
SEQ.C: Read Deskew ; DQ 3 ; Rank 0 ; Left edge 30 ; Right edge 24 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read Deskew ; DQ 4 ; Rank 0 ; Left edge 17 ; Right edge 27 ; DQ delay 3 ; DQS delay 12
SEQ.C: Read Deskew ; DQ 5 ; Rank 0 ; Left edge 31 ; Right edge 24 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read Deskew ; DQ 6 ; Rank 0 ; Left edge 11 ; Right edge 27 ; DQ delay 0 ; DQS delay 12
SEQ.C: Read Deskew ; DQ 7 ; Rank 0 ; Left edge 23 ; Right edge 11 ; DQ delay 11 ; DQS delay 12
SEQ.C: Write Deskew ; DQ 0 ; Rank 0 ; Left edge 31 ; Right edge 14 ; DQ delay 9 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 1 ; Rank 0 ; Left edge 31 ; Right edge 14 ; DQ delay 9 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 2 ; Rank 0 ; Left edge 29 ; Right edge 19 ; DQ delay 5 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 3 ; Rank 0 ; Left edge 29 ; Right edge 17 ; DQ delay 6 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 4 ; Rank 0 ; Left edge 31 ; Right edge 14 ; DQ delay 9 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 5 ; Rank 0 ; Left edge 31 ; Right edge 14 ; DQ delay 9 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 6 ; Rank 0 ; Left edge 28 ; Right edge 19 ; DQ delay 5 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 7 ; Rank 0 ; Left edge 29 ; Right edge 17 ; DQ delay 6 ; DQS delay 4
SEQ.C: DM Deskew ; Group 0 ; Left edge 27; Right edge 18; DM delay 4
SEQ.C: Read after Write ; DQ 0 ; Rank 0 ; Left edge 24 ; Right edge 19 ; DQ delay 3 ; DQS delay 12
SEQ.C: Read after Write ; DQ 1 ; Rank 0 ; Left edge 31 ; Right edge 9 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read after Write ; DQ 2 ; Rank 0 ; Left edge 19 ; Right edge 19 ; DQ delay 0 ; DQS delay 12
SEQ.C: Read after Write ; DQ 3 ; Rank 0 ; Left edge 31 ; Right edge 9 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read after Write ; DQ 4 ; Rank 0 ; Left edge 31 ; Right edge 19 ; DQ delay 3 ; DQS delay 12
SEQ.C: Read after Write ; DQ 5 ; Rank 0 ; Left edge 31 ; Right edge 10 ; DQ delay 8 ; DQ delay 12 ; DQS delay 12
SEQ.C: Read after Write ; DQ 6 ; Rank 0 ; Left edge 10 ; Right edge 10 ; DQ delay 0 ; DQS delay 12
SEQ.C: Read after Write ; DQ 7 ; Rank 0 ; Left edge 31 ; Right edge 8 ; DQ delay 12 ; DQS delay 12
SEQ.C: DQS Enable ; Group 1 ; Rank 0 ; Start VFIFO 5 ; Phase 5 ; Delay 13
SEQ.C: DQS Enable ; Group 1 ; Rank 0 ; End VFIFO 6 ; Phase 4 ; Delay 9
SEQ.C: DQS Enable ; Group 1 ; Rank 0 ; Center VFIFO 6 ; Phase 0 ; Delay 11
SEQ.C: Read Deskew ; DQ 8 ; Rank 0 ; Left edge 16 ; Right edge 27 ; DQ delay 1 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 9 ; Rank 0 ; Left edge 29 ; Right edge 26 ; DQ delay 8 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 10 ; Rank 0 ; Left edge 13 ; Right edge 27 ; DQ delay 0 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 11 ; Rank 0 ; Left edge 30 ; Right edge 24 ; DQ delay 10 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 12 ; Rank 0 ; Left edge 16 ; Right edge 27 ; DQ delay 1 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 13 ; Rank 0 ; Left edge 30 ; Right edge 24 ; DQ delay 10 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 14 ; Rank 0 ; Left edge 13 ; Right edge 27 ; DQ delay 0 ; DQS delay 11
SEQ.C: Read Deskew ; DQ 15 ; Rank 0 ; Left edge 30 ; Right edge 25 ; DQ delay 9 ; DQS delay 11
SEQ.C: Write Deskew ; DQ 8 ; Rank 0 ; Left edge 31 ; Right edge 15 ; DQ delay 8 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 9 ; Rank 0 ; Left edge 31 ; Right edge 16 ; DQ delay 6 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 10 ; Rank 0 ; Left edge 31 ; Right edge 15 ; DQ delay 6 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 11 ; Rank 0 ; Left edge 28 ; Right edge 16 ; DQ delay 6 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 12 ; Rank 0 ; Left edge 31 ; Right edge 14 ; DQ delay 8 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 13 ; Rank 0 ; Left edge 31 ; Right edge 16 ; DQ delay 7 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 14 ; Rank 0 ; Left edge 27 ; Right edge 17 ; DQ delay 5 ; DQS delay 4
SEQ.C: Write Deskew ; DQ 15 ; Rank 0 ; Left edge 29 ; Right edge 17 ; DQ delay 6 ; DQS delay 4
SEQ.C: DM Deskew ; Group 1 ; Left edge 27; Right edge 18; DM delay 4
SEQ.C: Read after Write ; DQ 8 ; Rank 0 ; Left edge 21 ; Right edge 20 ; DQ delay 1 ; DQS delay 12
SEQ.C: Read after Write ; DQ 9 ; Rank 0 ; Left edge 31 ; Right edge 11 ; DQ delay 11 ; DQS delay 12
SEQ.C: Read after Write ; DQ 10 ; Rank 0 ; Left edge 19 ; Right edge 20 ; DQ delay 0 ; DQS delay 12
SEQ.C: Read after Write ; DQ 11 ; Rank 0 ; Left edge 31 ; Right edge 9 ; DQ delay 12 ; DQS delay 12
SEQ.C: Read after Write ; DQ 12 ; Rank 0 ; Left edge 21 ; Right edge 20 ; DQ delay 1 ; DQS delay 12

```

## Analysis of Debug Report

The following analysis will help you interpret the debug report.

- The Read Deskew and Write Deskew results shown in the debug report are before calibration. (Before calibration results are actually from the window seen *during* calibration, and are most useful for debugging.)
- For each DQ group, the Write Deskew, Read Deskew, DM Deskew, and Read after Write results map to the before-calibration margins reported in the EMIF Debug Toolkit.

**Note:** The Write Deskew, Read Deskew, DM Deskew, and Read after Write results are reported in delay steps (nominally 25ps, in Arria V and Cyclone V devices), not in picoseconds.

- DQS Enable calibration is reported as a VFIFO setting (in one clock period steps), a phase tap (in one-eighth clock period steps), and a delay chain step (in 25ps steps).

```
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; Start  VFIFO 5 ; Phase 6 ; Delay 4
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; End   VFIFO 6 ; Phase 5 ; Delay 9
SEQ.C: DQS Enable ; Group 0 ; Rank 0 ; Center VFIFO 6 ; Phase 2 ; Delay 1
```

Analysis of DQS Enable results: A VFIFO tap is 1 clock period, a phase is 1/8 clock period (45 degrees) and delay is nominally 25ps per tap. The DQSen window is the difference between the start and end—for the above example, assuming a frequency of 400 MHz (2500ps), that calculates as follows: `start is 5*2500 + 6*2500/8 +4*25 = 14475ps`. By the same calculation, the end is 16788ps. Consequently, the DQSen window is 2313ps.

- The size of a read window or write window is equal to `(left edge + right edge) * delay chain step size`. Both the left edge and the right edge can be negative or positive.:

```
SEQ.C: Read Deskew ; DQ 0 ; Rank 0 ; Left edge 18 ; Right edge 27 ; DQ
delay 0 ; DQS delay 8
SEQ.C: Write Deskew ; DQ 0 ; Rank 0 ; Left edge 30 ; Right edge 17 ; DQ
delay 6 ; DQS delay 4
```

Analysis of DQ and DQS delay results: The DQ and DQS output delay (write) is the D5 delay chain. The DQ input delay (read) is the D1 delay chain, the DQS input delay (read) is the D4 delay chain.

- Consider the following example of latency results:

```
SEQ.C: LFIFO Calibration ; Latency 10
```

Analysis of latency results: This is the calibrated PHY read latency. The EMIF Debug Toolkit does not report this figure. This latency is reported in clock cycles.

- Consider the following example of FOM results:

```
SEQ.C: FOM IN  = 83
SEQ.C: FOM OUT = 91
```

Analysis of FOM results: The FOM IN value is a measure of the health of the read interface; it is calculated as the sum over all groups of the minimum margin on DQ plus the margin on DQS, divided by 2. The FOM OUT is a measure of the health of the write interface; it is calculated as the sum over all groups of the minimum margin on DQ plus the margin on DQS, divided by 2. You may refer to these values as indicators of improvement when you are experimenting with various termination schemes, assuming there are no individual misbehaving DQ pins.

- The debug report does not provide delay chain step size values. The delay chain step size varies with device speed grade. Refer to your device data sheet for exact incremental delay values for delay chains.

## Related Information

### Functional Description—UniPHY

For more information about calibration, refer to the *Calibration Stages* section in the *Functional Description—UniPHY* chapter of the *External Memory Interface Handbook*.

## Writing a Predefined Data Pattern to SDRAM in the Preloader

You can include your own code to write a predefined data pattern to the SDRAM in the preloader for debugging purposes.

- Include your code in the file: `<project_folder>\software\spl_bsp\uboot-socfpga\arch\arm\cpu\armv7\socfpga\spl.c`.

Adding the following code to the `spl.c` file causes the controller to write walking 1s and walking 0s, repeated five times, to the SDRAM.

```
/*added for demo, place after the last #define statement in spl.c */
#define ROTATE_RIGHT(X) ( (X>>1) | (X&1?0X80000000:0) )
/*added for demo, place after the calibration code */
test_data_walk0((long *)0x100000,PHYS_SDRAM_1_SIZE);
int test_data_walk0(long *base, long maxsize)
{
    volatile long *addr;
    long          cnt;
    ulong         data_temp[3];
    ulong         expected_data[3];
    ulong         read_data;
    int           i = 0; //counter to loop different data pattern
    int           num_address;

    num_address=50;

    data_temp[0]=0xFFFFFFFF; //initial data for walking 0 pattern
    data_temp[1]=0X00000001; //initial data for walking 1 pattern
    data_temp[2]=0XAAAAAAA; //initial data for A->5 switching

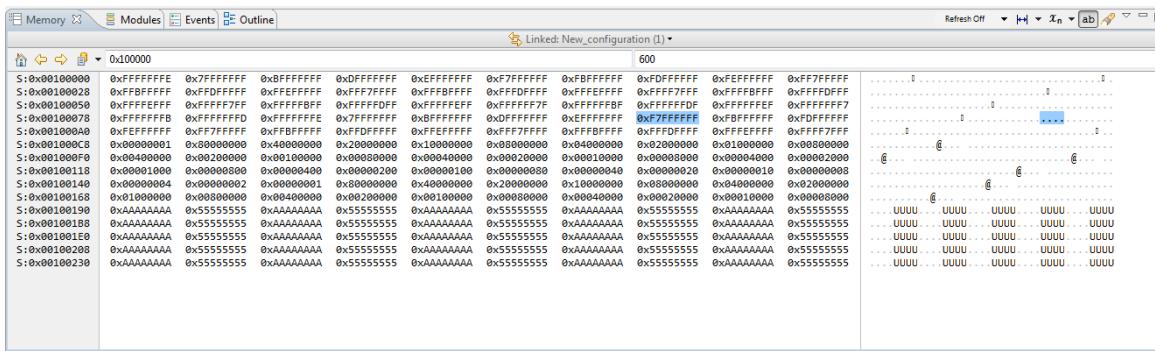
    expected_data[0]=0xFFFFFFFF; //initial data for walking 0 pattern
    expected_data[1]=0X00000001; //initial data for walking 1 pattern
    expected_data[2]=0XAAAAAAA; //initial data for A->5 switching

    for (i=0;i<3;i++) {

        printf("\nSTARTED %08X DATA PATTERN !!!!\n",data_temp[i]);
        /*write*/
        for (cnt = (0+i*num_address); cnt < ((i+1)*num_address) ; cnt++ ) {
            addr = base + cnt; /* pointer arith! */
            sync ();
            *addr = data_temp[i];
            data_temp[i]=ROTATE_RIGHT(data_temp[i]);
        }

        /*read*/
        for (cnt = (0+i*num_address); cnt < ((i+1)*num_address) ; cnt = cnt++ ) {
            addr = base + cnt; /* pointer arith! */
            sync ();
            read_data=*addr;
            printf("Address:%X Expected: %08X      Read:%08X \n",addr,
expected_data[i],read_data);
            if (expected_data[i] !=read_data) {
                puts("!!!!!!FAILED!!!!!!\n\n");
                hang();
            }
            expected_data[i]=ROTATE_RIGHT(expected_data[i]);
        }
    }
}
=====//End Of Code//=====
```

Figure 5-10: Memory Contents After Executing Example Code



## SDRAM Controller Address Map and Register Definitions

This section lists the SDRAM register address map and describes the registers.

### SDRAM Controller Address Map

Address map for the SDRAM Interface registers

Base Address: 0xFFC20000

### SDRAM Controller Module

Register	Offset	Width	Access	Reset Value	Description
<a href="#">ctrlcfg</a> on page 5-45	0x5000	32	RW	0x0	Controller Configuration Register
<a href="#">dramtiming1</a> on page 5-47	0x5004	32	RW	0x0	DRAM Timings 1 Register
<a href="#">dramtiming2</a> on page 5-48	0x5008	32	RW	0x0	DRAM Timings 2 Register
<a href="#">dramtiming3</a> on page 5-49	0x500C	32	RW	0x0	DRAM Timings 3 Register
<a href="#">dramtiming4</a> on page 5-50	0x5010	32	RW	0x0	DRAM Timings 4 Register
<a href="#">lowpwrtiming</a> on page 5-50	0x5014	32	RW	0x0	Lower Power Timing Register
<a href="#">dramodt</a> on page 5-51	0x5018	32	RW	0x0	ODT Control Register
<a href="#">dramaddrw</a> on page 5-52	0x502C	32	RW	0x0	DRAM Address Widths Register
<a href="#">dramifwidth</a> on page 5-53	0x5030	32	RW	0x0	DRAM Interface Data Width Register

Register	Offset	Width	Access	Reset Value	Description
<a href="#">dramsts</a> on page 5-53	0x5038	32	RW	0x0	DRAM Status Register
<a href="#">dramintr</a> on page 5-54	0x503C	32	RW	0x0	ECC Interrupt Register
<a href="#">sbeccount</a> on page 5-55	0x5040	32	RW	0x0	ECC Single Bit Error Count Register
<a href="#">dbecount</a> on page 5-55	0x5044	32	RW	0x0	ECC Double Bit Error Count Register
<a href="#">erraddr</a> on page 5-56	0x5048	32	RW	0x0	ECC Error Address Register
<a href="#">dropcount</a> on page 5-57	0x504C	32	RW	0x0	ECC Auto-correction Dropped Count Register
<a href="#">dropaddr</a> on page 5-57	0x5050	32	RW	0x0	ECC Auto-correction Dropped Address Register
<a href="#">lowpwreq</a> on page 5-58	0x5054	32	RW	0x0	Low Power Control Register
<a href="#">lowpwrack</a> on page 5-59	0x5058	32	RW	0x0	Low Power Acknowledge Register
<a href="#">staticcfg</a> on page 5-59	0x505C	32	RW	0x0	Static Configuration Register
<a href="#">ctrlwidth</a> on page 5-60	0x5060	32	RW	0x0	Memory Controller Width Register
<a href="#">portcfg</a> on page 5-61	0x507C	32	RW	0x0	Port Configuration Register
<a href="#">fpgaportrst</a> on page 5-63	0x5080	32	RW	0x0	FPGA Ports Reset Control Register
<a href="#">protportdefault</a> on page 5-64	0x508C	32	RW	0x0	Memory Protection Port Default Register
<a href="#">proruleaddr</a> on page 5-65	0x5090	32	RW	0x0	Memory Protection Address Register
<a href="#">proruleid</a> on page 5-66	0x5094	32	RW	0x0	Memory Protection ID Register
<a href="#">proruledata</a> on page 5-67	0x5098	32	RW	0x0	Memory Protection Rule Data Register
<a href="#">protrulerdwr</a> on page 5-68	0x509C	32	RW	0x0	Memory Protection Rule Read-Write Register
<a href="#">mppriority</a> on page 5-69	0x50AC	32	RW	0x0	Scheduler priority Register
<a href="#">remappriority</a> on page 5-69	0x50E0	32	RW	0x0	Controller Command Pool Priority Remap Register

## Port Sum of Weight Register

Register	Offset	Width	Access	Reset Value	Description
<a href="#">mpweight_0_4</a> on page 5-70	0x50B0	32	RW	0x0	Port Sum of Weight Register[1/4]
<a href="#">mpweight_1_4</a> on page 5-71	0x50B4	32	RW	0x0	Port Sum of Weight Register[2/4]
<a href="#">mpweight_2_4</a> on page 5-71	0x50B8	32	RW	0x0	Port Sum of Weight Register[3/4]
<a href="#">mpweight_3_4</a> on page 5-72	0x50BC	32	RW	0x0	Port Sum of Weight Register[4/4]

## SDRAM Controller Module Register Descriptions

Address map for the SDRAM controller and multi-port front-end. All registers in this group reset to zero.

Offset: 0x5000

### [ctrlcfg](#) on page 5-45

The Controller Configuration Register determines the behavior of the controller.

### [dramtiming1](#) on page 5-47

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

### [dramtiming2](#) on page 5-48

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

### [dramtiming3](#) on page 5-49

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

### [dramtiming4](#) on page 5-50

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

### [lowpwrtiming](#) on page 5-50

This register controls the behavior of the low power logic in the controller.

### [dramodt](#) on page 5-51

This register controls which ODT pin asserts with chip select 0 (CS0) assertion and which ODT pin asserts with chip select 1 (CS1) assertion.

### [dramaddrw](#) on page 5-52

This register configures the width of the various address fields of the DRAM. The values specified in this register must match the memory devices being used.

### [dramifwidth](#) on page 5-53

This register controls the interface width of the SDRAM controller.

### [dramsts](#) on page 5-53

This register provides the status of the calibration and ECC logic.

**dramintr** on page 5-54

This register can enable, disable and clear the SDRAM error interrupts.

**sbeccount** on page 5-55

This register tracks the single-bit error count.

**dbecount** on page 5-55

This register tracks the double-bit error count.

**erraddr** on page 5-56

This register holds the address of the most recent ECC error.

**dropcount** on page 5-57

This register holds the address of the most recent ECC error.

**dropaddr** on page 5-57

This register holds the last dropped address.

**lowpwreq** on page 5-58

This register instructs the controller to put the DRAM into a power down state. Note that some commands are only valid for certain memory types.

**lowpwrack** on page 5-59

This register gives the status of the power down commands requested by the Low Power Control register.

**staticcfg** on page 5-59

This register controls configuration values which cannot be updated during active transfers. First configure the `memb1` and `eccn` fields and then re-write these fields while setting the `applycfg` bit. The `applycfg` bit is write only.

**ctrlwidth** on page 5-60

This register controls the width of the physical DRAM interface.

**portcfg** on page 5-61

Each bit of the `autopchen` field maps to one of the control ports. If a port executes mostly sequential memory accesses, the corresponding `autopchen` bit should be 0. If the port has highly random accesses, then its `autopchen` bit should be set to 1.

**fpgaportrst** on page 5-63

This register implements functionality to allow the CPU to control when the MPFE will enable the ports to the FPGA fabric.

**protportdefault** on page 5-64

This register controls the default protection assignment for a port. Ports which have explicit rules which define regions which are illegal to access should set the bits to pass by default. Ports which have explicit rules which define legal areas should set the bit to force all transactions to fail. Leaving this register to all zeros should be used for systems which do not desire any protection from the memory controller.

**protruleaddr** on page 5-65

This register is used to control the memory protection for port 0 transactions. Address ranges can either be used to allow access to memory regions or disallow access to memory regions. If TrustZone is being used, access can be enabled for protected transactions or disabled for unprotected transactions. The default state of this register is to allow all access. Address values used for protection are only physical addresses.

**protruleid** on page 5-66

This register configures the AxID for a given protection rule.

**protruledata** on page 5-67

This register configures the protection memory characteristics of each protection rule.

**protrulerdwr** on page 5-68

This register is used to perform read and write operations to the internal protection table.

**mppriority** on page 5-69

This register is used to configure the DRAM burst operation scheduling.

**remappriority** on page 5-69

This register applies another level of port priority after a transaction is placed in the single port queue.

**Port Sum of Weight Register Register Descriptions** on page 5-70

This register is used to configure the DRAM burst operation scheduling.

**ctrlcfg**

The Controller Configuration Register determines the behavior of the controller.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25000

Offset: 0x5000

Access: RW

Bit Fields																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
						burst termen	burst intren	nodmp ins	dgstr ken	starvelimit							
						RW 0x0	RW 0x0	RW 0x0	RW 0x0	RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
reordern RW 0x0	gendbe RW 0x0	gensbe RW 0x0	cfg_e enabl e_ecc_ code_ overwrit es RW 0x0	eccc rrren	eccen	addrorder RW 0x0		membl RW 0x0				memtype RW 0x0					

**ctrlcfg Fields**

Bit	Name	Description	Access	Reset
25	burstterminen	Set to a one to enable the controller to issue burst terminate commands. This must only be set when the DRAM memory type is LPDDR2.	RW	0x0

Bit	Name	Description	Access	Reset
24	burstintren	Set to a one to enable the controller to issue burst interrupt commands. This must only be set when the DRAM memory type is LPDDR2.	RW	0x0
23	nodmpins	Set to a one to enable DRAM operation if no DM pins are connected.	RW	0x0
22	dqstrken	Enables DQS tracking in the PHY.	RW	0x0
21:16	starvelimit	Specifies the number of DRAM burst transactions an individual transaction will allow to reorder ahead of it before its priority is raised in the memory controller.	RW	0x0
15	reorderen	This bit controls whether the controller can re-order operations to optimize SDRAM bandwidth. It should generally be set to a one.	RW	0x0
14	gendbe	Enable the deliberate insertion of double bit errors in data written to memory. This should only be used for testing purposes.	RW	0x0
13	gensbe	Enable the deliberate insertion of single bit errors in data written to memory. This should only be used for testing purposes.	RW	0x0
12	cfg_enable_ecc_code_overwrites	Set to a one to enable ECC overwrites. ECC overwrites occur when a correctable ECC error is seen and cause a new read/modify/write to be scheduled for that location to clear the ECC error.	RW	0x0
11	ecccorgen	Enable auto correction of the read data returned when single bit error is detected.	RW	0x0
10	eccen	Enable the generation and checking of ECC. This bit must only be set if the memory connected to the SDRAM interface is 24 or 40 bits wide. If you set this, you must clear the useeccasdata field in the staticcfg register.	RW	0x0

Bit	Name	Description	Access	Reset														
9:8	addrorder	<p>This bit field selects the order for address interleaving. Programming this field with different values gives different mappings between the AXI or Avalon-MM address and the SDRAM address.</p> <p>Program this field with the following binary values to select the ordering.</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>chip, row, bank, column</td></tr> <tr> <td>0x1</td><td>chip, bank, row, column</td></tr> <tr> <td>0x2</td><td>row, chip, bank, column</td></tr> <tr> <td>0x3</td><td>reserved</td></tr> </tbody> </table>	Value	Description	0x0	chip, row, bank, column	0x1	chip, bank, row, column	0x2	row, chip, bank, column	0x3	reserved	RW	0x0				
Value	Description																	
0x0	chip, row, bank, column																	
0x1	chip, bank, row, column																	
0x2	row, chip, bank, column																	
0x3	reserved																	
7:3	membl	Configures burst length as a static decimal value. Legal values are valid for JEDEC allowed DRAM values for the DRAM selected in cfg_type. For DDR3, this should be programmed with 8 (binary "01000"), for DDR2 it can be either 4 or 8 depending on the exact DRAM chip. LPDDR2 can be programmed with 4, 8, or 16 and LPDDR can be programmed with 2, 4, or 8. You must also program the membl field in the staticcfg register.	RW	0x0														
2:0	memtype	<p>This bit field selects the memory type. This field can be programmed with the following binary values:</p> <table> <thead> <tr> <th>Value</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x0</td><td>Reserved</td></tr> <tr> <td>0x1</td><td>Memory type is DDR2 SDRAM</td></tr> <tr> <td>0x2</td><td>Memory type is DDR3 SDRAM</td></tr> <tr> <td>0x3</td><td>reserved</td></tr> <tr> <td>0x4</td><td>Memory type is LPDDR2 SDRAM</td></tr> <tr> <td>0x5-0x7</td><td>Reserved</td></tr> </tbody> </table>	Value	Description	0x0	Reserved	0x1	Memory type is DDR2 SDRAM	0x2	Memory type is DDR3 SDRAM	0x3	reserved	0x4	Memory type is LPDDR2 SDRAM	0x5-0x7	Reserved	RW	0x0
Value	Description																	
0x0	Reserved																	
0x1	Memory type is DDR2 SDRAM																	
0x2	Memory type is DDR3 SDRAM																	
0x3	reserved																	
0x4	Memory type is LPDDR2 SDRAM																	
0x5-0x7	Reserved																	

### dramtiming1

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25004

Offset: 0x5004

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
trfc RW 0x0								tfaw RW 0x0							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
trrd RW 0x0		tcl RW 0x0				tal RW 0x0				tcwl RW 0x0					

### dramtiming1 Fields

Bit	Name	Description	Access	Reset
31:24	trfc	The refresh cycle timing parameter.	RW	0x0
23:18	tfaw	The four-activate window timing parameter.	RW	0x0
17:14	trrd	The activate to activate, different banks timing parameter.	RW	0x0
13:9	tcl	Memory read latency.	RW	0x0
8:4	tal	Memory additive latency.	RW	0x0
3:0	tcwl	Memory write latency.	RW	0x0

### dramtiming2

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25008

Offset: 0x5008

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved			twtr RW 0x0				twr RW 0x0				trp RW 0x0				trcd
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
trcd RW 0x0			trefi RW 0x0												

**dramtiming2 Fields**

Bit	Name	Description	Access	Reset
28:25	twtr	The write to read timing parameter.	RW	0x0
24:21	twr	The write recovery timing.	RW	0x0
20:17	trp	The precharge to activate timing parameter.	RW	0x0
16:13	trcd	The activate to read/write timing parameter.	RW	0x0
12:0	trefi	The refresh interval timing parameter.	RW	0x0

**dramtiming3**

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2500C

Offset: 0x500C

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved										tcccd				tmrd		
										RW 0x0				RW 0x0		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
tmrd	trc						tras						trtp			
	RW 0x0						RW 0x0						RW 0x0			

**dramtiming3 Fields**

Bit	Name	Description	Access	Reset
22:19	tcccd	The CAS to CAS delay time.	RW	0x0
18:15	tmrd	Mode register timing parameter.	RW	0x0
14:9	trc	The activate to activate timing parameter.	RW	0x0
8:4	tras	The activate to precharge timing parameter.	RW	0x0
3:0	trtp	The read to precharge timing parameter.	RW	0x0

**dramtiming4**

This register implements JEDEC standardized timing parameters. It should be programmed in clock cycles, for the value specified by the memory vendor.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25010

Offset: 0x5010

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved										minpwrsavecycles			pwrdownexit			
										RW 0x0			RW 0x0			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
pwrdownexit								selfrfshexit								
								RW 0x0								

**dramtiming4 Fields**

Bit	Name	Description	Access	Reset
23:20	minpwrsavecycles	The minimum number of cycles to stay in a low power state. This applies to both power down and self-refresh and should be set to the greater of tPD and tCKESR.	RW	0x0
19:10	pwrdownexit	The power down exit cycles, tXP DLL.	RW	0x0
9:0	selfrfshexit	The self refresh exit cycles, tXS.	RW	0x0

**lowpwrtiming**

This register controls the behavior of the low power logic in the controller.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25014

Offset: 0x5014

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved												clkdisablecycles RW 0x0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
autopdcycles RW 0x0																

### lowpwrtiming Fields

Bit	Name	Description	Access	Reset
19:16	clkdisablecycles	Set to a the number of clocks after the execution of an self-refresh to stop the clock. This register is generally set based on PHY design latency and should generally not be changed.	RW	0x0
15:0	autopdcycles	The number of idle clock cycles after which the controller should place the memory into power-down mode.	RW	0x0

### dramodt

This register controls which ODT pin asserts with chip select 0 (CS0) assertion and which ODT pin asserts with chip select 1 (CS1) assertion.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25018

Offset: 0x5018

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												cfg_read_odt_chip RW 0x0				
												cfg_write_odt_chip RW 0x0				

**dramodt Fields**

Bit	Name	Description	Access	Reset
7:4	cfg_read_odt_chip	This register controls which ODT pin is asserted during reads. Bits[5:4] select the ODT pin that asserts with CS0 and bits[7:6] select the ODT pin that asserts with CS1. For example, a value of 0x9 asserts ODT[0] for accesses CS0 and ODT[1] for accesses with CS1. This field can be set to 0x1 if there is only one chip select available.	RW	0x0
3:0	cfg_write_odt_chip	This register controls which ODT pin is asserted during writes. Bits[1:0] select the ODT pin that asserts with CS0 and bits[3:2] select the ODT pin that asserts with CS1. For example, a value of 0x9 asserts ODT[0] for accesses CS0 and ODT[1] for accesses with CS1. This field can be set to 0x1 if there is only one chip select available.	RW	0x0

**dramaddrw**

This register configures the width of the various address fields of the DRAM. The values specified in this register must match the memory devices being used.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2502C

Offset: 0x502C

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
csbits				bankbits				rowbits				colbits				
RW 0x0				RW 0x0				RW 0x0				RW 0x0				

**dramaddrw Fields**

Bit	Name	Description	Access	Reset
15:13	csbits	The number of chip select address bits for the memory devices in your memory interface.	RW	0x0
12:10	bankbits	The number of bank address bits for the memory devices in your memory interface.	RW	0x0
9:5	rowbits	The number of row address bits for the memory devices in your memory interface.	RW	0x0

Bit	Name	Description	Access	Reset
4:0	colbits	The number of column address bits for the memory devices in your memory interface.	RW	0x0

**dramifwidth**

This register controls the interface width of the SDRAM controller.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25030

Offset: 0x5030

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved										ifwidth RW 0x0						

**dramifwidth Fields**

Bit	Name	Description	Access	Reset
7:0	ifwidth	This register controls the width of the SDRAM interface, including any bits used for ECC. For example, for a 32-bit interface with ECC, program this register to 0x28. The ctrlwidth register must also be programmed.	RW	0x0

**dramsts**

This register provides the status of the calibration and ECC logic.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25038

Offset: 0x5038

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												corrdrop	dbeer	sbeer	calfa	calsucces
												r	r	il	ss	
												RW	RW	RW	RW	RW 0x0
												0x0	0x0	0x0	0x0	0x0

**dramsts Fields**

Bit	Name	Description	Access	Reset
4	corrdrop	This bit is set to 1 when any auto-corrections have been dropped.	RW	0x0
3	dbeer	This bit is set to 1 when any ECC double bit errors are detected.	RW	0x0
2	sbeer	This bit is set to 1 when any ECC single bit errors are detected.	RW	0x0
1	calfail	This bit is set to 1 when the PHY is unable to calibrate.	RW	0x0
0	calsuccess	This bit will be set to 1 if the PHY was able to successfully calibrate.	RW	0x0

**dramintr**

This register can enable, disable and clear the SDRAM error interrupts.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2503C

Offset: 0x503C

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												intrclr	corrdropma	dbemask	sbemask	intren
												sk	sk	sk	sk	
												RW	RW	RW	RW	RW 0x0
												0x0	0x0	0x0	0x0	0x0

**dramintr Fields**

Bit	Name	Description	Access	Reset
4	intrclr	Writing to this self-clearing bit clears the interrupt signal. Writing to this bit also clears the error count and error address registers: sbecount, dbecount, dropcount, erraddr, and dropaddr.	RW	0x0
3	corrdropmask	Set this bit to a one to mask interrupts for an ECC correction write back needing to be dropped. This indicates a burst of memory errors in a short period of time.	RW	0x0
2	dbemask	Mask the double bit error interrupt.	RW	0x0
1	sbemask	Mask the single bit error interrupt.	RW	0x0
0	intren	Enable the interrupt output.	RW	0x0

**sbecount**

This register tracks the single-bit error count.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25040

Offset: 0x5040

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								count RW 0x0							

**sbecount Fields**

Bit	Name	Description	Access	Reset
7:0	count	Reports the number of single bit errors that have occurred since the status register counters were last cleared.	RW	0x0

**dbecount**

This register tracks the double-bit error count.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25044

Offset: 0x5044

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																
count																
RW 0x0																

**dbecount Fields**

Bit	Name	Description	Access	Reset
7:0	count	Reports the number of double bit errors that have occurred since the status register counters were last cleared.	RW	0x0

**erraddr**

This register holds the address of the most recent ECC error.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25048

Offset: 0x5048

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
addr																
RW 0x0																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
addr																
RW 0x0																

**erraddr Fields**

Bit	Name	Description	Access	Reset
31:0	addr	The address of the most recent ECC error.	RW	0x0

**dropcount**

This register holds the address of the most recent ECC error.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2504C

Offset: 0x504C

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved										corrdropcount						
RW 0x0																

**dropcount Fields**

Bit	Name	Description	Access	Reset
7:0	corrdropcount	This gives the count of the number of ECC write back transactions dropped due to the internal FIFO overflowing.	RW	0x0

**dropaddr**

This register holds the last dropped address.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25050

Offset: 0x5050

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
corrdropaddr																
RW 0x0																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
corrdropaddr										RW 0x0						

**dropaddr Fields**

Bit	Name	Description	Access	Reset
31:0	corrdropaddr	This register gives the last address which was dropped.	RW	0x0

**lowpwreq**

This register instructs the controller to put the DRAM into a power down state. Note that some commands are only valid for certain memory types.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25054

Offset: 0x5054

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved										selfrfshmask	selfrshreq	deeppwrndmask	deeppwrndreq			
										RW 0x0	RW 0x0	RW 0x0	RW 0x0			

**lowpwreq Fields**

Bit	Name	Description	Access	Reset
5:4	selfrfshmask	Write a one to each bit of this field to have a self refresh request apply to both chips.	RW	0x0
3	selfrshreq	Write a one to this bit to request the RAM be put into a self refresh state. This bit is treated as a static value so the RAM will remain in self-refresh as long as this register bit is set to a one. This power down mode can be selected for all DRAMs supported by the controller.	RW	0x0
2:1	deeppwrndmask	Write ones to this register to select which DRAM chip selects will be powered down. Typical usage is to set both of these bits when deeppwrndreq is set but the controller does support putting a single chip into deep power down and keeping the other chip running.	RW	0x0

Bit	Name	Description	Access	Reset
0	deeppwrdnreq	Write a one to this bit to request a deep power down. This bit should only be written with LPDDR2 DRAMs, DDR3 DRAMs do not support deep power down.	RW	0x0

**lowpwrack**

This register gives the status of the power down commands requested by the Low Power Control register.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25058

Offset: 0x5058

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved																
															selfrfshack	deeppwrdnack
															RW	0x0

**lowpwrack Fields**

Bit	Name	Description	Access	Reset
1	selfrfshack	This bit is a one to indicate that the controller is in a self-refresh state.	RW	0x0
0	deeppwrdnack	This bit is set to a one after a deep power down has been executed	RW	0x0

**staticcfg**

This register controls configuration values which cannot be updated during active transfers. First configure the membl and eccn fields and then re-write these fields while setting the applycfg bit. The applycfg bit is write only.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2505C

Offset: 0x505C

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved												applycfg RW 0x0	useeccasdata RW 0x0	membl RW 0x0		

### staticcfg Fields

Bit	Name	Description	Access	Reset
3	applycfg	Write with this bit set to apply all the settings loaded in SDR registers to the memory interface. This bit is write-only and always returns 0 if read.	RW	0x0
2	useeccasdata	This field allows the FPGA ports to directly access the extra data bits that are normally used to hold the ECC code. The interface width must be set to 24 or 40 in the dramifwidth register. If you set this, you must clear the eccen field in the ctrlcfg register.	RW	0x0
1:0	membl	<p>This field specifies the DRAM burst length. The following encodings set the burst length:</p> <ul style="list-style-type: none"> <li>• 0x0= Burst length of 2 clocks</li> <li>• 0x1= Burst length of 4 clocks</li> <li>• 0x2= Burst length of 8 clocks</li> <li>• 0x3= Burst length of 16 clocks</li> </ul> <p>If you program the this field, you must also set the membl field in the ctrlcfg register.</p>	RW	0x0

### ctrlwidth

This register controls the width of the physical DRAM interface.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25060

Offset: 0x5060

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved															ctrlwidth RW 0x0	

### ctrlwidth Fields

Bit	Name	Description	Access	Reset								
1:0	ctrlwidth	<p>This field specifies the SDRAM controller interface width:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>8-bit interface width</td> </tr> <tr> <td>0x1</td> <td>16-bit (no ECC) or 24-bit (ECC enabled) interface width</td> </tr> <tr> <td>0x2</td> <td>32-bit (no ECC) or 40-bit (ECC enabled) interface width</td> </tr> </tbody> </table> <p>Additionally, you must program the dramifwidth register.</p>	Value	Description	0x0	8-bit interface width	0x1	16-bit (no ECC) or 24-bit (ECC enabled) interface width	0x2	32-bit (no ECC) or 40-bit (ECC enabled) interface width	RW	0x0
Value	Description											
0x0	8-bit interface width											
0x1	16-bit (no ECC) or 24-bit (ECC enabled) interface width											
0x2	32-bit (no ECC) or 40-bit (ECC enabled) interface width											

### portcfg

Each bit of the autopchen field maps to one of the control ports. If a port executes mostly sequential memory accesses, the corresponding autopchen bit should be 0. If the port has highly random accesses, then its autopchen bit should be set to 1.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2507C

Offset: 0x507C

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
autopchen RW 0x0																
autopchen RW 0x0								portprotocol								

**portcfg Fields**

Bit	Name	Description	Access	Reset																										
19:10	autopchen	<p>Auto-Precharge Enable: One bit is assigned to each control port. For each bit, the encodings are as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>The controller requests an automatic precharge following a bus command completion (close the row automatically)</td> </tr> <tr> <td>0x1</td> <td>The controller attempts to keep a row open. All active ports with random dominated operations should set the <code>autopchen</code> bit to 1.</td> </tr> </tbody> </table> <p>The bits in this field correspond to the control ports as follows:</p> <table> <tbody> <tr> <td>Bit 9</td> <td>CPU write</td> </tr> <tr> <td>Bit 8</td> <td>L3 write</td> </tr> <tr> <td>Bit 7</td> <td>CPU read</td> </tr> <tr> <td>Bit 6</td> <td>L3 read</td> </tr> <tr> <td>Bit 5</td> <td>FPGA-to-SDRAM port 5</td> </tr> <tr> <td>Bit 4</td> <td>FPGA-to-SDRAM port 4</td> </tr> <tr> <td>Bit 3</td> <td>FPGA-to-SDRAM port 3</td> </tr> <tr> <td>Bit 2</td> <td>FPGA-to-SDRAM port 2</td> </tr> <tr> <td>Bit 1</td> <td>FPGA-to-SDRAM port 1</td> </tr> <tr> <td>Bit 0</td> <td>FPGA-to-SDRAM port 0</td> </tr> </tbody> </table>	Value	Description	0x0	The controller requests an automatic precharge following a bus command completion (close the row automatically)	0x1	The controller attempts to keep a row open. All active ports with random dominated operations should set the <code>autopchen</code> bit to 1.	Bit 9	CPU write	Bit 8	L3 write	Bit 7	CPU read	Bit 6	L3 read	Bit 5	FPGA-to-SDRAM port 5	Bit 4	FPGA-to-SDRAM port 4	Bit 3	FPGA-to-SDRAM port 3	Bit 2	FPGA-to-SDRAM port 2	Bit 1	FPGA-to-SDRAM port 1	Bit 0	FPGA-to-SDRAM port 0	RW	0x0
Value	Description																													
0x0	The controller requests an automatic precharge following a bus command completion (close the row automatically)																													
0x1	The controller attempts to keep a row open. All active ports with random dominated operations should set the <code>autopchen</code> bit to 1.																													
Bit 9	CPU write																													
Bit 8	L3 write																													
Bit 7	CPU read																													
Bit 6	L3 read																													
Bit 5	FPGA-to-SDRAM port 5																													
Bit 4	FPGA-to-SDRAM port 4																													
Bit 3	FPGA-to-SDRAM port 3																													
Bit 2	FPGA-to-SDRAM port 2																													
Bit 1	FPGA-to-SDRAM port 1																													
Bit 0	FPGA-to-SDRAM port 0																													

Bit	Name	Description	Access	Reset																										
9:0	portprotocol	<p>Port Protocol: One bit is assigned to each control port. Each bit is encoded as follows:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x1</td> <td>AXI (reset value for all ports)</td> </tr> <tr> <td>0x0</td> <td>Avalon-MM</td> </tr> </tbody> </table> <p>The bits in this field correspond to the control ports as follows:</p> <table> <tbody> <tr> <td>Bit 9</td> <td>CPU write</td> </tr> <tr> <td>Bit 8</td> <td>L3 write</td> </tr> <tr> <td>Bit 7</td> <td>CPU read</td> </tr> <tr> <td>Bit 6</td> <td>L3 read</td> </tr> <tr> <td>Bit 5</td> <td>FPGA-to-SDRAM port 5</td> </tr> <tr> <td>Bit 4</td> <td>FPGA-to-SDRAM port 4</td> </tr> <tr> <td>Bit 3</td> <td>FPGA-to-SDRAM port 3</td> </tr> <tr> <td>Bit 2</td> <td>FPGA-to-SDRAM port 2</td> </tr> <tr> <td>Bit 1</td> <td>FPGA-to-SDRAM port 1</td> </tr> <tr> <td>Bit 0</td> <td>FPGA-to-SDRAM port 0</td> </tr> </tbody> </table> <p><b>Note:</b> The AXI protocol requires both a read and a write port. Therefore, you must ensure that AXI ports are allocated in the pairs (port 0, port 1), (port 2, port 3), and (port 4, port 5). Aside from the requirement noted above, AXI and Avalon-MM ports can be mixed freely.</p>	Value	Description	0x1	AXI (reset value for all ports)	0x0	Avalon-MM	Bit 9	CPU write	Bit 8	L3 write	Bit 7	CPU read	Bit 6	L3 read	Bit 5	FPGA-to-SDRAM port 5	Bit 4	FPGA-to-SDRAM port 4	Bit 3	FPGA-to-SDRAM port 3	Bit 2	FPGA-to-SDRAM port 2	Bit 1	FPGA-to-SDRAM port 1	Bit 0	FPGA-to-SDRAM port 0	RW	0x0
Value	Description																													
0x1	AXI (reset value for all ports)																													
0x0	Avalon-MM																													
Bit 9	CPU write																													
Bit 8	L3 write																													
Bit 7	CPU read																													
Bit 6	L3 read																													
Bit 5	FPGA-to-SDRAM port 5																													
Bit 4	FPGA-to-SDRAM port 4																													
Bit 3	FPGA-to-SDRAM port 3																													
Bit 2	FPGA-to-SDRAM port 2																													
Bit 1	FPGA-to-SDRAM port 1																													
Bit 0	FPGA-to-SDRAM port 0																													

### fpgaportrst

This register implements functionality to allow the CPU to control when the MPFE will enable the ports to the FPGA fabric.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25080

Offset: 0x5080

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved		portrstn RW 0x0														

### fpgaportrst Fields

Bit	Name	Description	Access	Reset
13:0	portrstn	This register should be written to with a 1 to enable the selected FPGA port to exit reset. Writing a bit to a zero will stretch the port reset until the register is written. Read data ports are connected to bits 3:0, with read data port 0 at bit 0 to read data port 3 at bit 3. Write data ports 0 to 3 are mapped to 4 to 7, with write data port 0 connected to bit 4 to write data port 3 at bit 7. Command ports are connected to bits 8 to 13, with command port 0 at bit 8 to command port 5 at bit 13. Expected usage would be to set all the bits at the same time but setting some bits to a zero and others to a one is supported.	RW	0x0

### protportdefault

This register controls the default protection assignment for a port. Ports which have explicit rules which define regions which are illegal to access should set the bits to pass by default. Ports which have explicit rules which define legal areas should set the bit to force all transactions to fail. Leaving this register to all zeros should be used for systems which do not desire any protection from the memory controller.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2508C

Offset: 0x508C

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved							portdefault RW 0x0									

**protportdefault Fields**

Bit	Name	Description	Access	Reset
9:0	portdefault	<p>Determines the default action for specified transactions. When a bit is zero, the specified access is allowed by default. When a bit is one, the specified access is denied by default.</p> <ul style="list-style-type: none"> <li>Bit 9 CPU write</li> <li>Bit 8 L3 write</li> <li>Bit 7 CPU read</li> <li>Bit 6 L3 read</li> <li>Bit 5 Access to FPGA-to-SDRAM port 5</li> <li>Bit 4 Access to FPGA-to-SDRAM port 4</li> <li>Bit 3 Access to FPGA-to-SDRAM port 3</li> <li>Bit 2 Access to FPGA-to-SDRAM port 2</li> <li>Bit 1 Access to FPGA-to-SDRAM port 1</li> <li>Bit 0 Access to FPGA-to-SDRAM port 0</li> </ul>	RW	0x0

**protruleaddr**

This register is used to control the memory protection for port 0 transactions. Address ranges can either be used to allow access to memory regions or disallow access to memory regions. If TrustZone is being used, access can be enabled for protected transactions or disabled for unprotected transactions. The default state of this register is to allow all access. Address values used for protection are only physical addresses.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25090

Offset: 0x5090

Access: RW

Bit Fields																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16								
Reserved										highaddr													
										RW 0x0													
highaddr								lowaddr															
								RW 0x0															

**proruleaddr Fields**

Bit	Name	Description	Access	Reset
23:12	highaddr	Upper 12 bits of the address for a check. Address is compared to be greater than or equal to the address of a transaction. Note that since AXI transactions cannot cross a 4K byte boundary, the transaction start and transaction end address must also fall within the same 1MByte block pointed to by this address pointer.	RW	0x0
11:0	lowaddr	Lower 12 bits of the address for a check. Address is compared to be less than or equal to the address of a transaction. Note that since AXI transactions cannot cross a 4K byte boundary, the transaction start and transaction end address must also fall within the same 1MByte block pointed to by this address pointer.	RW	0x0

**proruleid**

This register configures the AxID for a given protection rule.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25094

Offset: 0x5094

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved								highid RW 0x0								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
highid RW 0x0				lowid RW 0x0												

**proruleid Fields**

Bit	Name	Description	Access	Reset
23:12	highid	AxID for the protection rule. Incoming AxID needs to be less than or equal to this value. For all AxIDs from a port, AxID high should be programmed to all ones.	RW	0x0
11:0	lowid	AxID for the protection rule. Incoming AxID needs to be greater than or equal to this value. For all AxIDs from a port, AxID high should be programmed to all ones.	RW	0x0

**protruledata**

This register configures the protection memory characteristics of each protection rule.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC25098

Offset: 0x5098

Access: RW

Bit Fields																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Reserved	ruleresult  RW 0x0	portmask  RW 0x0										validrule  RW 0x0	security  RW 0x0				

**protruledata Fields**

Bit	Name	Description	Access	Reset																						
13	ruleresult	Set this bit to a one to force a protection failure, zero to allow the access to succeed	RW	0x0																						
12:3	portmask	<p>The bits in this field determine which ports the rule applies to. If a port's bit is set, the rule applies to that port; if the bit is clear, the rule does not apply. The bits in this field correspond to the control ports as follows:</p> <table> <tbody> <tr><td>Bit 9</td><td>CPU write</td></tr> <tr><td>Bit 8</td><td>L3 write</td></tr> <tr><td>Bit 7</td><td>CPU read</td></tr> <tr><td>Bit 6</td><td>L3 read</td></tr> <tr><td>Bit 5</td><td>FPGA-to-SDRAM port 5</td></tr> <tr><td>Bit 4</td><td>FPGA-to-SDRAM port 4</td></tr> <tr><td>Bit 3</td><td>FPGA-to-SDRAM port 3</td></tr> <tr><td>Bit 2</td><td>FPGA-to-SDRAM port 2</td></tr> <tr><td>Bit 1</td><td>FPGA-to-SDRAM port 1</td></tr> <tr><td>Bit 0</td><td>FPGA-to-SDRAM port 0</td></tr> <tr><td colspan="2">&amp;</td></tr> </tbody> </table>	Bit 9	CPU write	Bit 8	L3 write	Bit 7	CPU read	Bit 6	L3 read	Bit 5	FPGA-to-SDRAM port 5	Bit 4	FPGA-to-SDRAM port 4	Bit 3	FPGA-to-SDRAM port 3	Bit 2	FPGA-to-SDRAM port 2	Bit 1	FPGA-to-SDRAM port 1	Bit 0	FPGA-to-SDRAM port 0	&		RW	0x0
Bit 9	CPU write																									
Bit 8	L3 write																									
Bit 7	CPU read																									
Bit 6	L3 read																									
Bit 5	FPGA-to-SDRAM port 5																									
Bit 4	FPGA-to-SDRAM port 4																									
Bit 3	FPGA-to-SDRAM port 3																									
Bit 2	FPGA-to-SDRAM port 2																									
Bit 1	FPGA-to-SDRAM port 1																									
Bit 0	FPGA-to-SDRAM port 0																									
&																										
2	validrule	Set to bit to a one to make a rule valid, set to a zero to invalidate a rule.	RW	0x0																						

Bit	Name	Description	Access	Reset								
1:0	security	<p>Valid security field encodings are:</p> <table> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0x0</td> <td>Rule applies to secure transactions</td> </tr> <tr> <td>0x1</td> <td>Rule applies to non-secure transactions</td> </tr> <tr> <td>0x2 or 0x3</td> <td>Rule applies to secure and non-secure transactions</td> </tr> </tbody> </table>	Value	Description	0x0	Rule applies to secure transactions	0x1	Rule applies to non-secure transactions	0x2 or 0x3	Rule applies to secure and non-secure transactions	RW	0x0
Value	Description											
0x0	Rule applies to secure transactions											
0x1	Rule applies to non-secure transactions											
0x2 or 0x3	Rule applies to secure and non-secure transactions											

### protrulerdwr

This register is used to perform read and write operations to the internal protection table.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC2509C

Offset: 0x509C

Access: RW

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										readrule	writerule	ruleoffset			
										RW	RW	RW 0x0			

### protrulerdwr Fields

Bit	Name	Description	Access	Reset
6	readrule	Write to this bit to have the memory_prot_data register loaded with the value from the internal protection table at offset. Table value will be loaded before a rd is returned so read data from the register will be correct for any follow-on reads to the memory_prot_data register.	RW	0x0
5	writerule	Write to this bit to have the memory_prot_data register to the table at the offset specified by port_offset. Bit automatically clears after a single cycle and the write operation is complete.	RW	0x0
4:0	ruleoffset	This field defines which of the 20 rules in the protection table you want to read or write.	RW	0x0

**mppriority**

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250AC

Offset: 0x50AC

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved		userpriority RW 0x0														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
userpriority RW 0x0																

**mppriority Fields**

Bit	Name	Description	Access	Reset
29:0	userpriority	User Priority: This field sets the absolute user priority of each port, which is represented as a 3-bit value. 0x0 is the lowest priority and 0x7 is the highest priority. Port 0 is configured by programming userpriority[2:0], port 1 is configured by programming userpriority[5:3], port 2 is configured by programming userpriority[8:6], and so on.	RW	0x0

**remappriority**

This register applies another level of port priority after a transaction is placed in the single port queue.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250E0

Offset: 0x50E0

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved								priorityremap RW 0x0								

**remappriority Fields**

Bit	Name	Description	Access	Reset
7:0	priorityremap	Each bit of this field represents a priority level. If bit N in the priorityremap field is set, then any port transaction with absolute user priority of N jumps to the front of the single port queue and is serviced ahead of any transactions in the queue. For example, if bit 5 is set in the priorityremap field of the remappriority register, then any port transaction with a userpriority value of 0x5 in the mppriority register is serviced ahead of any other transaction already in the single port queue.	RW	0x0

**Port Sum of Weight Register Register Descriptions**

This register is used to configure the DRAM burst operation scheduling.

Offset: 0xb0

**mpweight\_0\_4** on page 5-70

This register is used to configure the DRAM burst operation scheduling.

**mpweight\_1\_4** on page 5-71

This register is used to configure the DRAM burst operation scheduling.

**mpweight\_2\_4** on page 5-71

This register is used to configure the DRAM burst operation scheduling.

**mpweight\_3\_4** on page 5-72

This register is used to configure the DRAM burst operation scheduling.

**mpweight\_0\_4**

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250B0

Offset: 0x50B0

Access: RW

Bit Fields																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	staticweight_31_0																
RW 0x0																	staticweight_31_0															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	RW 0x0																

**mpweight\_0\_4 Fields**

Bit	Name	Description	Access	Reset
31:0	staticweight_31_0	Set static weight of the port. Each port is programmed with a 5 bit value. Port 0 is bits 4:0, port 1 is bits 9:5, up to port 9 being bits 49:45	RW	0x0

**mpweight\_1\_4**

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250B4

Offset: 0x50B4

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
sumofweights_13_0															staticweight_49_32	
RW 0x0															RW 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
staticweight_49_32															RW 0x0	

**mpweight\_1\_4 Fields**

Bit	Name	Description	Access	Reset
31:18	sumofweights_13_0	Set the sum of static weights for particular user priority. This register is used as part of the deficit round robin implementation. It should be set to the sum of the weights for the ports	RW	0x0
17:0	staticweight_49_32	Set static weight of the port. Each port is programmed with a 5 bit value. Port 0 is bits 4:0, port 1 is bits 9:5, up to port 9 being bits 49:45	RW	0x0

**mpweight\_2\_4**

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250B8

Offset: 0x50B8

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
sumofweights_45_14															RW 0x0	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
sumofweights_45_14															RW 0x0	

**mpweight\_2\_4 Fields**

Bit	Name	Description	Access	Reset
31:0	sumofweights_45_14	Set the sum of static weights for particular user priority. This register is used as part of the deficit round robin implementation. It should be set to the sum of the weights for the ports	RW	0x0

**mpweight\_3\_4**

This register is used to configure the DRAM burst operation scheduling.

Module Instance	Base Address	Register Address
sdr	0xFFC20000	0xFFC250BC

Offset: 0x50BC

Access: RW

Bit Fields																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved															sumofweights_63_46	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	RW 0x0
sumofweights_63_46																

**mpweight\_3\_4 Fields**

Bit	Name	Description	Access	Reset
17:0	sumofweights_63_46	Set the sum of static weights for particular user priority. This register is used as part of the deficit round robin implementation. It should be set to the sum of the weights for the ports	RW	0x0

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	<ul style="list-style-type: none"><li>Added information regarding calculation of ECC error byte address location from erraddr register in "User Notification of ECC Errors" sectoin</li><li>Added information regarding bus response to memory protection transaction failure in "Memory Protection" section</li><li>Clarified "Protection" row in "Fields for Rules in Memory Protection" table in the "Memory Protection" section</li><li>Clarified protruledata.security column in "Rules in Memory Protection Table for Example Configuration" table in the "Example of Configuration for TrustZone" section</li><li>Added note about double-bit error functionality in "ECC Write Backs" subsection of "ECC" section</li><li>Added the "DDR Calibration" subsection under "DDR PHY" section</li></ul>
May 2015	2015.05.04	<ul style="list-style-type: none"><li>Added the recommended sequence for writing or reading a rule in the "Memory Protection" section.</li></ul>
December 2014	2014.12.15	<ul style="list-style-type: none"><li>Added SDRAM Protection Access Flow Diagram to "Memory Protection" subsection in the "Single-Port Controller Operation" section.</li><li>Changed the "SDRAM Multi-Port Scheduling" section to "SDRAM Multi-Port Arbitration" and added detailed information on how to use and program the priority and weighted arbitration scheme.</li></ul>
June 2014	2014.6.30	<ul style="list-style-type: none"><li>Added <i>Port Mappings</i> section.</li><li>Added <i>SDRAM Controller Memory Options</i> section.</li><li>Enhanced <i>Example of Configuration for TrustZone</i> section.</li><li>Added SDRAM Controller address map and registers.</li></ul>
December 2013	2013.12.30	<ul style="list-style-type: none"><li>Added <i>Generating a Preloader Image for HPS with EMIF</i> section.</li><li>Added <i>Debugging HPS SDRAM in the Preloader</i> section.</li><li>Enhanced <i>Simulation</i> section.</li></ul>
November 2012	1.1	Added address map and register definitions section.
January 2012	1.0	Initial release.

# Functional Description—HPC II Controller

6

2016.05.02

EMI\_RM



Subscribe



Send Feedback

The High Performance Controller II works with the UniPHY-based DDR2, DDR3, and LPDDR2 interfaces. The controller provides high memory bandwidth, high clock rate performance, and run-time programmability. The controller can reorder data to reduce row conflicts and bus turn-around time by grouping reads and writes together, allowing for efficient traffic patterns and reduced latency.

**Note:** The controller described here is the High Performance Controller II (HPC II) with advanced features for designs generated in the Quartus II software version 11.0 and later, and the Quartus Prime software. Designs created in earlier versions and regenerated in version 11.0 and later do not inherit the new advanced features; for information on HPC II without the version 11.0 and later advanced features, refer to the External Memory Interface Handbook for Quartus II version 10.1, available in the External Memory Interfaces section of the Altera Literature website.

## Related Information

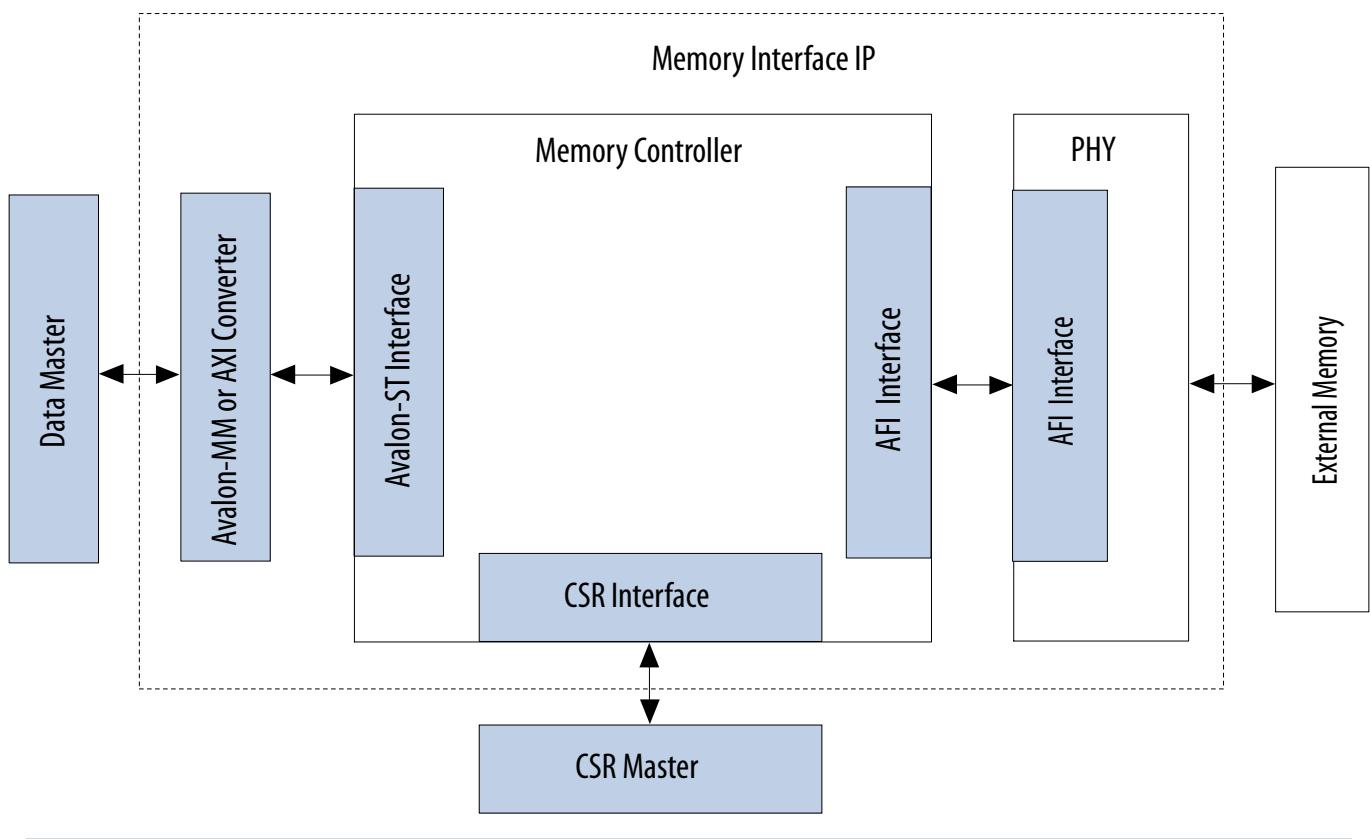
[External Memory Interface Handbook, v10.1](#)

## HPC II Memory Interface Architecture

The memory interface consists of the memory controller logic block, the physical logic layer (PHY), and their associated interfaces. The following figure shows a high-level block diagram of the overall external memory interface architecture.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

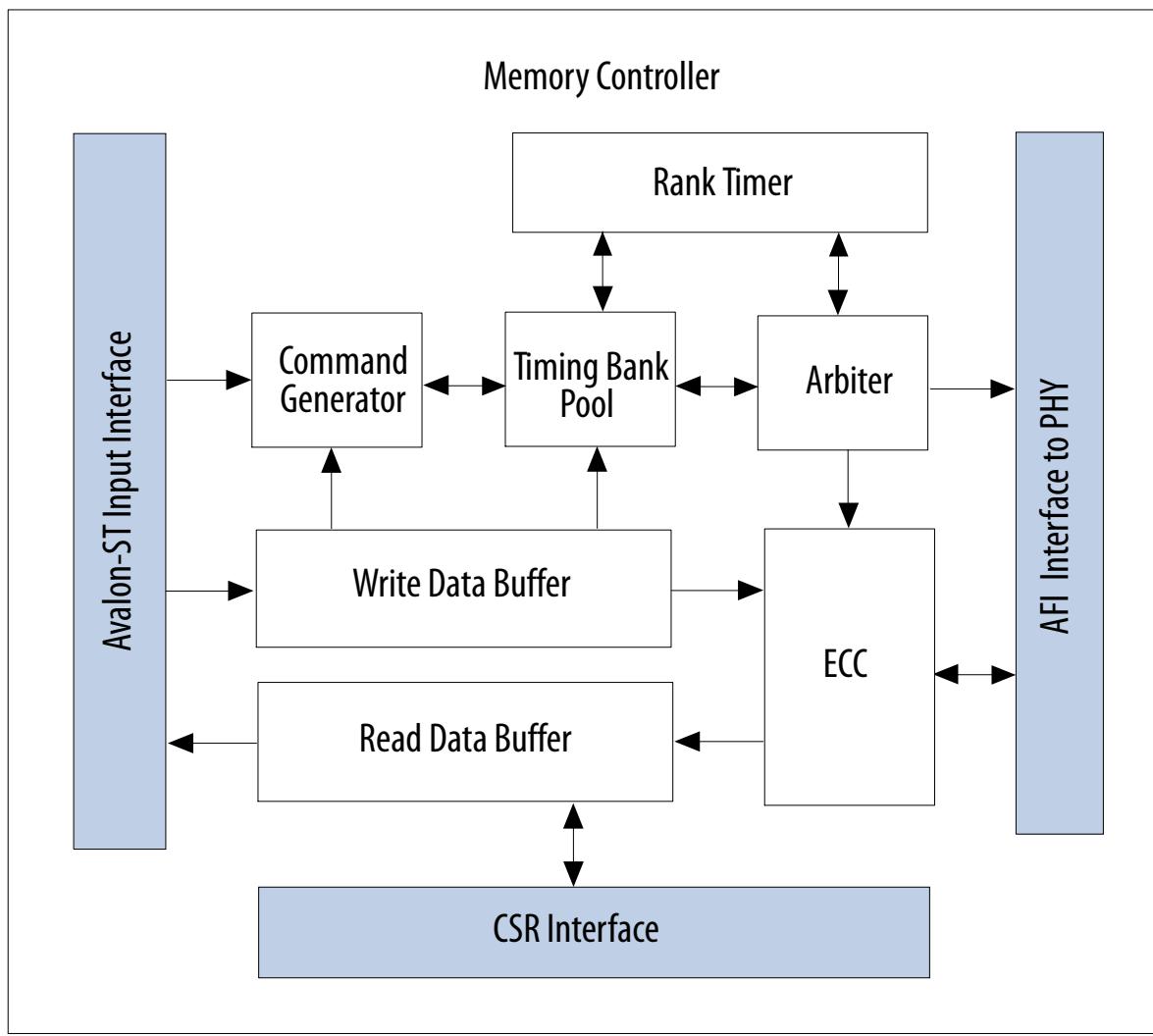
ISO  
9001:2008  
Registered

**Figure 6-1: High-Level Diagram of Memory Interface Architecture**

## HPC II Memory Controller Architecture

The memory controller logic block uses an Avalon Streaming (Avalon-ST) interface as its native interface, and communicates with the PHY layer by the Altera PHY Interface (AFI).

The following figure shows a block diagram of the memory controller architecture.

**Figure 6-2: Memory Controller Architecture Block Diagram**

### Avalon-ST Input Interface

The Avalon-ST interface serves as the entry point to the memory controller, and provides communication with the requesting data masters.

For information about the Avalon interface, refer to *Avalon Interface Specifications*.

### AXI to Avalon-ST Converter

The HPC II memory controller includes an AXI to Avalon-ST converter for communication with the AXI protocol. The AXI to Avalon-ST converter provides write address, write data, write response, read address, and read data channels on the AXI interface side, and command, write data, and read data channels on the Avalon-ST interface side.

### Handshaking

The AXI protocol employs a handshaking process similar to the Avalon-ST protocol, based on `ready` and `valid` signals.

## Command Channel Implementation

The AXI interface includes separate read and write channels, while the Avalon-ST interface has only one command channel. Arbitration of the read and write channels is based on these policies:

- Round robin
- Write priority—write channel has priority over read channel
- Read priority—read channel has priority over write channel

You can choose an arbitration policy by setting the `COMMAND_ARB_TYPE` parameter to one of `ROUND_ROBIN`, `WRITE_PRIORITY`, or `READ_PRIORITY` in the `alt_mem_ddrx_axi_st_converter.v` file.

## Data Ordering

The AXI specification requires that write data IDs must arrive in the same order as write address IDs are received. Similarly, read data must be returned in the same order as its associated read address is received.

Consequently, the AXI to Avalon-ST converter does not support interleaving of write data; all data must arrive in the same order as its associated write address IDs. On the read side, the controller returns read data based on the read addresses received.

## Burst Types

The AXI to Avalon-ST converter supports the following burst types:

- Incrementing burst—the address for each transfer is an increment of the previous transfer address; the increment value depends on the size of the transfer.
- Wrapping burst—similar to the incrementing burst, but wraps to the lower address when the burst boundary is reached. The starting address must be aligned to the size of the transfer. Burst length must be 2, 4, 8, or 16. The burst wrap boundary = burst size \* burst length.

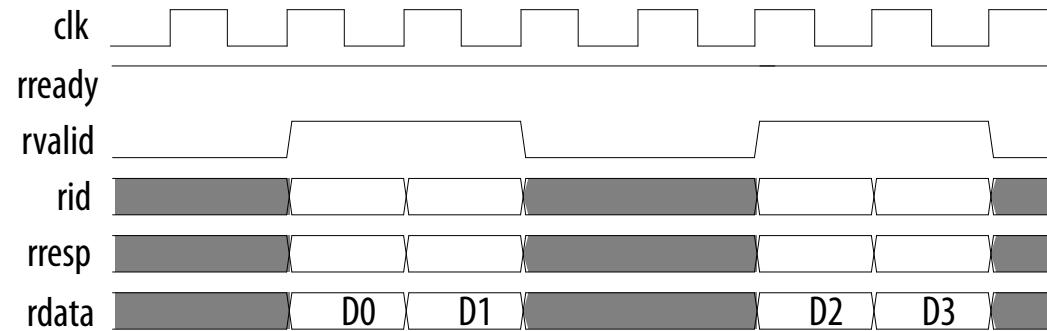
## Related Information

### Avalon Interface Specifications

## Backpressure Support

The write response and read data channels do not support data transfer with backpressure; consequently, you must assert the ready signal for the write response and read data channels to 1 to ensure acceptance of data at any time.

**Figure 6-3: Data Transfer Without Backpressure**



For information about data transfer with and without backpressure, refer to the *Avalon Interface Specifications*.

## Related Information

### Avalon Interface Specifications

## Command Generator

The command generator accepts commands from the front-end Avalon-ST interface and from local ECC internal logic, and provides those commands to the timing bank pool.

## Timing Bank Pool

The timing bank pool is a parallel queue that works with the arbiter to enable data reordering. The timing bank pool tracks incoming requests, ensures that all timing requirements are met and, upon receiving write-data-ready notification from the write data buffer, passes the requests to the arbiter in an ordered and efficient manner.

## Arbiter

The arbiter determines the order in which requests are passed to the memory device. When the arbiter receives a single request, that request is passed immediately; however, when multiple requests are received, the arbiter uses arbitration rules to determine the order in which to pass requests to the memory device.

### Arbitration Rules

The arbiter uses the following arbitration rules:

- If only one master is issuing a request, grant that request immediately.
- If there are outstanding requests from two or more masters, the arbiter applies the following tests, in order:
  - Is there a read request? If so, the arbiter grants the read request ahead of any write requests.
  - If neither of the above conditions apply, the arbiter grants the oldest request first.

## Rank Timer

The rank timer maintains rank-specific timing information, and performs the following functions:

- Ensures that only four activates occur within a specified timing window.
- Manages the read-to-write and write-to-read bus turnaround time.
- Manages the time-to-activate delay between different banks.

## Read Data Buffer and Write Data Buffer

The read data buffer receives data from the PHY and passes that data through the input interface to the master. The write data buffer receives write data from the input interface and passes that data to the PHY, upon approval of the write request.

## ECC Block

The error-correcting code (ECC) block comprises an encoder and a decoder-corrector, which can detect and correct single-bit errors, and detect double-bit errors. The ECC block can remedy errors resulting from noise or other impairments during data transmission.

## AFI and CSR Interfaces

The AFI interface provides communication between the controller and the physical layer logic (PHY). The CSR interface provides communication with your system's internal control status registers.

For more information about AFI signals, refer to *AFI 4.0 Specification* in the *Functional Description - UniPHY chapter*.

**Note:** Unaligned reads and writes on the AFI interface are not supported.

### Related Information

- [Avalon Interface Specifications](#)
- [Functional Description—UniPHY](#) on page 1-1

## HPC II Controller Features

The HPC II memory controller offers a variety of features.

### Data Reordering

The controller implements data reordering to maximize efficiency for read and write commands. The controller can reorder read and write commands as necessary to mitigate bus turn-around time and reduce conflict between rows.

Inter-bank data reordering reorders commands going to different bank addresses. Commands going to the same bank address are not reordered. This reordering method implements simple hazard detection on the bank address level.

The controller implements logic to limit the length of time that a command can go unserved. This logic is known as starvation control. In starvation control, a counter is incremented for every command served. You can set a starvation limit, to ensure that a waiting command is served immediately, when the starvation counter reaches the specified limit.

### Pre-emptive Bank Management

Data reordering allows the controller to issue bank-management commands pre-emptively, based on the patterns of incoming commands. The desired page in memory can be already open when a command reaches the AFI interface.

### Quasi-1T and Quasi-2T

One controller clock cycle equals two memory clock cycles in a half-rate interface, and to four memory clock cycles in a quarter-rate interface. To fully utilize the command bandwidth, the controller can operate in Quasi-1T half-rate and Quasi-2T quarter-rate modes.

In Quasi-1T and Quasi-2T modes, the controller issues two commands on every controller clock cycle. The controller is constrained to issue a row command on the first clock phase and a column command on the second clock phase, or vice versa. Row commands include activate and precharge commands; column commands include read and write commands.

The controller operates in Quasi-1T in half-rate mode, and in Quasi-2T in quarter-rate mode; this operation is transparent and has no user settings.

## User Autoprecharge Commands

The autoprecharge read and autoprecharge write commands allow you to indicate to the memory device that this read or write command is the last access to the currently open row. The memory device automatically closes or autoprecharges the page it is currently accessing so that the next access to the same bank is quicker.

This command is useful for applications that require fast random accesses.

Since the HPC II controller can reorder transactions for best efficiency, when you assert the `local_autopch_req` signal, the controller evaluates the current command and buffered commands to determine the best autoprecharge operation.

## Address and Command Decoding Logic

When the main state machine issues a command to the memory, it asserts a set of internal signals. The address and command decoding logic turns these signals into AFI-specific commands and address.

The following signals are generated:

- Clock enable and reset signals: `afi_cke`, `afi_rst_n`
- Command and address signals: `afi_cs_n`, `afi_ba`, `afi_addr`, `afi_ras_n`, `afi_cas_n`, `afi_we_n`

## Low-Power Logic

There are two types of low-power logic: the user-controlled self-refresh logic and automatic power-down with programmable time-out logic.

### User-Controlled Self-Refresh

When you assert the `local_self_rfsh_req` signal, the controller completes any currently executing reads and writes, and then interrupts the command queue and immediately places the memory into self-refresh mode. When the controller places the memory into self-refresh mode, it responds by asserting an acknowledge signal, `local_self_rfsh_ack`. You can leave the memory in self-refresh mode for as long as you choose.

To bring the memory out of self-refresh mode, you must deassert the request signal, and the controller responds by deasserting the acknowledge signal when the memory is no longer in self-refresh mode.

**Note:** If a user-controlled refresh request and a system-generated refresh request occur at the same time, the user-controlled refresh takes priority; the system-generated refresh is processed only after the user-controlled refresh request is completed.

### Automatic Power-Down with Programmable Time-Out

The controller automatically places the memory in power-down mode to save power if the requested number of idle controller clock cycles is observed in the controller. The **Auto Power Down Cycles** parameter on the **Controller Settings** tab allows you to specify a range between 1 to 65,535 idle controller clock cycles. The counter for the programmable time-out starts when there are no user read or write requests in the command queue. Once the controller places the memory in power-down mode, it responds by asserting the acknowledge signal, `local_power_down_ack`.

## ODT Generation Logic

The on-die termination (ODT) generation logic generates the necessary ODT signals for the controller, based on the scheme that Altera recommends.

**DDR2 SDRAM**

**Note:** There is no ODT for reads.

**Table 6-1: ODT—DDR2 SDRAM Single Slot Single Chip-select Per DIMM (Write)**

Write On	ODT Enabled
mem_cs[0]	mem_odt[0]

**Note:** There is no ODT for reads.

**Table 6-2: ODT—DDR2 SDRAM Single Slot Dual Chip-select Per DIMM (Write)**

Write On	ODT Enabled
mem_cs[0]	mem_odt[0]
mem_cs[1]	mem_odt[1]

**Table 6-3: ODT—DDR2 SDRAM Dual Slot Single Chip-select Per DIMM (Write)**

Write On	ODT Enabled
mem_cs[0]	mem_odt[1]
mem_cs[1]	mem_odt[0]

**Table 6-4: ODT—DDR2 SDRAM Dual Slot Dual Chip-select Per DIMM (Write)**

Write On	ODT Enabled
mem_cs[0]	mem_odt[2]
mem_cs[1]	mem_odt[3]
mem_cs[2]	mem_odt[0]
mem_cs[3]	mem_odt[1]

**DDR3 SDRAM**

**Note:** There is no ODT for reads.

**Table 6-5: ODT—DDR3 SDRAM Single Slot Single Chip-select Per DIMM (Write)**

Write On	ODT Enabled
mem_cs[0]	mem_odt[0]

**Note:** There is no ODT for reads.

**Table 6-6: ODT—DDR3 SDRAM Single Slot Dual Chip-select Per DIMM (Write)**

Write On	ODT Enabled
mem_cs[0]	mem_odt[0]
mem_cs[1]	mem_odt[1]

**Table 6-7: ODT—DDR3 SDRAM Dual Slot Single Chip-select Per DIMM (Write)**

Write On	ODT Enabled
mem_cs[0]	mem_odt[0] and mem_odt[1]
mem_cs[1]	mem_odt[0] and mem_odt[1]

**Table 6-8: ODT—DDR3 SDRAM Dual Slot Single Chip-select Per DIMM (Read)**

Read On	ODT Enabled
mem_cs[0]	mem_odt[1]
mem_cs[1]	mem_odt[0]

**Table 6-9: ODT—DDR3 SDRAM Dual Slot Dual Chip-select Per DIMM (Write)**

Write On	ODT Enabled
mem_cs[0]	mem_odt[0] and mem_odt[2]
mem_cs[1]	mem_odt[1] and mem_odt[3]

Write On	ODT Enabled
mem_cs[2]	mem_odt[0] and mem_odt[2]
mem_cs[3]	mem_odt[1] and mem_odt[3]

**Table 6-10: ODT—DDR3 SDRAM Dual Slot Dual Rank Per DIMM (Read)**

Read On	ODT Enabled
mem_cs[0]	mem_odt[2]
mem_cs[1]	mem_odt[3]
mem_cs[2]	mem_odt[0]
mem_cs[3]	mem_odt[1]

## Burst Merging

The burst merging feature improves controller efficiency by merging two burst chop commands of sequential addresses into one burst length command.

Burst merging is opportunistic and happens when the controller receives commands faster than it can process (for example, Avalon commands of multiple burst length) or when the controller temporarily stops processing commands due to Refresh.

The burst merging feature is turned off by default when you generate a controller. If your traffic exercises patterns that you can merge, you should turn on burst merging, as follows:

1. In a text editor, open the `<variation_name>.c0.v` top-level file for your design.
2. Search for the `ENABLE_BURST_MERGE` parameter in the `.v` file.
3. Change the `ENABLE_BURST_MERGE` value from 0 to 1.

## ECC

The ECC logic comprises an encoder and a decoder-corrector, which can detect and correct single-bit errors, and detect double-bit errors. The ECC logic is available in multiples of 16, 24, 40, and 72 bits.

**Note:** For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, ECC logic is limited to widths of 24 and 40.

- The ECC logic has the following features:
- Has Hamming code ECC logic that encodes every 64, 32, 16, or 8 bits of data into 72, 40, 24, or 16 bits of codeword.
- Has a latency increase of one clock for both writes and reads.
- For a 128-bit interface, ECC is generated as one 64-bit data path with 8-bits of ECC path, plus a second 64-bit data path with 8-bits of ECC path.
- Detects and corrects all single-bit errors.

- Detects all double-bit errors.
- Counts the number of single-bit and double-bit errors.
- Accepts partial writes, which trigger a read-modify-write cycle, for memory devices with DM pins.
- Can inject single-bit and double-bit errors to trigger ECC correction for testing and debugging purposes.
- Generates an interrupt signal when an error occurs.

**Note:** When using ECC, you must initialize your entire memory content to zero before beginning to write to the memory. If you do not initialize the content to zero, and if you read from uninitialized memory locations without having first written to them, you will see junk data which will trigger an ECC interrupt.

When a single-bit or double-bit error occurs, the ECC logic triggers the `ecc_interrupt` signal to inform you that an ECC error has occurred. When a single-bit error occurs, the ECC logic reads the error address, and writes back the corrected data. When a double-bit error occurs, the ECC logic does not do any error correction but it asserts the `avl_rdata_error` signal to indicate that the data is incorrect. The `avl_rdata_error` signal follows the same timing as the `avl_rdata_valid` signal.

Enabling autocorrection allows the ECC logic to delay all controller pending activities until the correction completes. You can disable autocorrection and schedule the correction manually when the controller is idle to ensure better system efficiency. To manually correct ECC errors, follow these steps:

1. When an interrupt occurs, read out the `SBE_ERROR` register. When a single-bit error occurs, the `SBE_ERROR` register is equal to one.
2. Read out the `ERR_ADDR` register.
3. Correct the single-bit error by issuing a dummy write to the memory address stored in the `ERR_ADDR` register. A dummy write is a write request with the `local_be` signal zero, that triggers a partial write which is effectively a read-modify-write event. The partial write corrects the data at that address and writes it back.

## Partial Writes

The ECC logic supports partial writes.

Along with the address, data, and burst signals, the Avalon-MM interface also supports a signal vector, `local_be`, that is responsible for byte-enable. Every bit of this signal vector represents a byte on the data-bus. Thus, a logic low on any of these bits instructs the controller not to write to that particular byte, resulting in a partial write. The ECC code is calculated on all bytes of the data-bus. If any bytes are changed, the IP core must recalculate the ECC code and write the new code back to the memory.

For partial writes, the ECC logic performs the following steps:

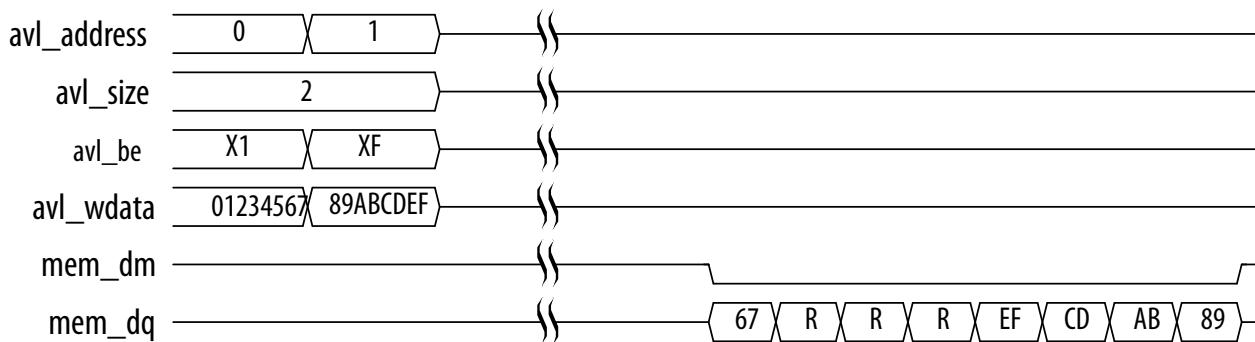
1. The ECC logic sends a read command to the partial write address.
2. Upon receiving a return data from the memory for the particular address, the ECC logic decodes the data, checks for errors, and then merges the corrected or correct dataword with the incoming information.
3. The ECC logic issues a write to write back the updated data and the new ECC code.

The following corner cases can occur:

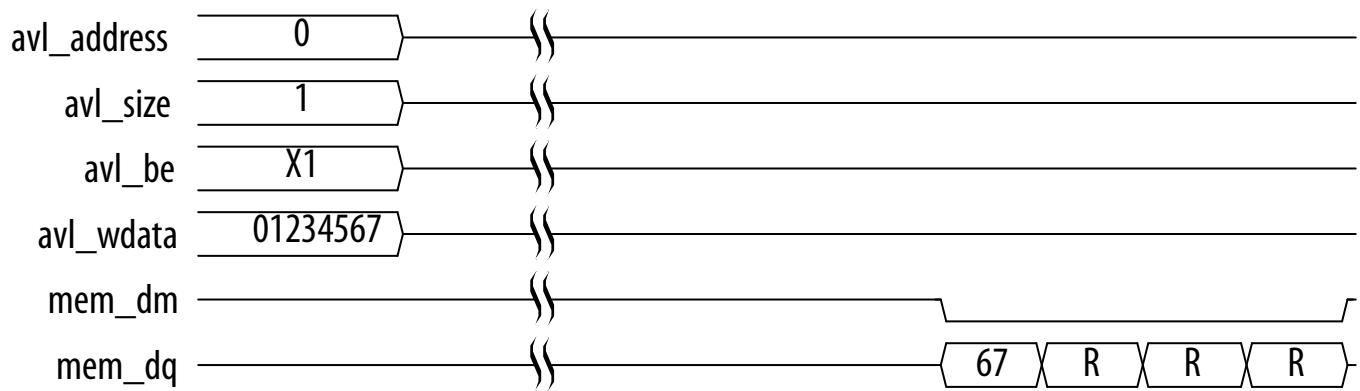
- A single-bit error during the read phase of the read-modify-write process. In this case, the IP core corrects the single-bit error first, increments the single-bit error counter and then performs a partial write to this corrected decoded data word.
- A double-bit error during the read phase of the read-modify-write process. In this case, the IP core increments the double-bit error counter and issues an interrupt. The IP core writes a new write word to the location of the error. The ECC status register keeps track of the error information.

The following figures show partial write operations for the controller, for full and half rate configurations, respectively.

**Figure 6-4: Partial Write for the Controller--Full Rate**



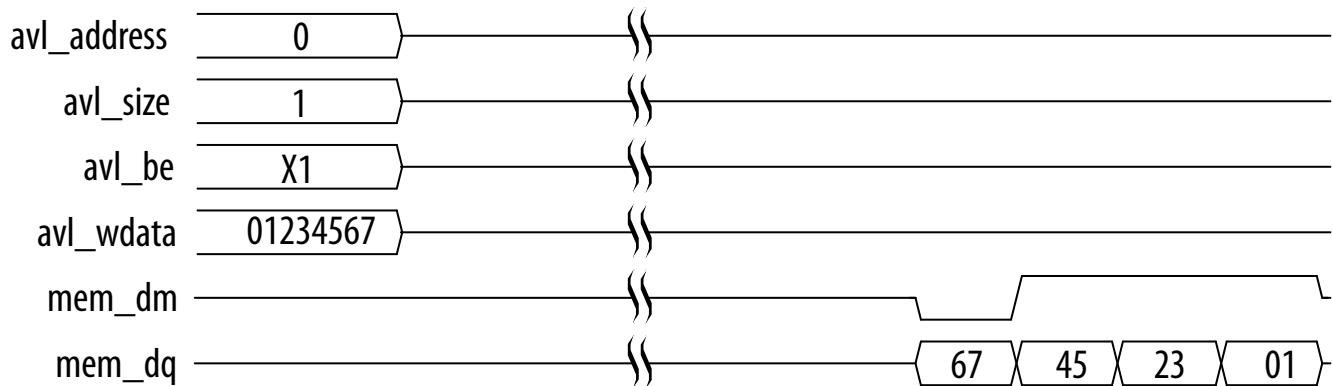
**Figure 6-5: Partial Write for the Controller--Half Rate**



## Partial Bursts

DIMMs that do not have the DM pins do not support partial bursts. You must write a minimum (or multiples) of memory-burst-length-equivalent words to the memory at the same time.

The following figure shows a partial burst operation for the controller.

**Figure 6-6: Partial Burst for Controller**

## External Interfaces

This section discusses the interfaces between the controller and other external memory interface components.

### Clock and Reset Interface

The clock and reset interface is part of the AFI interface.

The controller can have up to two clock domains, which are synchronous to each other. The controller operates with a single clock domain when there is no integrated half-rate bridge, and with two-clock domains when there is an integrated half-rate bridge. The clocks are provided by UniPHY.

The main controller clock is `afi_clk`, and the optional half-rate controller clock is `afi_half_clk`. The main and half-rate clocks must be synchronous and have a 2:1 frequency ratio. The optional quarter-rate controller clock is `afi_quarter_clk`, which must also be synchronous and have a 4:1 frequency ratio.

### Avalon-ST Data Slave Interface

The Avalon-ST data slave interface consists of the following Avalon-ST channels, which together form a single data slave:

- The command channel, which serves as command and address for both read and write operations.
- The write data channel, which carries write data.
- The read data channel, which carries read data.

#### Related Information

#### [Avalon Interface Specifications](#)

### AXI Data Slave Interface

The AXI data interface consists of the following channels, which communicate with the Avalon-ST interface through the AXI to Avalon-ST converter:

- The write address channel, which carries address information for write operations.
- The write data channel, which carries write data.
- The write response channel, which carries write response data.
- The read address channel, which carries address information for read operations.
- The read data channel, which carries read data.

## Enabling the AXI Interface

This section provides guidance for enabling the AXI interface.

1. To enable the AXI interface, first open in an editor the file appropriate for the required flow, as indicated below:
  - For synthesis flow: <working\_dir>/<variation\_name>/<variation\_name>.c0.v
  - For simulation flow: <working\_dir>/<variation\_name>.sim/<variation\_name>/<variation\_name>.c0.v
  - Example design fileset for synthesis: <working\_dir>/<variation\_name>.example\_design/<example\_project>/<variation\_name>.example/submodules/<variation\_name>.example\_if0\_c0.v
  - Example design fileset for simulation: <working\_dir>/<variation\_name>.example\_design/simulation/verilog/submodules/<variation\_name>.example\_sim\_e0\_if0\_c0.v
2. Locate and remove the **alt\_mem\_ddrx\_mm\_st\_converter** instantiation from the .v file opened in the preceding step.
3. Instantiate the **alt\_mem\_ddrx\_axi\_st\_converter** module into the open .v file. Refer to the following code fragment as a guide:

```

module ? # ( parameter
  // AXI parameters
  AXI_ID_WIDTH = <replace parameter value>,
  AXI_ADDR_WIDTH = <replace parameter value>,
  AXI_LEN_WIDTH = <replace parameter value>,
  AXI_SIZE_WIDTH = <replace parameter value>,
  AXI_BURST_WIDTH = <replace parameter value>,
  AXI_LOCK_WIDTH = <replace parameter value>,
  AXI_CACHE_WIDTH = <replace parameter value>,
  AXI_PROT_WIDTH = <replace parameter value>,
  AXI_DATA_WIDTH = <replace parameter value>,
  AXI_RESP_WIDTH = <replace parameter value>
)
(
// Existing ports
...
// AXI Interface ports
// Write address channel
input wire [AXI_ID_WIDTH - 1 : 0] awid,
input wire [AXI_ADDR_WIDTH - 1 : 0] awaddr,
input wire [AXI_LEN_WIDTH - 1 : 0] awlen,
input wire [AXI_SIZE_WIDTH - 1 : 0] awsize,
input wire [AXI_BURST_WIDTH - 1 : 0] awburst,
input wire [AXI_LOCK_WIDTH - 1 : 0] awlock,
input wire [AXI_CACHE_WIDTH - 1 : 0] awcache,
input wire [AXI_PROT_WIDTH - 1 : 0] awprot,
input wire awvalid,
output wire awready,
// Write data channel
input wire [AXI_ID_WIDTH - 1 : 0] wid,
input wire [AXI_DATA_WIDTH - 1 : 0] wdata,
input wire [AXI_DATA_WIDTH / 8 - 1 : 0] wstrb,
input wire wlast,
input wire wvalid,
output wire wready,
// Write response channel

```

```
output wire [AXI_ID_WIDTH - 1 : 0] bid,
output wire [AXI_RESP_WIDTH - 1 : 0] bresp,
output wire bvalid,
input wire bready,
// Read address channel
input wire [AXI_ID_WIDTH - 1 : 0] arid,
input wire [AXI_ADDR_WIDTH - 1 : 0] araddr,
input wire [AXI_LEN_WIDTH - 1 : 0] arlen,
input wire [AXI_SIZE_WIDTH - 1 : 0] arsize,
input wire [AXI_BURST_WIDTH - 1 : 0] arburst,
input wire [AXI_LOCK_WIDTH - 1 : 0] arlock,
input wire [AXI_CACHE_WIDTH - 1 : 0] arcache,
input wire [AXI_PROT_WIDTH - 1 : 0] arprot,
input wire arvalid,
output wire arready,
// Read data channel
output wire [AXI_ID_WIDTH - 1 : 0] rid,
output wire [AXI_DATA_WIDTH - 1 : 0] rdata,
output wire [AXI_RESP_WIDTH - 1 : 0] rresp,
output wire rlast,
output wire rvalid,
input wire rready
);
// Existing wire, register declaration and instantiation
...
// AXI interface instantiation
alt_mem_ddrx_axi_st_converter #
(
    .AXI_ID_WIDTH (AXI_ID_WIDTH ),
    .AXI_ADDR_WIDTH (AXI_ADDR_WIDTH ),
    .AXI_LEN_WIDTH (AXI_LEN_WIDTH ),
    .AXI_SIZE_WIDTH (AXI_SIZE_WIDTH ),
    .AXI_BURST_WIDTH (AXI_BURST_WIDTH ),
    .AXI_LOCK_WIDTH (AXI_LOCK_WIDTH ),
    .AXI_CACHE_WIDTH (AXI_CACHE_WIDTH ),
    .AXI_PROT_WIDTH (AXI_PROT_WIDTH ),
    .AXI_DATA_WIDTH (AXI_DATA_WIDTH ),
    .AXI_RESP_WIDTH (AXI_RESP_WIDTH ),
    .ST_ADDR_WIDTH (ST_ADDR_WIDTH ),
    .ST_SIZE_WIDTH (ST_SIZE_WIDTH ),
    .ST_ID_WIDTH (ST_ID_WIDTH ),
    .ST_DATA_WIDTH (ST_DATA_WIDTH ),
    .COMMAND_ARB_TYPE (COMMAND_ARB_TYPE)
)
a0
(
    .ctl_clk (afi_clk),
    .ctl_reset_n (afi_reset_n),
    .awid (awid),
    .awaddr (awaddr),
    .awlen (awlen),
    .awsizes (awsizes),
    .awburst (awburst),
    .awlock (awlock),
    .awcache (awcache),
    .awprot (awprot),
    .awvalid (awvalid),
    .awready (awready),
    .wid (wid),
    .wdata (wdata),
    .wstrb (wstrb),
    .wlast (wlast),
    .wvalid (wvalid),
    .wready (wready),
    .bid (bid),
    .bresp (bresp),
    .bvalid (bvalid),
```

```

.bready (bready),
.arid (arid),
.araddr (araddr),
.arlen (arlen),
.arsize (arsize),
.arburst (arburst),
.arlock (arlock),
.arcache (arcache),
.aprot (aprot),
.arvalid (arvalid),
.arready (arready),
.rid (rid),
.rdata (rdata),
.rresp (rresp),
.rlast (rlast),
.rvalid (rvalid),
.rready (rready),
.itf_cmd_ready (ng0_native_st_itf_cmd_ready),
.itf_cmd_valid (a0_native_st_itf_cmd_valid),
.itf_cmd (a0_native_st_itf_cmd),
.itf_cmd_address (a0_native_st_itf_cmd_address),
.itf_cmd_burstlen (a0_native_st_itf_cmd_burstlen),
.itf_cmd_id (a0_native_st_itf_cmd_id),
.itf_cmd_priority (a0_native_st_itf_cmd_priority),
.itf_cmd_autoprecharge (a0_native_st_itf_cmd_autopercharge),
.itf_cmd_multicast (a0_native_st_itf_cmd_multicast),
.itf_wr_data_ready (ng0_native_st_itf_wr_data_ready),
.itf_wr_data_valid (a0_native_st_itf_wr_data_valid),
.itf_wr_data (a0_native_st_itf_wr_data),
.itf_wr_data_byte_en (a0_native_st_itf_wr_data_byte_en),
.itf_wr_data_begin (a0_native_st_itf_wr_data_begin),
.itf_wr_data_last (a0_native_st_itf_wr_data_last),
.itf_wr_data_id (a0_native_st_itf_wr_data_id),
.itf_rd_data_ready (a0_native_st_itf_rd_data_ready),
.itf_rd_data_valid (ng0_native_st_itf_rd_data_valid),
.itf_rd_data (ng0_native_st_itf_rd_data),
.itf_rd_data_error (ng0_native_st_itf_rd_data_error),
.itf_rd_data_begin (ng0_native_st_itf_rd_data_begin),
.itf_rd_data_last (ng0_native_st_itf_rd_data_last),
.itf_rd_data_id (ng0_native_st_itf_rd_data_id)
);

```

4. Set the required parameters for the AXI interface. The following table lists the available parameters.
5. Export the AXI interface to the top-level wrapper, making it accessible to the AXI master.
6. To add the AXI interface to the Quartus Prime project:
  - On the Assignments > Settings menu in the Quartus Prime software, open the File tab.
  - Add the `alt_mem_ddrx_axi_st_converter.v` file to the project.

## AXI Interface Parameters

**Table 6-11: AXI Interface Parameters**

Parameter Name	Description / Value
AXI_ID_WIDTH	Width of the AXI ID bus. Default value is 4.

Parameter Name	Description / Value
AXI_ADDR_WIDTH	Width of the AXI address bus. Must be set according to the Avalon interface address and data bus width as shown below: $\text{AXI\_ADDR\_WIDTH} = \text{LOCAL\_ADDR\_WIDTH} + \log_2(\text{LOCAL\_DATA\_WIDTH}/8)$ LOCAL_ADDR_WIDTH is the memory controller Avalon interface address width. LOCAL_DATA_WIDTH is the memory controller Avalon data interface width.
AXI_LEN_WIDTH	Width of the AXI length bus. Default value is 8. Should be set to LOCAL_SIZE_WIDTH - 1, where LOCAL_SIZE_WIDTH is the memory controller Avalon interface burst size width
AXI_SIZE_WIDTH	Width of the AXI size bus. Default value is 3.
AXI_BURST_WIDTH	Width of the AXI burst bus. Default value is 2.
AXI_LOCK_WIDTH	Width of the AXI lock bus. Default value is 2.
AXI_CACHE_WIDTH	Width of the AXI cache bus. Default value is 4.
AXI_PROT_WIDTH	Width of the AXI protection bus. Default value is 3.
AXI_DATA_WIDTH	Width of the AXI data bus. Should be set to match the Avalon interface data bus width. $\text{AXI\_DATA\_WIDTH} = \text{LOCAL\_DATA\_WIDTH}$ , where LOCAL_DATA_WIDTH is the memory controller Avalon interface input data width.
AXI_RESP_WIDTH	Width of the AXI response bus. Default value is 2.
ST_ADDR_WIDTH	Width of the Avalon interface address. Must be set to match the Avalon interface address bus width. $\text{ST\_ADDR\_WIDTH} = \text{LOCAL\_ADDR\_WIDTH}$ , where LOCAL_ADDR_WIDTH is the memory controller Avalon interface address width.
ST_SIZE_WIDTH	Width of the Avalon interface burst size. $\text{ST\_SIZE\_WIDTH} = \text{AXI\_LEN\_WIDTH} + 1$
ST_ID_WIDTH	Width of the Avalon interface ID. Default value is 4. $\text{ST\_ID\_WIDTH} = \text{AXI\_ID\_WIDTH}$
ST_DATA_WIDTH	Width of the Avalon interface data. $\text{ST\_DATA\_WIDTH} = \text{AXI\_DATA\_WIDTH}$ .

Parameter Name	Description / Value
COMMAND_ARB_TYPE	Specifies the AXI command arbitration type, as shown: ROUND_ROBIN: arbitrates between read and write address channel in round robin fashion. Default option.WRITE_PRIORITY: write address channel has priority if both channels send request simultaneously. READ_PRIORITY: read address channel has priority if both channels send request simultaneously.
REGISTERED	Setting this parameter to 1 adds an extra register stage in the AXI interface and incurs one extra clock cycle of latency. Default value is 1.

## AXI Interface Ports

Table 5–12 lists the AXI interface ports.

**Table 6-12: AXI Interface Ports**

Name	Direction	Description
awid	Input	AXI write address channel ID bus.
awaddr	Input	AXI write address channel address bus.
awlen	Input	AXI write address channel length bus.
awszie	Input	AXI write address channel size bus.
awburst	Input	AXI write address channel burst bus.(Interface supports only INCR and WRAP burst types.)
awlock	Input	AXI write address channel lock bus.(Interface does not support this feature.)
awcache	Input	AXI write address channel cache bus.(Interface does not support this feature.)
awprot	Input	AXI write address channel protection bus.(Interface does not support this feature.)

Name	Direction	Description
awvalid	Input	AXI write address channel valid signal.
awready	Output	AXI write address channel ready signal.
wid	Input	AXI write address channel ID bus.
wdata	Input	AXI write address channel data bus.
wstrb	Input	AXI write data channel strobe bus.
wlast	Input	AXI write data channel last burst signal.
wvalid	Input	AXI write data channel valid signal.
wready	Output	AXI write data channel ready signal.
bid	Output	AXI write response channel ID bus.
bresp	Output	AXI write response channel response bus. Response encoding information: 'b00 - OKAY' 'b01 - Reserved' 'b10 - Reserved' 'b11 - Reserved'
bvalid	Output	AXI write response channel valid signal.
bready	Input	AXI write response channel ready signal. Must be set to 1. Interface does not support back pressure for write response channel.
arid	Input	AXI read address channel ID bus.
araddr	Input	AXI read address channel address bus.

Name	Direction	Description
arlen	Input	AXI read address channel length bus.
arsize	Input	AXI read address channel size bus.
arburst	Input	AXI read address channel burst bus.(Interface supports only INCR and WRAP burst types.)
arlock	Input	AXI read address channel lock bus.(Interface does not support this feature.)
arcache	Input	AXI read address channel cache bus.(Interface does not support this feature.)
arprot	Input	AXI read address channel protection bus.(Interface does not support this feature.)
arvalid	Input	AXI read address channel valid signal.
arready	Output	AXI read address channel ready signal.
rid	Output	AXI read data channel ID bus.
rdata	Output	AXI read data channel data bus.
rresp	Output	AXI read data channel response bus.Response encoding information: 'b00 - OKAY 'b01 - Reserved 'b10 - Data error 'b11 - Reserved
rlast	Output	AXI read data channel last burst signal.
rvalid	Output	AXI read data channel valid signal.
rready	Input	AXI read data channel ready signal.Must be set to 1. Interface does not support back pressure for write response channel.

For information about the AXI specification, refer to the ARM website, at [www.arm.com](http://www.arm.com).

#### Related Information

[www.arm.com](http://www.arm.com)

## Controller-PHY Interface

The interface between the controller and the PHY is part of the AFI interface. The controller assumes that the PHY performs all necessary calibration processes without any interaction with the controller.

For more information about AFI signals, refer to *AFI 4.0 Specification*.

## Memory Side-Band Signals

The HPC II controller supports several optional side-band signals.

### Self-Refresh (Low Power) Interface

The optional low power self-refresh interface consists of a request signal and an acknowledgement signal, which you can use to instruct the controller to place the memory device into self-refresh mode. This interface is clocked by `afi_clk`.

When you assert the request signal, the controller places the memory device into self-refresh mode and asserts the acknowledge signal. To bring the memory device out of self-refresh mode, you deassert the request signal; the controller then deasserts the acknowledge signal when the memory device is no longer in self-refresh mode.

**Note:** For multi-rank designs using the HPC II memory controller, a self-refresh and a user-refresh cannot be made to the same memory chip simultaneously. Also, the self-refresh `ack` signal indicates that at least one device has entered self-refresh, but does not necessarily mean that all devices have entered self-refresh.

### User-Controlled Refresh Interface

The optional user-controlled refresh interface consists of a request signal, a chip select signal, and an acknowledgement signal. This interface provides increased control over worst-case read latency and enables you to issue refresh bursts during idle periods. This interface is clocked by `afi_clk`.

When you assert a refresh request signal to instruct the controller to perform a refresh operation, that request takes priority over any outstanding read or write requests that might be in the command queue. In addition to the request signal, you must also choose the chip to be refreshed by asserting the refresh chip select signal along with the request signal. If you do not assert the chip select signal with the request signal, unexpected behavior may result.

The controller attempts to perform a refresh as long as the refresh request signal is asserted; if you require only one refresh, you should deassert the refresh request signal after the acknowledgement signal is received. If you maintain the request signal high after the acknowledgement is sent, it would indicate that further refresh is required. You should deassert the request signal after the required number of acknowledgement/refresh is received from the controller. You can issue up to a maximum of nine consecutive refresh commands.

**Note:** For multi-rank designs using the HPC II memory controller, a self-refresh and a user-refresh cannot be made to the same memory chip simultaneously.

### Configuration and Status Register (CSR) Interface

The controller has a configuration and status register (CSR) interface that allows you to configure timing parameters, address widths, and the behavior of the controller. The CSR interface is a 32-bit Avalon-MM slave of fixed address width; if you do not need this feature, you can disable it to save area.

This interface is clocked by `csr_clk`, which is the same as `afi_clk`, and is always synchronous relative to the main data slave interface.

## Controller External Interfaces

The following table lists the controller's external interfaces.

**Table 6-13: Summary of Controller External Interfaces**

Interface Name	Display Name	Type	Description
<b>Clock and Reset Interface</b>			
Clock and Reset Interface	Clock and Reset Interface	AFI <sup>(1)</sup>	Clock and reset generated by UniPHY to the controller.

### Avalon-ST Data Slave Interface

Command Channel	Avalon-ST Data Slave Interface	Avalon-ST <sup>(2)</sup>	Address and command channel for read and write, single command single data (SCSD).
Write Data Channel	Avalon-ST Data Slave Interface	Avalon-ST <sup>(2)</sup>	Write Data Channel, single command multiple data (SCMD).
Read Data Channel	Avalon-ST Data Slave Interface	Avalon-ST <sup>(2)</sup>	Read data channel, SCMD with read data error response.

### Controller-PHY Interface

AFI 4.0	AFI Interface	AFI <sup>(1)</sup>	Interface between controller and PHY.
---------	---------------	--------------------	---------------------------------------

### Memory Side-Band Signals

Self Refresh (Low Power) Interface	Self Refresh (Low Power) Interface	Avalon Control & Status Interface <sup>(2)</sup>	SDRAM-specific signals to place memory into low-power mode.
------------------------------------	------------------------------------	--	---

Interface Name	Display Name	Type	Description
User-Controller Refresh Interface	User-Controller Refresh Interface	Avalon Control & Status Interface <sup>(2)</sup>	SDRAM-specific signals to request memory refresh.

**Configuration and Status Register (CSR) Interface**

CSR	Configuration and Status Register Interface	Avalon-MM <sup>(2)</sup>	Enables on-the-fly configuration of memory timing parameters, address widths, and controller behaviour.
-----	---	--------------------------	---

Notes:

1. For information about AFI signals, refer to *AFI 4.0 Specification* in the *Functional Description - UniPHY chapter*.
2. For information about Avalon signals, refer to *Avalon Interface Specifications*.

**Related Information**

- [Avalon Interface Specifications](#)
- [Functional Description—UniPHY](#) on page 1-1

## Top-Level Signals Description

The top-level signals include clock and reset signals, local interface signals, controller interface signals, and CSR interface signals.

### Clock and Reset Signals

The following table lists the clock and reset signals.

**Note:** The suffix \_n denotes active low signals.

**Table 6-14: Clock and Reset Signals**

Name	Direction	Description
global_reset_n	Input	The asynchronous reset input to the controller. The IP core derives all other reset signals from resynchronized versions of this signal. This signal holds the PHY, including the PLL, in reset while low.
pll_ref_clk	Input	The reference clock input to PLL.

Name	Direction	Description
phy_clk	Output	The system clock that the PHY provides to the user. All user inputs to and outputs from the controller must be synchronous to this clock.
reset_phy_clk_n	Output	The reset signal that the PHY provides to the user. The IP core asserts <code>reset_phy_clk_n</code> asynchronously and deasserts synchronously to <code>phy_clk</code> clock domain.
aux_full_rate_clk	Output	An alternative clock that the PHY provides to the user. This clock always runs at the same frequency as the external memory interface. In half-rate designs, this clock is twice the frequency of the <code>phy_clk</code> and you can use it whenever you require a 2x clock. In full-rate designs, the same PLL output as the <code>phy_clk</code> signal drives this clock.
aux_half_rate_clk	Output	An alternative clock that the PHY provides to the user. This clock always runs at half the frequency as the external memory interface. In full-rate designs, this clock is half the frequency of the <code>phy_clk</code> and you can use it, for example to clock the user side of a half-rate bridge. In half-rate designs, or if the <b>Enable Half Rate Bridge</b> option is turned on. The same PLL output that drives the <code>phy_clk</code> signal drives this clock.
dll_reference_clk	Output	Reference clock to feed to an externally instantiated DLL.
reset_request_n	Output	Reset request output that indicates when the PLL outputs are not locked. Use this signal as a reset request input to any system-level reset controller you may have. This signal is always low when the PLL is trying to lock, and so any reset logic using Altera advises you detect a reset request on a falling edge rather than by level detection.
soft_reset_n	Input	Edge detect reset input for control by other system reset logic. Assert to cause a complete reset to the PHY, but not to the PLL that the PHY uses.

Name	Direction	Description
seriesterminationcontrol	Input (for OCT slave)	Required signal for PHY to provide series termination calibration value. Must be connected to a user-instantiated OCT control block (alt_oct) or another UniPHY instance that is set to OCT master mode.
	Output (for OCT master)	Unconnected PHY signal, available for sharing with another PHY.
parallelterminationcontrol	Input (for OCT slave)	Required signal for PHY to provide series termination calibration value. Must be connected to a user-instantiated OCT control block (alt_oct) or another UniPHY instance that is set to OCT master mode.
	Output (for OCT master)	Unconnected PHY signal, available for sharing with another PHY.
oct_rdn	Input (for OCT master)	Must connect to calibration resistor tied to GND on the appropriate RDN pin on the device. (Refer to appropriate device handbook.)
oct_rup	Input (for OCT master)	Must connect to calibration resistor tied to Vccio on the appropriate RUP pin on the device. (See appropriate device handbook.)
dqs_delay_ctrl_import	Input	Allows the use of DLL in another PHY instance in this PHY instance. Connect the export port on the PHY instance with a DLL to the import port on the other PHY instance.
csr_clk	Output	Clock for the configuration and status register (CSR) interface, which is the same as afi_clk and is always synchronous relative to the main data slave interface.

Note:

1. Applies only to the hard memory controller with multiport front end available in Arria V and Cyclone V devices.

## Local Interface Signals

The following table lists the controller local interface signals.

Table 6-15: Local Interface Signals

Signal Name	Direction	Description
avl_addr[] <sup>(1)</sup>	Input	<p>Memory address at which the burst should start. By default, the IP core maps local address to the bank interleaving scheme. You can change the ordering via the <b>Local-to-Memory Address Mapping</b> option in the <b>Controller Settings</b> page.</p> <p>This signal must remain stable only during the first transaction of a burst. The <code>constantBurstBehavior</code> property is always false for UniPHY controllers.</p> <p>The IP core sizes the width of this bus according to the following equations:</p> <ul style="list-style-type: none"> <li>• Full rate controllers:</li> </ul> <p>For one chip select: <math>\text{width} = \text{row bits} + \text{bank bits} + \text{column bits} - 1</math></p> <p>For multiple chip selects: <math>\text{width} = \text{chip bits}^* + \text{row bits} + \text{bank bits} + \text{column bits} - 1</math></p> <p>If the bank address is 2 bits wide, row is 13 bits wide and column is 10 bits wide, the local address is 24 bits wide. To map <code>local_address</code> to bank, row and column address:</p> <pre>avl_addr is 24 bits wide avl_addr[23:11] = row address[12:0] avl_addr[10:9] = bank address [1:0] avl_addr[8:0] = column address[9:1]</pre> <p>The IP core ignores the least significant bit (LSB) of the column address (multiples of two) on the memory side, because the local data width is twice that of the memory data bus width.</p> <ul style="list-style-type: none"> <li>• Half rate controllers:</li> </ul> <p>For one chip select: <math>\text{width} = \text{row bits} + \text{bank bits} + \text{column bits} - 2</math></p> <p>For multiple chip selects: <math>\text{width} = \text{chip bits}^* + \text{row bits} + \text{bank bits} + \text{column bits} - 2</math></p> <p>If the bank address is 2 bits wide, row is 13 bits wide and column is 10 bits wide, the local address is 23 bits wide. To map <code>local_address</code> to bank, row and column address:</p> <pre>avl_addr is 23 bits wide avl_addr[22:10] = row address[12:0] avl_addr[9:8] = bank address [1:0] avl_addr[7:0] = column address[9:2]</pre> <p>The IP core ignores two LSBs of the column address (multiples of four) on the memory side, because the local data width is four times that of the memory data bus width.</p> <ul style="list-style-type: none"> <li>• Quarter rate controllers:</li> </ul> <p>For one chip select: <math>\text{width} = \text{row bits} + \text{bank bits} + \text{column bits} - 3</math></p>

Signal Name	Direction	Description
		<p>For multiple chip selects: width = chip bits* + row bits + bank bits + column bits – 3</p> <p>If the bank address is 2 bits wide, row is 13 bits wide and column is 10 bits wide, the local address is 22 bits wide.</p> <p>(* <i>chip bits</i> is a derived value indicating the number of address bits necessary to uniquely address every memory rank in the system; this value is not user configurable.)</p> <ul style="list-style-type: none"> <li>• Full-rate hard memory controllers (Arria V and Cyclone V):</li> </ul> <p>For one chip select: width = row bits + bank bits + column bits - log2(local avalon data width/memory DQ width)</p> <p>For multiple chip selects: width = chip bits* + row bits + bank bits + column bits - log2(local avalon data width/memory data width)</p> <p>If the local Avalon data width is 32, the memory DQ width is 8, the bank address is 3 bits wide, the row is 12 bits wide and the column is 8 bits wide, the local address is 21 bits wide. To map <code>local_address</code> to bank, row and column address:</p> <p><code>avl_addr</code> is 21 bits wide</p> <pre>avl_addr[20:9] = row address[11:0] avl_addr[8:6] = bank address [2:0] avl_addr[5:0] = column address[7:2]</pre> <p>The IP core ignores the two least significant bits of the column address on the memory side because the local data width is four times that of the memory data bus width (Multi-Port Frontend).</p>
<code>avl_be[]</code> (2)	Input	<p>Byte enable signal, which you use to mask off individual bytes during writes. <code>avl_be</code> is active high; <code>mem_dm</code> is active low.</p> <p>To map <code>avl_wdata</code> and <code>avl_be</code> to <code>mem_dq</code> and <code>mem_dm</code>, consider a full-rate design with 32-bit <code>avl_wdata</code> and 16-bit <code>mem_dq</code>.</p> <pre>avl_wdata = &lt; 22334455 &gt;&lt; 667788AA &gt;&lt; BBCCDDEE&gt; avl_be = &lt; 1100 &gt;&lt; 0110 &gt;&lt; 1010 &gt;</pre> <p>These values map to:</p> <pre>Mem_dq = &lt;4455&gt;&lt;2233&gt;&lt;88AA&gt;&lt;6677&gt;&lt;DDEE&gt;&lt;BBCC&gt; Mem_dm = &lt;1 1 &gt;&lt;0 0 &gt;&lt;0 1 &gt;&lt;1 0 &gt;&lt;0 1 &gt;&lt;0 1 &gt;</pre>

Signal Name	Direction	Description
avl_burstbegin <sup>(3)</sup>	Input	<p>The Avalon burst begin strobe, which indicates the beginning of an Avalon burst. Unlike all other Avalon-MM signals, the burst begin signal is not dependant on <code>avl_ready</code>.</p> <p>For write transactions, assert this signal at the beginning of each burst transfer and keep this signal high for one cycle per burst transfer, even if the slave deasserts <code>avl_ready</code>. The IP core samples this signal at the rising edge of <code>phy_clk</code> when <code>avl_write_req</code> is asserted. After the slave deasserts the <code>avl_ready</code> signal, the master keeps all the write request signals asserted until <code>avl_ready</code> signal becomes high again.</p> <p>For read transactions, assert this signal for one clock cycle when read request is asserted and <code>avl_addr</code> from which the data should be read is given to the memory. After the slave deasserts <code>avl_ready</code> (<code>waitrequest_n</code> in Avalon interface), the master keeps all the read request signals asserted until <code>avl_ready</code> becomes high again.</p>
avl_read_req <sup>(4)</sup>	Input	Read request signal. You cannot assert read request and write request signals at the same time. The controller must deassert <code>reset_phy_clk_n</code> before you can assert <code>avl_read_req</code> .
local_refresh_req	Input	User-controlled refresh request. If <b>Enable User Auto-Refresh Controls</b> option is turned on, <code>local_refresh_req</code> becomes available and you are responsible for issuing sufficient refresh requests to meet the memory requirements. This option allows complete control over when refreshes are issued to the memory including grouping together multiple refresh commands. Refresh requests take priority over read and write requests, unless the IP core is already processing the requests.
local_refresh_chip	Input	<p>Controls which chip to issue the user refresh to. The IP core uses this active high signal with <code>local_refresh_req</code>. This signal is as wide as the memory chip select. This signal asserts a high value to each bit that represents the refresh for the corresponding memory chip.</p> <p>For example: If <code>local_refresh_chip</code> signal is assigned with a value of <code>4'b0101</code>, the controller refreshes the memory chips 0 and 2, and memory chips 1 and 3 are not refreshed.</p>
avl_size[] <sup>(5)</sup>	Input	<p>Controls the number of beats in the requested read or write access to memory, encoded as a binary number. In UniPHY, the IP core supports Avalon burst lengths from 1 to 1024. The IP core derives the width of this signal based on the burst count that you specify in the <b>Maximum Avalon-MM burst length</b> option. With the derived width, you specify a value ranging from 1 to the local maximum burst count specified.</p> <p>This signal must remain stable only during the first transaction of a burst. The <code>constantBurstBehavior</code> property is always false for UniPHY controllers.</p>

Signal Name	Direction	Description
avl_wdata[ ] <sup>(6)</sup>	Input	Write data bus. The width of <code>avl_wdata</code> is twice that of the memory data bus for a full-rate controller, four times the memory data bus for a half-rate controller, and eight times the memory data bus for a quarter-rate controller. If <b>Generate power-of-2 data bus widths for Qsys and SOPC Builder</b> is enabled, the width is rounded down to the nearest power of 2.
avl_write_req <sup>(7)</sup>	Input	Write request signal. You cannot assert read request and write request signal at the same time. The controller must deassert <code>reset_phy_clk_n</code> before you can assert <code>avl_write_req</code> .
local_autopch_req <sup>(8)</sup>	Input	<p>User control of autoprecharge. If you turn on <b>Enable Auto-Precharge Control</b>, the <code>local_autopch_req</code> signal becomes available and you can request the controller to issue an autoprecharge write or autoprecharge read command.</p> <p>These commands cause the memory to issue a precharge command to the current bank at the appropriate time without an explicit precharge command from the controller. This feature is particularly useful if you know the current read or write is the last one you intend to issue to the currently open row. The next time you need to use that bank, the access could be quicker as the controller does not need to precharge the bank before activating the row you wish to access.</p> <p>Upon receipt of the <code>local_autopch_req</code> signal, the controller evaluates the pending commands in the command buffer and determines the most efficient autoprecharge operation to perform, reordering commands if necessary.</p> <p>The controller must deassert <code>reset_phy_clk_n</code> before you can assert <code>local_autopch_req</code>.</p>
local_self_rfsh_chip	Input	<p>Controls which chip to issue the user refresh to. The IP core uses this active high signal with <code>local_self_rfsh_req</code>. This signal is as wide as the memory chip select. This signal asserts a high value to each bit that represents the refresh for the corresponding memory chip.</p> <p>For example: If <code>local_self_rfsh_chip</code> signal is assigned with a value of <code>4'b0101</code>, the controller refreshes the memory chips 0 and 2, and memory chips 1 and 3 are not refreshed.</p>

Signal Name	Direction	Description
local_self_rfsh_req	Input	User control of the self-refresh feature. If you turn on <b>Enable Self-Refresh Controls</b> , you can request that the controller place the memory devices into a self-refresh state by asserting this signal. The controller places the memory in the self-refresh state as soon as it can without violating the relevant timing parameters and responds by asserting local_self_rfsh_ack. You can hold the memory in the self-refresh state by keeping this signal asserted. You can release the memory from the self-refresh state at any time by deasserting local_self_rfsh_req and the controller responds by deasserting local_self_rfsh_ack when it has successfully brought the memory out of the self-refresh state.
local_init_done	Output	When the memory initialization, training, and calibration are complete, the PHY sequencer asserts ctrl_usr_mode_rdy to the memory controller, which then asserts this signal to indicate that the memory interface is ready for use.
local_cal_success	Output	When the memory initialization, training, and calibration completes successfully, the controller asserts this signal coincident with local_init_done to indicate the memory interface is ready for use.
local_cal_fail	Output	When the memory initialization, training, or calibration fails, the controller asserts this signal to indicate that calibration failed. The local_init_done signal will not assert when local_cal_fail asserts.
avl_rdata[] <sup>(9)</sup>	Output	Read data bus. The width of avl_rdata is twice that of the memory data bus for a full rate controller; four times the memory data bus for a half rate controller. If <b>Generate power-of-2 data bus widths for Qsys and SOPC Builder</b> is enabled, the width is rounded down to the nearest power of 2.
avl_rdata_error <sup>(10)</sup>	Output	Asserted if the current read data has an error. This signal is only available if you turn on <b>Enable Error Detection and Correction Logic</b> . The controller asserts this signal with the avl_rdata_valid signal.  If the controller encounters double-bit errors, no correction is made and the controller asserts this signal.
avl_rdata_valid <sup>(11)</sup>	Output	Read data valid signal. The avl_rdata_valid signal indicates that valid data is present on the read data bus.

Signal Name	Direction	Description
av1_ready <sup>(12)</sup>	Output	<p>The <code>av1_ready</code> signal indicates that the controller is ready to accept request signals. If controller asserts the <code>av1_ready</code> signal in the clock cycle that it asserts a read or write request, the controller accepts that request. The controller deasserts the <code>av1_ready</code> signal to indicate that it cannot accept any more requests. The controller can buffer eight read or write requests, after which the <code>av1_ready</code> signal goes low.</p> <p>The <code>av1_ready</code> signal is deasserted when any of the following are true:</p> <ul style="list-style-type: none"><li>• The Timing bank Pool is full.</li><li>• The FIFO register that stores read data from the memory device is full.</li><li>• The write data FIFO register is full.</li><li>• The controller is waiting for write data when in ECC mode.</li></ul>
local_refresh_ack	Output	Refresh request acknowledge, which the controller asserts for one clock cycle every time it issues a refresh. Even if you do not turn on <b>Enable User Auto-Refresh Controls</b> , <code>local_refresh_ack</code> still indicates to the local interface that the controller has just issued a refresh command.
local_self_rfsh_ack	Output	Self refresh request acknowledge signal. The controller asserts and deasserts this signal in response to the <code>local_self_rfsh_req</code> signal.
local_power_down_ack	Output	Auto power-down acknowledge signal. The controller asserts this signal for one clock cycle every time auto power-down is issued.
ecc_interrupt <sup>(13)</sup>	Output	Interrupt signal from the ECC logic. The controller asserts this signal when the ECC feature is turned on, and the controller detects an error.

Signal Name	Direction	Description
-------------	-----------	-------------

Notes to Table:

1. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_addr becomes a per port value, avl\_addr\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
2. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_be becomes a per port value, avl\_be\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
3. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_burstbegin becomes a per port value, avl\_burstbegin\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
4. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_read\_req becomes a per port value, avl\_read\_req\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
5. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_size becomes a per port value, avl\_size\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
6. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_wdata becomes a per port value, avl\_wdata\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
7. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_write\_req becomes a per port value, avl\_write\_req\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
8. This signal is not applicable to the hard memory controller.
9. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_rdata becomes a per port value, avl\_rdata\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
10. This signal is not applicable to the hard memory controller.
11. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_rdata\_valid becomes a per port value, avl\_rdata\_valid\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
12. For the hard memory controller with multiport front end available in Arria V and Cyclone V devices, avl\_ready becomes a per port value, avl\_ready\_#, where # is a numeral from 0–5, based on the number of ports selected in the Controller tab.
13. This signal is not applicable to the hard memory controller.

## Controller Interface Signals

The following table lists the controller interface signals.

**Table 6-16: Interface Signals**

Signal Name	Direction	Description
mem_dq[ ]	Bidirectional	Memory data bus. This bus is half the width of the local read and write data busses.

Signal Name	Direction	Description
mem_dqs[ ]	Bidirectional	Memory data strobe signal, which writes data into the memory device and captures read data into the Altera device.
mem_dqs_n[ ]	Bidirectional	Inverted memory data strobe signal, which with the mem_dqs signal improves signal integrity.
mem_ck	Output	Clock for the memory device.
mem_ck_n	Output	Inverted clock for the memory device.
mem_addr[ ]	Output	Memory address bus.
mem_ac_parity <sup>(1)</sup>	Output	Address or command parity signal generated by the PHY and sent to the DIMM. DDR3 SDRAM only.
mem_ba[ ]	Output	Memory bank address bus.
mem_cas_n	Output	Memory column address strobe signal.
mem_cke[ ]	Output	Memory clock enable signals.
mem_cs_n[ ]	Output	Memory chip select signals.
mem_dm[ ]	Output	Memory data mask signal, which masks individual bytes during writes.
mem_odt	Output	Memory on-die termination control signal.
mem_ras_n	Output	Memory row address strobe signal.
mem_we_n	Output	Memory write enable signal.
parity_error_n <sup>(1)</sup>	Output	This signal is not used and should be ignored. The PHY does not have capture circuitry to trigger on the falling edge of mem_error_out_n. See below for a complete description of mem_error_out_n, and recommendations.

Signal Name	Direction	Description
mem_err_out_n <sup>(1)</sup>	Input	This is an output of registered DIMMs. When an address-and-command parity error is detected, DDR3 registered DIMMs assert the mem_err_out_n signal in accordance with the memory buffer configuration. Unlike ECC on the data bus, the controller does not automatically correct errors on the address-and-command bus. You should connect this pin to your own falling edge detection circuitry in order to capture when a parity error occurs. Upon error detection, action may be taken such as causing a system interrupt, or an appropriate event.

Note:

1. This signal is for registered DIMMs only.

## CSR Interface Signals

The following table lists the CSR interface signals.

**Table 6-17: CSR Interface Signals**

Signal Name	Direction	Description
csr_addr[ ]	Input	Register map address. The width of csr_addr is 16 bits.
csr_be[ ]	Input	Byte-enable signal, which you use to mask off individual bytes during writes. csr_be is active high.
csr_clk <sup>(1)</sup>	Output	Clock for the configuration and status register (CSR) interface, which is the same as afi_clk and is always synchronous relative to the main data slave interface.
csr_wdata[ ]	Input	Write data bus. The width of csr_wdata is 32 bits.
csr_write_req	Input	Write request signal. You cannot assert csr_write_req and csr_read_req signals at the same time.
csr_read_req	Input	Read request signal. You cannot assert csr_read_req and csr_write_req signals at the same time.
csr_rdata[ ]	Output	Read data bus. The width of csr_rdata is 32 bits.
csr_rdata_valid	Output	Read data valid signal. The csr_rdata_valid signal indicates that valid data is present on the read data bus.

Signal Name	Direction	Description
csr_waitrequest	Output	The <code>csr_waitrequest</code> signal indicates that the HPC II is busy and not ready to accept request signals. If the <code>csr_waitrequest</code> signal goes high in the clock cycle when a read or write request is asserted, that request is not accepted. If the <code>csr_waitrequest</code> signal goes low, the HPC II is then ready to accept more requests.

Note to Table:

1. Applies only to the hard memory controller with multiport front end available in Arria V and Cyclone V devices.

## Soft Controller Register Map

The soft controller register map allows you to control the soft memory controller settings.

**Note:** Dynamic reconfiguration is not currently supported.

The following table lists the register map for the controller.

**Table 6-18: Soft Controller Register Map**

Address	Bit	Name	Default	Access	Description
0x100	0	Reserved.	0	—	Reserved for future use.
	1	Reserved.	0	—	Reserved for future use.
	2	Reserved.	0	—	Reserved for future use.
	7:3	Reserved.	0	—	Reserved for future use.
	13:8	Reserved.	0	—	Reserved for future use.
	30:14	Reserved.	0	—	Reserved for future use.

Address	Bit	Name	Default	Access	Description
0x110	15:0	AUTO_PD_CYCLES	0x0	Read write	The number of idle clock cycles after which the controller should place the memory into power-down mode. The controller is considered to be idle if there are no commands in the command queue. Setting this register to 0 disables the auto power-down mode. The default value of this register depends on the values set during the generation of the design.
	16	Reserved.	0	—	Reserved for future use.
	17	Reserved.	0	—	Reserved for future use.
	18	Reserved.	0	—	Reserved for future use.
	19	Reserved.	0	—	Reserved for future use.
	21:20	ADDR_ORDER	00	Read write	00 - Chip, row, bank, column.01 - Chip, bank, row, column.10 - reserved for future use.11 - Reserved for future use.
	22	Reserved.	0	—	Reserved for future use.
	24:23	Reserved.	0	—	Reserved for future use.
	30:24	Reserved	0	—	Reserved for future use.

Address	Bit	Name	Default	Access	Description
0x120	7:0	Column address width	—	Read write	The number of column address bits for the memory devices in your memory interface. The range of legal values is 7-12.
	15:8	Row address width	—	Read write	The number of row address bits for the memory devices in your memory interface. The range of legal values is 12-16.
	19:16	Bank address width	—	Read write	The number of bank address bits for the memory devices in your memory interface. The range of legal values is 2-3.
	23:20	Chip select address width	—	Read write	The number of chip select address bits for the memory devices in your memory interface. The range of legal values is 0-2. If there is only one single chip select in the memory interface, set this bit to 0.
	31:24	Reserved.	0	—	Reserved for future use.
0x121	31:0	Data width representation (word)	—	Read only	The number of DQS bits in the memory interface. This bit can be used to derive the width of the memory interface by multiplying this value by the number of DQ pins per DQS pin (typically 8).
0x122	7:0	Chip select representation	—	Read only	The number of chip select in binary representation. For example, a design with 2 chip selects has the value of 00000011.
	31:8	Reserved.	0	—	Reserved for future use.

Address	Bit	Name	Default	Access	Description
0x123	3:0	tRCD	—	Read write	The activate to read or write a timing parameter. The range of legal values is 2-11 cycles.
	7:4	tRRD	—	Read write	The activate to activate a timing parameter. The range of legal values is 2-8 cycles.
	11:8	tRP	—	Read write	The precharge to activate a timing parameter. The range of legal values is 2-11 cycles.
	15:12	tMRD	—	Read write	The mode register load time parameter. This value is not used by the controller, as the controller derives the correct value from the memory type setting.
	23:16	tRAS	—	Read write	The activate to precharge a timing parameter. The range of legal values is 4-29 cycles.
	31:24	tRC	—	Read write	The activate to activate a timing parameter. The range of legal values is 8-40 cycles.
0x124	3:0	tWTR	—	Read write	The write to read a timing parameter. The range of legal values is 1-10 cycles.
	7:4	tRTP	—	Read write	The read to precharge a timing parameter. The range of legal values is 2-8 cycles.
	15:8	tFAW	—	Read write	The four-activate window timing parameter. The range of legal values is 6-32 cycles.
	31:16	Reserved.	0	—	Reserved for future use.
0x125	15:0	tREFI	—	Read write	The refresh interval timing parameter. The range of legal values is 780-6240 cycles.
	23:16	tRFC	—	Read write	The refresh cycle timing parameter. The range of legal values is 12-255 cycles.
	31:24	Reserved.	0	—	Reserved for future use.

Address	Bit	Name	Default	Access	Description
0x126	3:0	Reserved.	0	—	Reserved for future use.
	7:4	Reserved.	0	—	Reserved for future use.
	11:8	Reserved.	0	—	Reserved for future use.
	15:12	Reserved.	0	—	Reserved for future use.
	23:16	Burst Length	—	Read write	Value must match memory burst length.
	31:24	Reserved.	0	—	Reserved for future use.

Address	Bit	Name	Default	Access	Description
0x13 0	0	ENABLE_ECC	1	Read write	When this bit equals 1, it enables the generation and checking of ECC. This bit is only active if ECC was enabled during IP parameterization.
	1	ENABLE_AUTO_CORR	—	Read write	When this bit equals 1, it enables auto-correction when a single-bit error is detected.
	2	GEN_SBE	0	Read write	When this bit equals 1, it enables the deliberate insertion of single-bit errors, bit 0, in the data written to memory. This bit is used only for testing purposes.
	3	GEN_DBE	0	Read write	When this bit equals 1, it enables the deliberate insertion of double-bit errors, bits 0 and 1, in the data written to memory. This bit is used only for testing purposes.
	4	ENABLE_INTR	1	Read write	When this bit equals 1, it enables the interrupt output.
	5	MASK_SBE_INTR	0	Read write	When this bit equals 1, it masks the single-bit error interrupt.
	6	MASK_DBE_INTR	0	Read write	When this bit equals 1, it masks the double-bit error interrupt
	7	CLEAR	0	Read write	When this bit equals 1, writing to this self-clearing bit clears the interrupt signal, and the error status and error address registers.
	8	MASK_CORDROP_INTR	0	Read write	When this bit equals 1, the dropped autocorrection error interrupt is dropped.
	9	Reserved.	0	—	Reserved for future use.

Address	Bit	Name	Default	Access	Description
0x13	0	SBE_ERROR	0	Read only	Set to 1 when any single-bit errors occur.
	1	DBE_ERROR	0	Read only	Set to 1 when any double-bit errors occur.
	2	CORDROP_ERROR	0	Read only	Value is set to 1 when any controller-scheduled autocorrections are dropped.
	7:3	Reserved.	0	—	Reserved for future use.
	15:8	SBE_COUNT	0	Read only	Reports the number of single-bit errors that have occurred since the status register counters were last cleared.
	23:16	DBE_COUNT	0	Read only	Reports the number of double-bit errors that have occurred since the status register counters were last cleared.
	31:24	CORDROP_COUNT	0	Read only	Reports the number of controller-scheduled autocorrections dropped since the status register counters were last cleared.
	31:0	ERR_ADDR	0	Read only	The address of the most recent ECC error. This address is a memory burst-aligned local address.
	31:0	CORDROP_ADDR	0	Read only	The address of the most recent autocorrection that was dropped. This is a memory burst-aligned local address.
	0	REORDER_DATA	—	Read write	
0x13	15:1	Reserved.	0	—	Reserved for future use.
	23:16	STARVE_LIMIT	0	Read write	Number of commands that can be served before a starved command.
	31:24	Reserved.	0	—	Reserved for future use.

## Hard Controller Register Map

The hard controller register map allows you to control the hard memory controller settings.

**Note:** Dynamic reconfiguration is not currently supported.

The following table lists the register map for the hard controller.

**Table 6-19: Hard Controller Register Map**

Address	Bit	Name	Default	Access	Description
0x00 0	3:0	CFG_CAS_WR_LAT	0	Read/Write	Memory write latency.
0x00 1	4:0	CFG_ADD_LAT	0	Read/Write	Memory additive latency.
0x00 2	4:0	CFG_TCL	0	Read/Write	Memory read latency.
0x00 3	3:0	CFG_TRRD	0	Read/Write	The activate to activate different banks timing parameter.
0x00 4	5:0	CFG_TFAW	0	Read/Write	The four-activate window timing parameter.
0x00 5	7:0	CFG_TRFC	0	Read/Write	The refresh cycle timing parameter.
0x00 6	12:0	CFG_TREFI	0	Read/Write	The refresh interval timing parameter.
0x00 8	3:0	CFG_TREFI	0	Read/Write	The activate to read/write timing parameter.
0x00 9	3:0	CFG_TRP	0	Read/Write	The precharge to activate timing parameter.
0x00 A	3:0	CFG_TWR	0	Read/Write	The write recovery timing.
0x00 B	3:0	CFG_TWTR	0	Read/Write	The write to read timing parameter.
0x00 C	3:0	CFG_TRTP	0	Read/Write	The read to precharge timing parameter.
0x00 D	4:0	CFG_TRAS	0	Read/Write	The activate to precharge timing parameter.
0x00 E	5:0	CFG_TRC	0	Read/Write	The activate to activate timing parameter.
0x00 F	15:0	CFG_AUTO_PD_CYCLES	0	Read/Write	The number of idle clock cycles after which the controller should place the memory into power-down mode.
0x01 1	9:0	CFG_SELF_RFSH_EXIT_CYCLES	0	Read/Write	The self-refresh exit cycles.

Address	Bit	Name	Default	Access	Description
0x01	9:0	CFG_PDN_EXIT_CYCLES	0	Read/Write	The power down exit cycles.
0x01	5	CFG_TMRD	0	Read/Write	Mode register timing parameter.
0x01	6	CFG_COL_ADDR_WIDTH	0	Read/Write	The number of column address bits for the memory devices in your memory interface.
0x01	7	CFG_ROW_ADDR_WIDTH	0	Read/Write	The number of row address bits for the memory devices in your memory interface.
0x01	8	CFG_BANK_ADDR_WIDTH	0	Read/Write	The number of bank address bits for the memory devices in your memory interface.
0x01	9	CFG_CS_ADDR_WIDTH	0	Read/Write	The number of chip select address bits for the memory devices in your memory interface.
0x03	5	CFG_TCCD	0	Read/Write	CAS#-to-CAS# command delay.
0x03	5	CFG_WRITE_ODT_CHIP	0	Read/Write	CAS#-to-CAS# command delay.
0x03	7	CFG_READ_ODT_CHIP	0	Read/Write	Read ODT Control.

Address	Bit	Name	Default	Access	Description
0x040	2:0	CFG_TYPE	0	Read/Write	Selects memory type.
	7:3	CFG_BURST_LENGTH	0	Read/Write	Configures burst length as a static decimal value.
	9:8	CFG_ADDR_ORDER	0	Read/Write	Address order selection.
	10	CFG_ENABLE_ECC	0	Read/Write	Enable the generation and checking of ECC.
	11	CFG_ENABLE_AUTO_CORR	0	Read/Write	Enable auto correction when single bit error is detected.
	12	CFG_GEN_SBE	0	Read/Write	When this bit equals 1, it enables the deliberate insertion of single-bit errors, bit 0, in the data written to memory. This bit is used only for testing purposes.
	13	CFG_GEN_DBE	0	Read/Write	When this bit equals 1, it enables the deliberate insertion of double-bit errors, bits 0 and 1, in the data written to memory. This bit is used only for testing purposes.
	14	CFG_REORDER_DATA	0	Read/Write	Enable Data Reordering.
	15	CFG_USER_RFSH	0	Read/Write	Enable User Refresh.
	16	CFG_REGDIMM_ENABLE	0	Read/Write	REG DIMM Configuration.
	17	CFG_ENABLE_DQS_TRACKING	0	Read/Write	Enable DQS Tracking.
	18	CFG_OUTPUT_REGD	0	Read/Write	Enable Registered Output.
	19	CFG_ENABLE_NO_DM	0	Read/Write	No Data Mask Configuration.
	20	CFG_ENABLE_ECC_CODE_OVERWRITES	0	Read/Write	Enable ECC Code Overwrite in Double Error Bit Detection.
0x043	7:0	CFG_INTERFACE_WIDTH	0	Read/Write	Memory Interface Width.
0x044	3:0	CFG_DEVICE_WIDTH	0	Read/Write	Memory Device Width.
0x045	0	CFG_CAL_REQ	0	Read/Write	Request re-calibration.
	6:1	CFG_CLOCK_OFF	0	Read/Write	Disable memory clock.
0x047	0	STS_CAL_SUCCESS	0	Read Only	Calibration Success.
	1	STS_CAL_FAIL	0	Read Only	Calibration Fail.
	2	STS_SBE_ERROR	0	Read Only	Single Bit Error Detected.
	3	STS_DBE_ERROR	0	Read Only	Double Bit Error Detected.
	4	STS_CORR_DROPPED	0	Read Only	Auto Correction Dropped.

Address	Bit	Name	Default	Access	Description
0x048	0	CFG_ENABLE_INTR	0	Read/Write	Enable Interrupt
	1	CFG_MASK_SBE_INTR	0	Read/Write	Mask Single Bit Error Interrupt.
	2	CFG_MASK_DBE_INTR	0	Read/Write	Mask Double Bit Error Interrupt.
	3	CFG_MASK_DBE_INTR	0	Write Clear	Clear Interrupt.
0x049	7:0	STS_SBD_COUNT	0	Read Only	Reports the number of SBE errors that have occurred since the status register counters were last cleared.
0x04A	7:0	STS_DBE_COUNT	0	Read Only	Reports the number of SBE errors that have occurred since the status register counters were last cleared.
0x04B	31:0	STS_ERR_ADDR	0	Read Only	The address of the most recent ECC error.
0x04F	0	CFG_MASK_CORR_DROPPED_INTR	0	Read/Write	Auto Correction Dropped Count.
0x050	7:0	CFG_MASK_CORR_DROPPED_INTR	0	Read Only	Auto Correction Dropped Count.
0x051	31:0	STS_CORR_DROPPED_ADDR	0	Read Only	Auto Correction Dropped Address.
0x055	5:0	CFG_STARVE_LIMIT	0	Read/Write	Starvation Limit.
0x056	1:0	CFG_MEM_BL	0	Read/Write	Burst Length.
	2	CFG_MEM_BL	0	Read/Write	ECC Enable.
0x057	1:0	CFG_MEM_BL	0	Read/Write	Specifies controller interface width.
0x058	11:0	CMD_PORT_WIDTH	0	Read/Write	Specifies per command port data width.
0x05A	11:0	CMD_FIFO_MAP	0	Read/Write	Specifies command port to Write FIFO association.
0x05C	11:0	CFG_CPORT_RFIFO_MAP	0	Read/Write	Specifies command port to Read FIFO association.
	23:12	CFG_RFIFO_CPORT_MAP	0	Read/Write	Port assignment (0 - 5) associated with each of the N FIFO.
	31:24	CFG_WFIFO_CPORT_MAP	0	Read/Write	Port assignment (0 - 5) associated with each of the N FIFO. (con't)
0x05D	3:0	CFG_WFIFO_CPORT_MAP	0	Read/Write	Port assignment (0 - 5) associated with each of the N FIFO.
	14:4	CFG_CPORT_TYPE	0	Read/Write	Command port type.

Address	Bit	Name	Default	Access	Description
0x06 2	2:0	CFG_CLOSE_TO_FULL	0	Read/Write	Indicates when the FIFO has this many empty entries left.
	5:3	CFG_CLOSE_TO_EMPTY	0	Read/Write	Indicates when the FIFO has this many valid entries left.
	11:6	CFG_CLOSE_TO_EMPTY	0	Read/Write	Port works in synchronous mode.
	12	CFG_INC_SYNC	0	Read/Write	Set the number of FF as clock synchronizer.
0x06 7	5:0	CFG_ENABLE_BONDING	0	Read/Write	Enables bonding for each of the control ports.
	7:6	CFG_DELAY_BONDING	0	Read/Write	Set to the value used for the bonding input to bonding output delay.
0x06 9	5:0	CFG_AUTO_PCH_ENABLE	0	Read/Write	Control auto-precharge options.
0x06 A	17:0	MP_SCHEDULER_PRIORITY	0	Read/Write	Set absolute user priority of the port
0x06 D	29:0	RCFG_ST_WT	0	Read/Write	Set static weight of the port.
	31:30	RCFG_SUM_PRI_WT	0	Read/Write	Set the sum of static weights for particular user priority.
0x06 E	31:0	RCFG_SUM_PRI_WT	0	Read/Write	Set the sum of static weights for particular user priority.
0x06 F	29:0	RCFG_SUM_PRI_WT	0	Read/Write	Set the sum of static weights for particular user priority.
0x0B9	0	CFG_DISABLE_MERGING	0	Read/Write	Set to a one to disable command merging.

## Sequence of Operations

Various blocks pass information in specific ways in response to write, read, and read-modify-write commands.

### Write Command

When a requesting master issues a write command together with write data, the following events occur:

- The input interface accepts the write command and the write data.
- The input interface passes the write command to the command generator and the write data to the write data buffer.
- The command generator processes the command and sends it to the timing bank pool.
- Once all timing requirements are met and a write-data-ready notification has been received from the write data buffer, the timing bank pool sends the command to the arbiter.

- When rank timing requirements are met, the arbiter grants the command request from the timing bank pool and passes the write command to the AFI interface.
- The AFI interface receives the write command from the arbiter and requests the corresponding write data from the write data buffer.
- The PHY receives the write command and the write data, through the AFI interface.

## Read Command

When a requesting master issues a read command, the following events occur:

- The input interface accepts the read command.
- The input interface passes the read command to the command generator.
- The command generator processes the command and sends it to the timing bank pool.
- Once all timing requirements are met, the timing bank pool sends the command to the arbiter.
- When rank timing requirements are met, the arbiter grants the command request from the timing bank pool and passes the read command to the AFI interface.
- The AFI interface receives the read command from the arbiter and passes the command to the PHY.
- The PHY receives the read command through the AFI interface, and returns read data through the AFI interface.
- The AFI interface passes the read data from the PHY to the read data buffer.
- The read data buffer sends the read data to the master through the input interface.

## Read-Modify-Write Command

A read-modify-write command can occur when enabling ECC for partial write, and for ECC correction commands. When a read-modify-write command is issued, the following events occur:

- The command generator issues a read command to the timing bank pool.
- The timing bank pool and arbiter passes the read command to the PHY through the AFI interface.
- The PHY receives the read command, reads data from the memory device, and returns the read data through the AFI interface.
- The read data received from the PHY passes to the ECC block.
- The read data is processed by the write data buffer.
- When the write data buffer issues a read-modify-write data ready notification to the command generator, the command generator issues a write command to the timing bank pool. The arbiter can then issue the write request to the PHY through the AFI interface.
- When the PHY receives the write request, it passes the data to the memory device.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	<ul style="list-style-type: none"><li>• Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li><li>• Added <code>CFG_GEN_SBE</code> and <code>CFG_GEN_DBE</code> to <i>Hard Controller Register Map</i> table.</li></ul>
May 2015	2015.05.04	Maintenance release.

Date	Version	Changes
December 2014	2014.12.15	<ul style="list-style-type: none"> <li>Renamed <i>Controller Register Map</i> to <i>Soft Controller Register Map</i>.</li> <li>Added <i>Hard Controller Register Map</i>.</li> </ul>
August 2014	2014.08.15	<ul style="list-style-type: none"> <li>Added "asynchronous" to descriptions of <code>mp_cmd_reset_n</code>, <code>#_reset_n</code>, <code>mp_rfifo_reset_n</code>, <code>#_reset_n</code>, and <code>mp_wfifo_reset_n</code>, <code>#_reset_n</code> signals in the <i>MPFE Signals</i> table.</li> <li>Added <i>Reset</i> description to <i>Hard Memory Controller</i> section.</li> <li>Added full-rate hard memory controller information for Arria V and Cyclone V to description of <code>avl_addr[ ]</code> in the <i>Local Interface Signals</i> table.</li> <li>Reworded <code>avl_burstbegin</code> description in the <i>Local Interface Signals</i> table.</li> </ul>
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>Removed references to ALTMEMPHY.</li> <li>Removed references to SOPC Builder.</li> <li>Removed Half-Rate Bridge information.</li> <li>Modified Burst Merging description.</li> <li>Expanded description of <code>avl_ready</code> in Local Interface Signals table.</li> <li>Added descriptions of <code>local_cal_success</code> and <code>local_cal_fail</code> to Local Interface Signals table.</li> <li>Modified description of <code>avl_size</code> in Local Interface Signals table.</li> <li>Added guidance to initialize memory before use.</li> </ul>
November 2012	2.1	<ul style="list-style-type: none"> <li>Added Controller Register Map information.</li> <li>Added Burst Merging information.</li> <li>Updated User-Controlled Refresh Interface information.</li> <li>Changed chapter number from 4 to 5.</li> </ul>
June 2012	2.0	<ul style="list-style-type: none"> <li>Added LPDDR2 support.</li> <li>Added Feedback icon.</li> </ul>
November 2011	1.1	<ul style="list-style-type: none"> <li>Revised Figure 5-1.</li> <li>Added AXI to Avalon-ST Converter information.</li> <li>Added AXI Data Slave Interface information.</li> <li>Added Half-Rate Bridge information.</li> </ul>

# Functional Description—QDR II Controller

7

2016.05.02

EMI\_RM



Subscribe



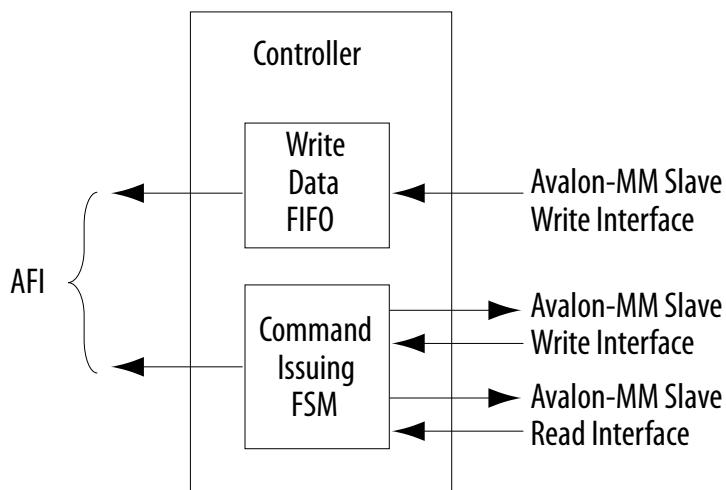
Send Feedback

The QDR II, QDR II+, and QDR II+ Xtreme controller translates memory requests from the Avalon Memory-Mapped (Avalon-MM) interface to AFI, while satisfying timing requirements imposed by the memory configuration. QDR II, QDR II+, and QDR II+ Xtreme SRAM has unidirectional data buses, therefore read and write operations are highly independent of each other and each has its own interface and state machine.

## Block Description

The following figure shows a block diagram of the QDR II, QDR II+, and QDR II+ Xtreme SRAM controller architecture.

**Figure 7-1: QDR II, QDR II+, and QDR II+ Xtreme SRAM Controller Architecture Block Diagram**



## Avalon-MM Slave Read and Write Interfaces

The read and write blocks accept read and write requests, respectively, from the Avalon-MM interface. Each block has a simple state machine that represents the state of the command and address registers, which stores the command and address when a request arrives.

The read data passes through without the controller registering it, as the PHY takes care of read latency. The write data goes through a pipeline stage to delay for a fixed number of cycles as specified by the write

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

latency. In the full-rate burst length of four controller, the write data is also multiplexed into a burst of 2, which is then multiplexed again in the PHY to become a burst of 4 in DDR.

The user interface to the controller has separate read and write Avalon-MM interfaces because reads and writes are independent of each other in the memory device. The separate channels give efficient use of available bandwidth.

## Command Issuing FSM

The command-issuing finite-state machine (FSM) has two states: `INIT` and `INIT_COMPLETE`. In the `INIT_COMPLETE` state, commands are issued immediately as requests arrive using combinational logic and do not require state transitions.

## AFI

The QDR II, QDR II+, and QDR II+ Xtreme controller communicates with the PHY using the AFI interface.

In the full-rate burst-length-of-two configuration, the controller can issue both read and write commands in the same clock cycle. In the memory device, both commands are clocked on the positive edge, but the read address is clocked on the positive edge, while the write address is clocked on the negative edge. Care must be taken on how these signals are ordered in the AFI.

For the half-rate burst-length-of-four configuration, the controller also issues both read and write commands, but the AFI width is doubled to fill two memory clocks per controller clock. Because the controller issues only one write command and one read command per controller clock, the AFI read and write signals corresponding to the other memory cycle are tied to no operation (NOP).

For the full-rate burst-length-of-four configuration, the controller alternates between issuing read and write commands every clock cycle. The memory device requires two clock cycles to complete the burst-length-of-four operation and requires an interleaving of read and write commands.

For information on the AFI, refer to *AFI 4.0 Specification* in chapter 1, *Functional Description - UniPHY*.

### Related Information

- [AFI 4.0 Specification](#) on page 2-52
- [Functional Description—UniPHY](#) on page 1-1

## Avalon-MM and Memory Data Width

The following table lists the data width ratio between the memory interface and the Avalon-MM interface.

The half-rate controller does not support burst-of-2 devices because it under-uses the available memory bandwidth. Regardless of full or half-rate decision and the device burst length, the Avalon-MM interface must supply all the data for the entire memory burst in a single clock cycle. Therefore the Avalon-MM data width of the full-rate controller with burst-of-4 devices is four times the memory data width. For width-expanded configurations, the data width is further multiplied by the expansion factor (not shown in table 5-1 and 5-2).

**Table 7-1: Data Width Ratio**

Memory Burst Length	Half-Rate Designs	Full-Rate Designs
QDR II 2-word burst	No Support	2:1
QDR II, QDR II+, and QDR II+ Extreme 4-word burst		4:1

## Signal Descriptions

The following tables lists the signals of the controller's Avalon-MM slave interface.

**Table 7-2: Avalon-MM Slave Read Signals**

UniPHY Signal Name	Arria 10 Signal Name	Width (UniPHY)	Width (Arria 10)	Direction	Avalon-MM Signal Type
avl_r_ready	amm_ready_0	1	1	Out	waitrequest_n
avl_r_read_req	amm_read_0	1	1	In	read
avl_r_addr	amm_address_0	15–25	17–23	In	address
avl_r_rdata_valid	amm_readdatavalid_0	1	1	Out	readdata-valid
avl_r_rdata	amm_readdata_0	9, 18, 36 (or 8, 16, 32 if power-of-2-bus is enabled in the controller)	9, 18, 36 (or 8, 16, 32 if power-of-2-bus is enabled in the controller)	Out	readdata
avl_r_size	amm_burstcount_0	$\log_2(\text{MAX\_BURST\_SIZE}) + 1$	$\text{ceil}(\log_2(\text{CTRL\_QDR2\_AVL\_MAX\_BURST\_COUNT} + 1))$	In	burstcount

**Note:** To obtain the actual signal width, you must multiply the widths in the above table by the data width ratio and the width expansion ratio.

**Table 7-3: Avalon-MM Slave Write Signals**

UniPHY Signal Name	Arria 10 Signal Name	Width (UniPHY)	Width (Arria 10)	Direction	Avalon-MM Signal Type
avl_w_ready	amm_ready_1	1	1	Out	waitrequest_n
avl_w_write_req	amm_write_1	1	1	In	write
avl_w_addr	amm_address_1	15–25	17–23	In	address
avl_w_wdata	amm_writedata_1	9, 18, 36 (or 8, 16, 32 if power-of-2-bus is enabled in the controller)	9, 18, 36 (or 8, 16, 32 if power-of-2-bus is enabled in the controller)	In	writedata
avl_w_be	amm_byteenable_1	1,2,4	1,2,4	In	byteenable
avl_w_size	amm_burstcount_1	$\log_2(\text{MAX\_BURST\_SIZE}) + 1$	$\text{ceil}(\log_2(\text{CTRL\_QDR2\_AVL\_MAX\_BURST\_COUNT} + 1))$	In	burstcount

**Note:** To obtain the actual signal width, you must multiply the widths in the above table by the data width ratio and the width expansion ratio.

#### Related Information

#### [Avalon Interface Specifications](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.

Date	Version	Changes
August 2014	2014.08.15	<ul style="list-style-type: none"><li>Added QDR II+ Xtreme throughout the chapter.</li><li>Added full-rate, burst-length-of-four information to <i>AFI</i> section.</li><li>Revised <i>Avalon-MM Slave Read Signals</i> table to include Arria 10 information.</li><li>Revised <i>Avalon-MM Slave Write Signals</i> table to include Arria 10 information.</li></ul>
December 2013	2013.12.16	Removed references to SOPC Builder.
November 2012	3.3	Changed chapter number from 5 to 6.
June 2012	3.2	Added Feedback icon.
November 2011	3.1	Harvested Controller chapter from <i>11.0 QDR II and QDR II+ SRAM Controller with UniPHY User Guide</i> .

# Functional Description—QDR-IV Controller

8

2016.05.02

EMI\_RM



Subscribe



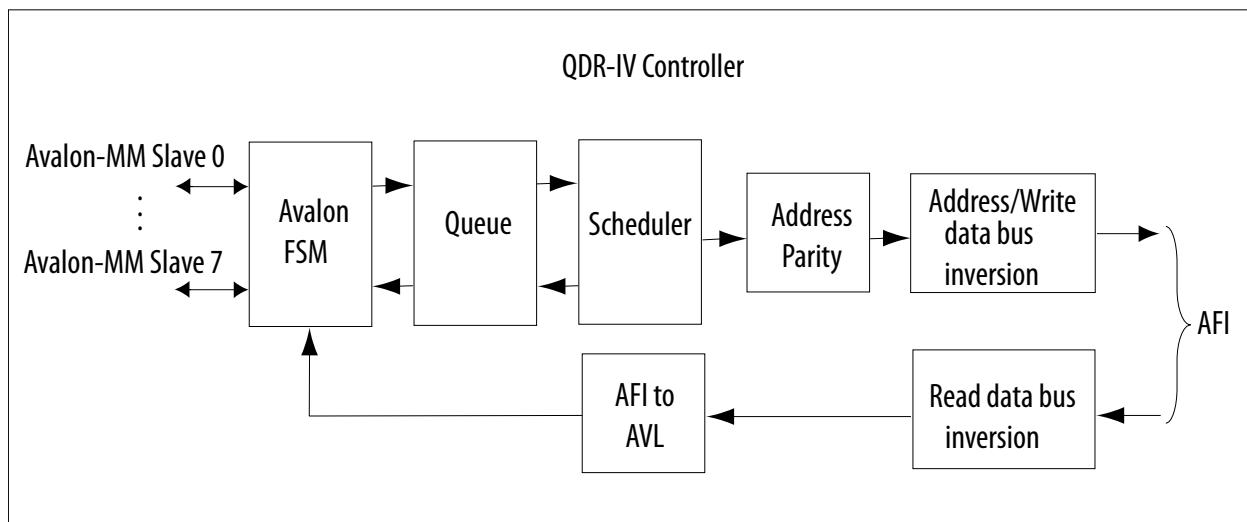
Send Feedback

The QDR-IV controller translates memory requests from the Avalon Memory-Mapped (Avalon-MM) interface to AFI, while satisfying timing requirements imposed by the memory configuration. QDR-IV has two independent bidirectional data ports, each of which can perform read and write operations on each memory clock cycle, subject to bank address and bus turnaround restrictions. To maximize data bus utilization, the QDR-IV controller provides eight separate Avalon interfaces, one for each QDR-IV data port and time slot (at quarter rate).

## Block Description

The following figure shows a block diagram of the QDR-IV controller architecture.

**Figure 8-1: QDR-IV Controller Architecture Block Diagram**



**Note:** To achieve maximum performance for a QDR-IV interface on an Arria 10 device, read data bus inversion must be enabled.

## Avalon-MM Slave Read and Write Interfaces

To maximize available bandwidth utilization, the QDR-IV external memory controller provides eight separate bidirectional Avalon interfaces—one channel for each of the two QDR-IV data ports in each of

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

the four memory time slots. At the memory device, the PHY ensures that port A commands are issued on the rising edge of the clock, and port B commands are issued on the falling edge of the clock.

The Avalon finite state machine (FSM) implements the standard Avalon-MM interface for each channel. New commands arrive in a queue, before being sent to the PHY by the scheduler. The scheduler ensures that bank policy and bus turnaround restrictions are met. The controller handles address/data bus inversion, if you have enabled those features.

The controller schedules commands in a simple round-robin fashion, always maintaining the time slot relationship shown in the following table. At each AFI cycle, the scheduler attempts to issue up to eight commands—one from each channel. If a particular command cannot be issued due to bank policy or bus turnaround violations, only commands from the preceding channels are issued. The remaining commands are issued in the following one or more cycles, depending on whether further violations occur. Read data passes through without the controller registering it, after AFI to Avalon bus conversion. The PHY implements read latency requirements.

**Table 8-1: Avalon-MM to Memory Time Slot and Port Mapping**

Avalon-MM Slave Interface	Memory Time Slot	Memory Port
0	0	A
1	0	B
2	1	A
3	1	B
4	2	A
5	2	B
6	3	A
7	3	B

## AFI

The QDR-IV controller communicates with the PHY using the AFI interface.

The controller supports a quarter-rate burst-length-of-two configuration, and can issue up to eight read and/or write commands per controller clock cycle. The controller may have to reduce the actual number of commands issued to avoid banking and bus turnaround violations. If you use QDR-IV devices with banked operation, you cannot access the same bank in the same memory clock cycle. Write latency is much shorter than read latency, therefore you must be careful not to place write data at the same time that read data is driven on the bus, on the same port. In addition, when switching between read and write operations, a delay may be required to avoid signal reflection and allow enough time for dynamic on-chip termination (OCT) to work. If necessary, the scheduler in the controller can delay the issuing of commands to avoid these issues.

For information on the AFI, refer to *AFI 4.0 Specification* in *Functional Description - Arria 10 EMIF IP*.

### Related Information

- [AFI 4.0 Specification](#) on page 2-52
- [Functional Description—Arria 10 EMIF](#) on page 2-1

## Avalon-MM and Memory Data Width

The following table lists the data width ratio between the memory interface and the Avalon-MM interface.

QDR-IV memory devices use a burst length of 2. Because the Avalon-MM interface must supply all of the data for the entire memory burst in a single controller clock cycle, the Avalon-MM data width of the controller is double the memory data width. For width-expanded configurations, the data width is further multiplied by the expansion factor.

**Table 8-2: Data Width Ratio**

Memory Burst Length	Controller
2	2:1

## Signal Descriptions

The following tables lists the signals of the controller's Avalon-MM slave interface.

**Table 8-3: Avalon-MM Slave Interface Signals**

Signal Name	Width	Direction	Avalon-MM Signal Type
amm_ready_i	1	Out	waitrequest_n
amm_read_i	1	In	read
amm_write_i	1	In	write
amm_address_i	21-25	In	address
amm_readdatavalid_i	1	Out	readdatavalid
amm_readdata_i	18, 36, 72	Out	readdata
amm_writedata_i	18, 36, 72	In	writedata
amm_burstcount_i	$\text{ceil}(\log_2(\text{CTRL\_QDR4\_AVL\_MAX\_BURST\_COUNT}+1))$	In	burstcount
emif_usr_clk	1	Out	clk

Signal Name	Width	Direction	Avalon-MM Signal Type
emif_usr_reset_n	1	Out	reset_n
global_reset_n	1	In	reset_n

- Note:** 1. In the above table, the \_i suffix represents the Avalon-MM interface index, which has the range of 0-7.
2. To obtain the actual signal width for the data ports (readdata, writedata), you must multiply the widths in the above table by the data width ratio and the width expansion ratio.
  3. • emif\_usr\_clk: User clock domain.
    - emif\_usr\_reset\_n: Reset for the clock domain. Asynchronous assertion and synchronous deassertion (to the emif\_usr\_clk clock signal).
    - global\_reset\_n: Asynchronous reset causes the memory interface to be reset and recalibrated.

#### Related Information

[Avalon Interface Specifications](#)

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Initial Release.

# Functional Description—RLDRAM II Controller

2016.05.02

[EMI\\_RM](#)



[Subscribe](#)



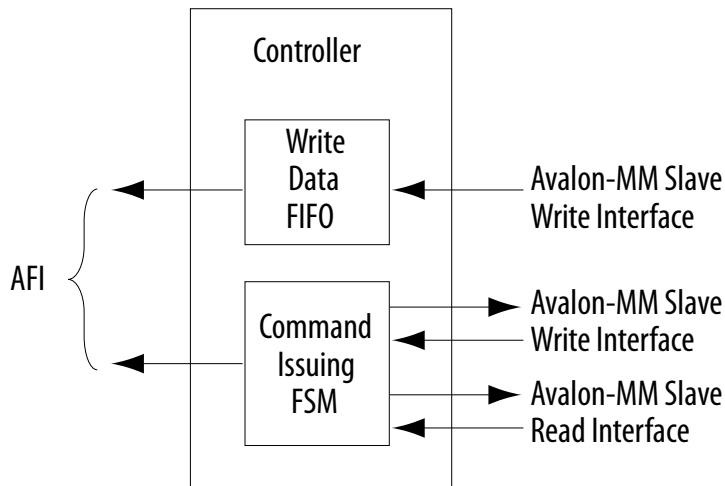
[Send Feedback](#)

The RLDARAM II controller translates memory requests from the Avalon Memory-Mapped (Avalon-MM) interface to AFI, while satisfying timing requirements imposed by the memory configurations.

## Block Description

The following figure shows a block diagram of the RLDARAM II controller architecture.

**Figure 9-1: RLDARAM II Controller Architecture Block Diagram**



## Avalon-MM Slave Interface

The Avalon-MM slave interface accepts read and write requests. A simple state machine represents the state of the command and address registers, which stores the command and address when a request arrives.

The Avalon-MM slave interface decomposes the Avalon-MM address to the memory bank, column, and row addresses. The IP automatically maps the bank address to the LSB of the Avalon address vector.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

The Avalon-MM slave interface includes a burst adaptor, which has two parts:

- The first part is a read and write request combiner that groups requests to sequential addresses into the native memory burst. Given that the second request arrives within the read and write latency window of the first request, the controller can combine and satisfy both requests with a single memory transaction.
- The second part is the burst divider in the front end of the Avalon-MM interface, which breaks long Avalon bursts into individual requests of sequential addresses, which then pass to the controller state machine.

## Write Data FIFO Buffer

The write data FIFO buffer accepts write data from the Avalon-MM interface. The AFI controls the subsequent consumption of the FIFO buffer write data.

## Command Issuing FSM

The command issuing finite-state machine (FSM) has three states.

The controller is in the `INIT` state when the PHY initializes the memory. Upon receiving the `afi_cal_success` signal, the state transitions to `INIT_COMPLETE`. If the calibration fails, `afi_cal_fail` is asserted and the state transitions to `INIT_FAIL`. The PHY receives commands only in the `INIT_COMPLETE` state.

When a refresh request arrives at the state machine at the same time as a read or write request, the refresh request takes precedence. The read or write request waits until there are no more refresh requests, and is issued immediately if timing requirements are met.

## Refresh Timer

With automatic refresh, the refresh timer periodically issues refresh requests to the command issuing FSM. The refresh interval can be set at generation.

## Timer Module

The timer module contains one DQ timer and eight bank timers (one per bank). The DQ timer tracks how often read and write requests can be issued, to avoid bus contention. The bank timers track the cycle time ( $t_{RC}$ ).

The 8-bit wide output bus of the bank timer indicates to the command issuing FSM whether each bank can be issued a read, write, or refresh command.

## AFI

The RLDRAM II controller communicates with the PHY using the AFI interface. For information on the AFI, refer to *AFI 4.0 Specification* in *Functional Description - UniPHY*.

### Related Information

- [AFI 4.0 Specification](#) on page 2-52
- [Functional Description—UniPHY](#) on page 1-1

## User-Controlled Features

The **General Settings** tab of the parameter editor contains several features which are disabled by default. You can enable these features as required for your external memory interface.

### Error Detection Parity

The error detection feature asserts an error signal if it detects any corrupted data during the read process.

The error detection parity protection feature creates a simple parity encoder block which processes all read and write data. For every 8 bits of write data, a parity bit is generated and concatenated to the data before it is written to the memory. During the subsequent read operation, the parity bit is checked against the data bits to ensure data integrity.

When you enable the error detection parity protection feature, the local data width is reduced by one. For example, a nine-bit memory interface will present eight bits of data to the controller interface.

You can enable error detection parity protection in the **Controller Settings** section of the **General Settings** tab of the parameter editor.

### User-Controlled Refresh

The user-controlled refresh feature allows you to take control of the refresh process that the controller normally performs automatically. You can control when refresh requests occur, and, if there are multiple memory devices, you control which bank receives the refresh signal.

When you enable this feature, you disable auto-refresh, and assume responsibility for maintaining the necessary average periodic refresh rate. You can enable user-controlled refresh in the **Controller Settings** section of the **General Settings** tab of the parameter editor.

## Avalon-MM and Memory Data Width

The following table lists the data width ratio between the memory interface and the Avalon-MM interface. The half-rate controller does not support burst-of-2 devices because it under-uses the available memory bandwidth.

**Table 9-1: Data Width Ratio**

Memory Burst Length	Half-Rate Designs	Full-Rate Designs	
2-word	No Support	2:1	
4-word	4:1		
8-word			

## Signal Descriptions

The following table lists the signals of the controller's Avalon-MM slave interface.

For information on the AFI signals, refer to *AFI 4.0 Specification* in *Functional Description - UniPHY*.

**Table 9-2: Avalon-MM Slave Signals**

Signal	Width	Direction	Avalon-MM Signal Type	Description
avl_size	1 to 11	In	burstcount	—
avl_ready	1	Out	waitrequest_n	—
avl_read_req	1	In	read	—
avl_write_req	1	In	write	—
avl_addr	25	In	address	—
avl_rdata_valid	1	Out	readdatavalid	—
avl_rdata	18, 36, 72, 144	Out	readdata	—
avl_wdata	18, 36, 72, 144	In	writedata	—

**Note:** If you are using Qsys, the data width of the Avalon-MM interface is restricted to powers of two. Non-power-of-two data widths are available with the IP Catalog.

**Note:** The RLDRAM II controller does not support the byteenable signal. If the RLDRAM II controller is used with the Avalon-MM Efficiency Monitor and Protocol Checker, data corruption can occur if the byteenable signal on the efficiency monitor is used. For example, this can occur if using the JTAG Avalon-MM Master component to drive the efficiency monitor.

#### Related Information

- [AFI 4.0 Specification](#) on page 2-52
- [Functional Description—UniPHY](#) on page 1-1

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Removed occurrence of MegaWizard Plug-In Manager.

Date	Version	Changes
December 2013	2013.12.16	Removed references to SOPC Builder.
November 2012	3.3	Changed chapter number from 6 to 7.
June 2012	3.2	Added Feedback icon.
November 2011	3.1	Harvested Controller chapter from 11.0 <i>RLDRAM II Controller with UniPHY IP User Guide</i> .

# Functional Description—RLDRAM 3 PHY-Only IP 10

2016.05.02

EMI\_RM



Subscribe



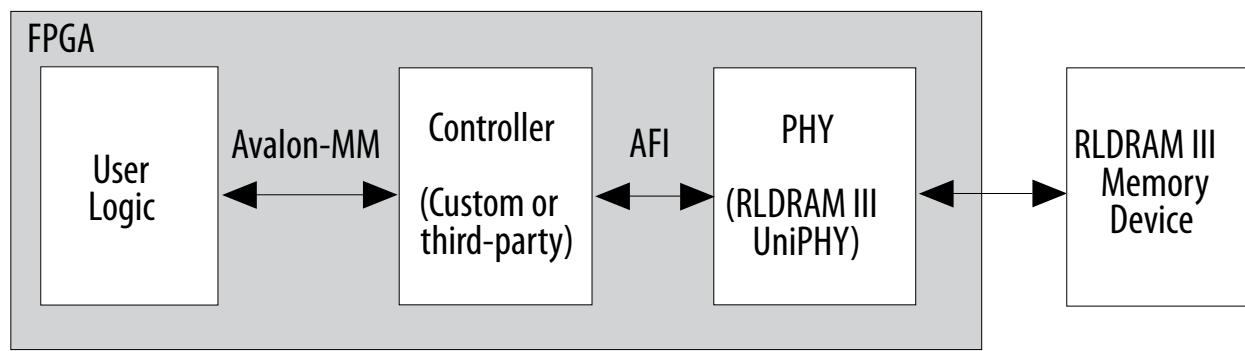
Send Feedback

The RLDARAM 3 PHY-only IP works with a customer-supplied memory controller to translate memory requests from user logic to RLDARAM 3 memory devices, while satisfying timing requirements imposed by the memory configurations.

## Block Description

The RLDARAM 3 UniPHY-based IP is a PHY-only offering which you can use with a third-party controller or a controller that you develop yourself. The following figure shows a block diagram of the RLDARAM 3 system architecture.

Figure 10-1: RLDARAM 3 System Architecture



## Features

The RLDARAM 3 UniPHY-based IP supports features available from major RLDARAM 3 device vendors at speeds of up to 800 MHz.

The following list summarizes key features of the RLDARAM 3 UniPHY-based IP:

- support for Arria V GZ and Stratix V devices
- standard AFI interface between the PHY and the memory controller
- quarter-rate and half-rate AFI interface
- maximum frequency of 533 MHz for half-rate operation and 800 MHz for quarter-rate operation
- burst length of 2, 4, or 8

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

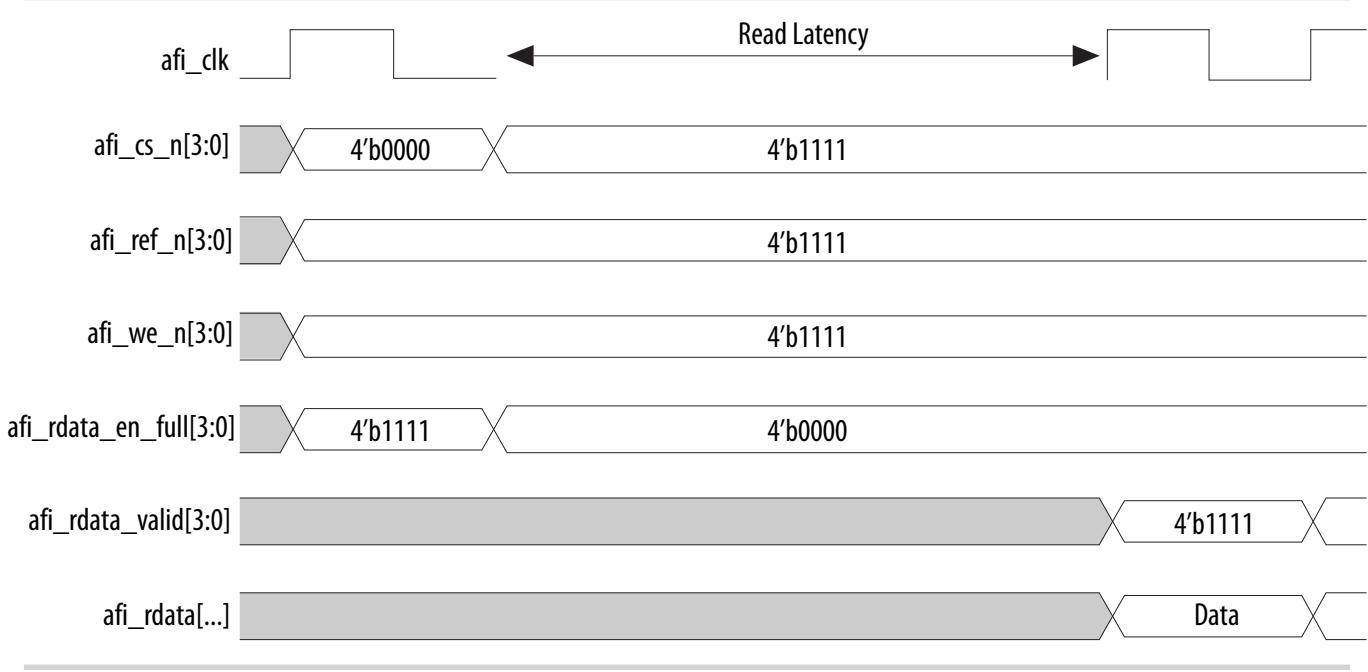
- x18 and x36 memory organization
- common I/O device support
- nonmultiplexed addressing
- multibank write and refresh protocol (programmable through mode register)
- optional use of data mask pins

## RLDRAM 3 AFI Protocol

The RLDRAM 3 UniPHY-based IP communicates with the memory controller using an AFI interface that follows the AFI 4.0 specification. To maximize bus utilization efficiency, the RLDRAM 3 UniPHY-based IP can issue multiple memory read/write operations within a single AFI cycle.

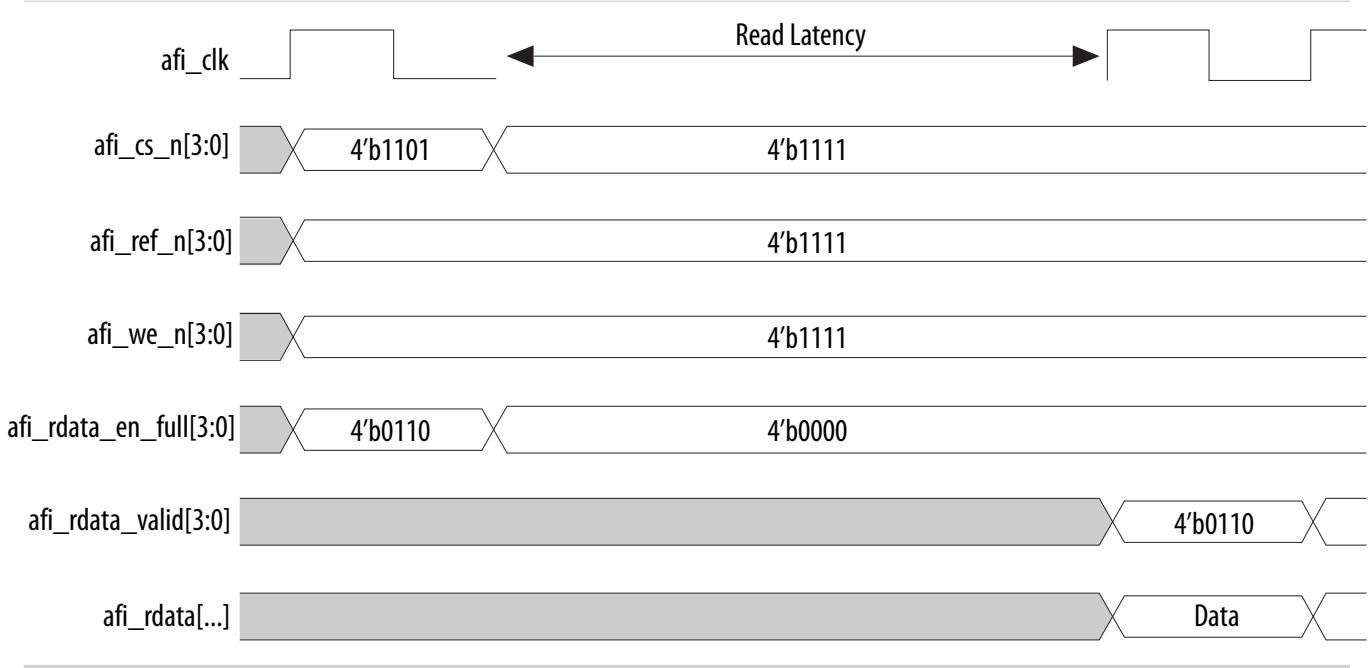
The following figure illustrates AFI bus activity when a quarter-rate controller issues four consecutive burst-length 2 read requests.

**Figure 10-2: AFI Bus Activity for Quarter-Rate Controller Issuing Four Burst-Length 2 Read Requests**



The controller does not have to begin a read or write command using channel 0 of the AFI bus. The flexibility afforded by being able to begin a command on any bus channel can facilitate command scheduling in the memory controller.

The following figure illustrates the AFI bus activity when a quarter-rate controller issues a single burst-length 4 read command to the memory on channel 1 of the AFI bus.

**Figure 10-3: AFI Bus Activity for Quarter-Rate Controller Issuing One Burst-Length 4 Read Request**

**Note:** For information on the AFI, refer to *AFI 4.0 Specification* in *Functional Description - UniPHY*.

#### Related Information

- [AFI 4.0 Specification](#) on page 2-52
- [Functional Description—UniPHY](#) on page 1-1

## RLDRAM 3 Controller with Arria 10 EMIF Interfaces

The following table lists the RLDRAm 3 signals available for each interface when using Arria 10 EMIF IP.

Signal	Interface Type	Description
<b>pll_ref_clk interface</b>		
<code>pll_ref_clock</code>	Clock input	Clock input to the PLL inside EMIF.
<b>afi_clk_interface</b>		
<code>afi_clock</code>	Clock output	AFI clock output from the PLL inside EMIF.
<b>afi_half_clk_interface</b>		
<code>afi_half_clock</code>	Clock output	AFI clock output from the PLL inside EMIF running at half speed.
<b>Memory interface</b>		

Signal	Interface Type	Description
mem_a	Conduit	Interface signal between the PHY and the memory device.
mem_ba		
mem_ck		
mem_ck_n		
mem_cs_n		
mem_dk		
mem_dk_n		
mem_dm		
mem_dq		
mem_qk		
mem_qk_n		
mem_ref_n		
mem_we_n		
mem_reset_n		
<b>Status interface</b>		
local_init_done	Conduit	Memory interface status signal.
local_cal_success		
local_cal_fail		
local_cal_request		
<b>oct interface</b>		
oct_rzqin	Conduit	OCT reference resistor pins for RZQ.
<b>afi_interface</b>		

Signal	Interface Type	Description
afi_addr	Avalon-MM slave	Altera PHY interface (AFI) signal between the PHY and the memory controller.
afi_ba		
afi_cs_n		
afi_we_n		
afi_ref_n		
afi_wdata_valid		
afi_wdata		
afi_dm		
afi_rdata		
afi_rdata_en_full		
afi_rdata_valid		
afi_RST_n		
afi_CAL_success		
afi_CAL_fail		
afi_wlat		
afi_rlat		

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Added <i>RLDRAM 3 Controller with Arria 10 EMIF Interfaces</i> .
December 2013	2013.12.16	Maintenance release.
November 2012	1.0	Initial release.

# Functional Description—Example Designs 11

2016.05.02

EMI\_RM



Subscribe



Send Feedback

Generation of your external memory interface IP creates two independent example designs. These example designs illustrate how to instantiate and connect the memory interface for both synthesis and simulation flows.

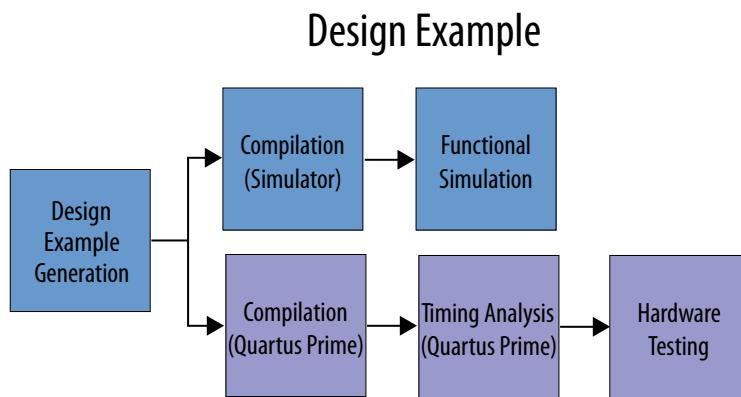
## Arria 10 EMIF IP Example Designs Quick Start Guide

A new interface and more automated design example flow is available for Arria 10 external memory interfaces.

The **Example Designs** tab is available in the parameter editor when you specify an Arria 10 target device. This tab allows you to select from a list of presets for Altera development kits and target interface protocols. All tabs are automatically parameterized with appropriate values, based on the preset that you select. You can specify that the system create directories for simulation and synthesis file sets, and generate the file sets automatically.

You can generate an example design specifically for an Altera development kit, or for any EMIF IP that you generate.

**Figure 11-1: Using the Example Design**

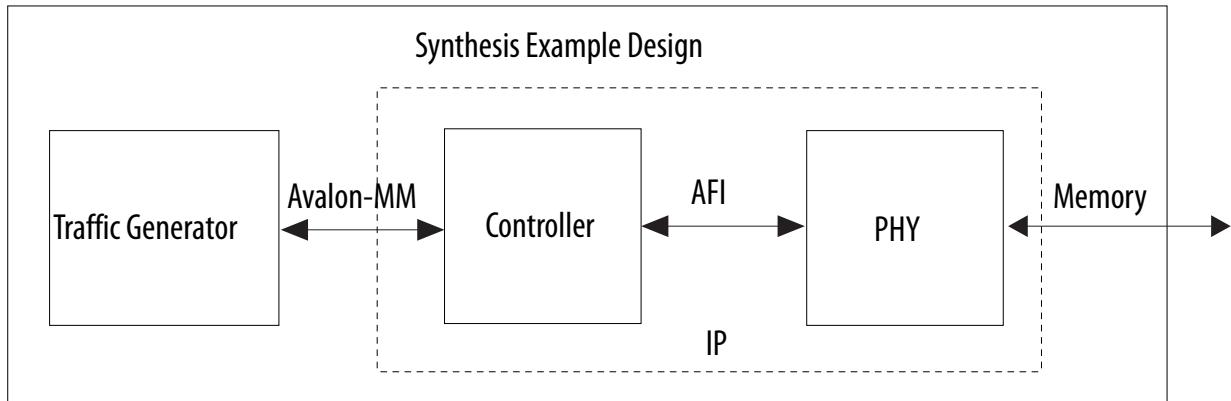
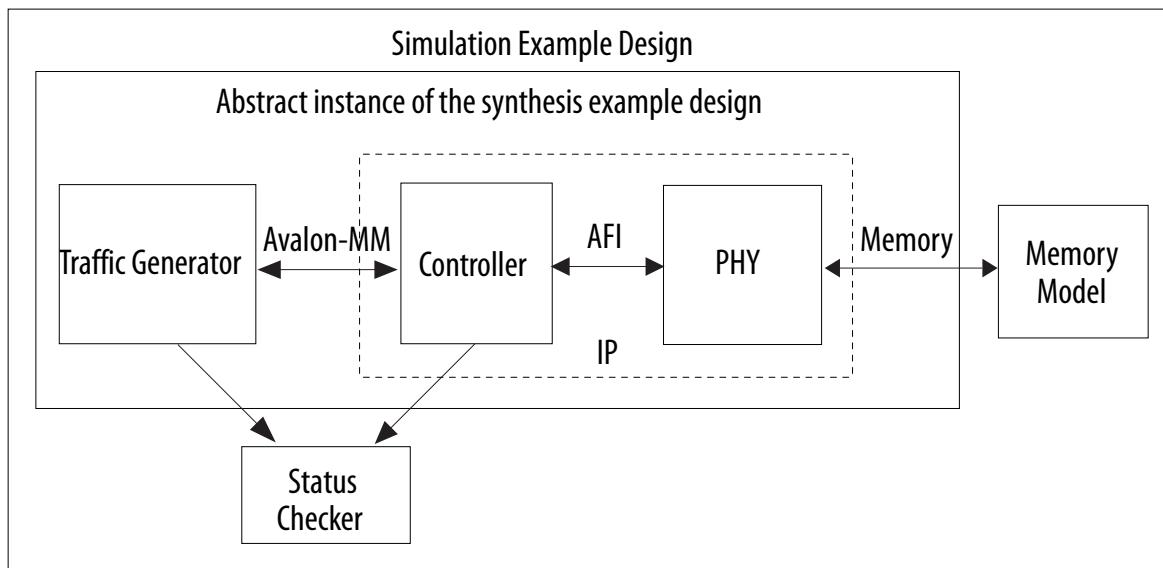


## Typical Example Design Workflow

When you generate your EMIF IP, the system creates an example design that includes a traffic generator and external memory interface controller.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

**Figure 11-2: Example Design for Synthesis****Figure 11-3: Example Design for Simulation**

## Workflow Overview

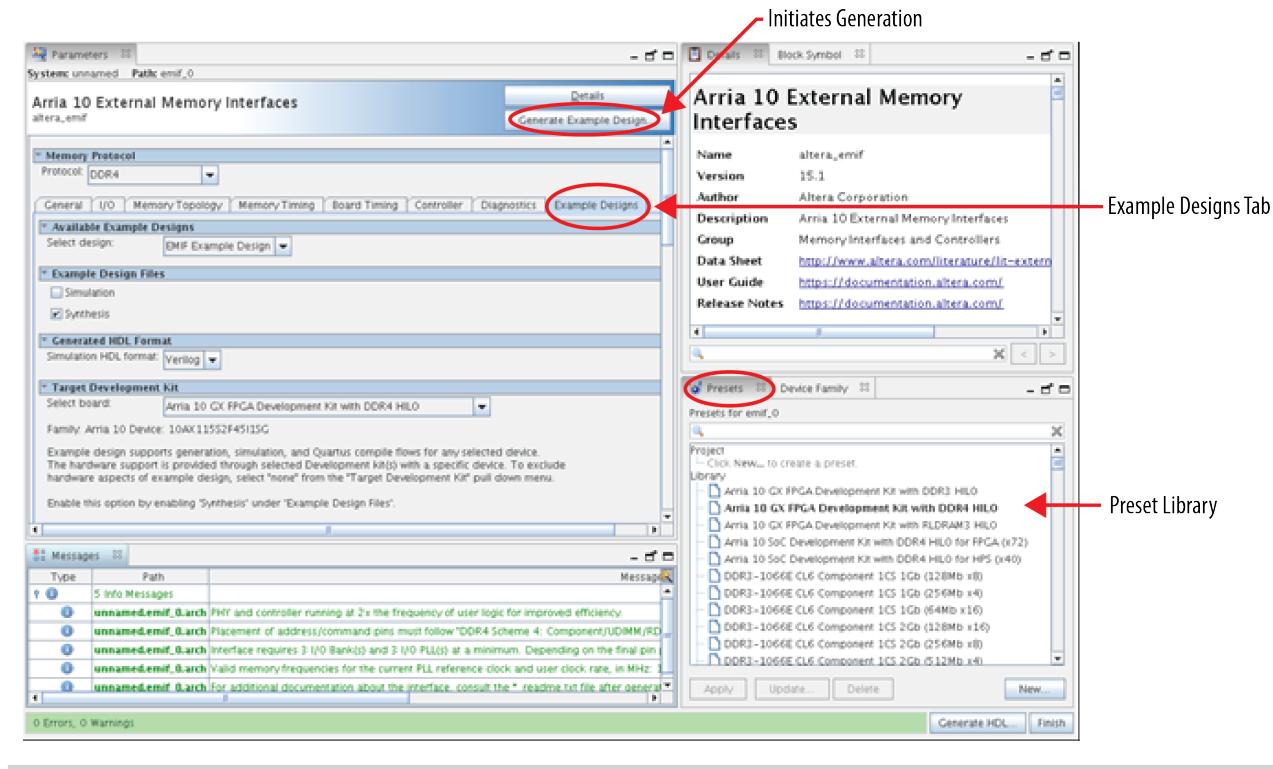
**Figure 11-4: Generating a Simulation or Synthesis File set**

If you select **Simulation** or **Synthesis** under **Example Design Files** on the **Example Designs** tab, the system creates a complete simulation file set or a complete synthesis file set, in accordance with your selection.

## Example Designs Interface Tab

If you have selected an Arria 10 device, the parameter editor includes an **Example Designs** tab which allows you to parameterize and generate your example designs.

**Figure 11-5: Example Designs Tab in the External Memory Interfaces Parameter Editor**



### Available Example Designs Section

The **Select design** pulldown allows you to select the desired example design. At present, **EMIF Example Design** is the only available choice, and is selected by default.

### Example Design Files Section

The example design supports generation, simulation, and Quartus Prime compilation flows for any selected device. The **Simulation** and **Synthesis** checkboxes in this section allow you to specify whether to generate a simulation file set, a synthesis file set, or both. Check **Simulation** to use the example design for simulation, or check **Synthesis** to use the example design for compilation and hardware. The system creates the specified file sets when you click the **Generate Example Design** button.

**Note:** If you don't select the **Simulation** or **Synthesis** checkbox, the destination directory will contain Qsys design files, which are not compilable by Quartus Prime directly, but can be viewed or edited under Qsys.

- To create a compilable project, you must run the **quartus\_sh -t make\_qii\_design.tcl** script in the destination directory.
- To create a simulation project, you must run the **quartus\_sh -t make\_sim\_design.tcl** script in the destination directory.

### Generated HDL Format Section

The **Simulation HDL format** pulldown allows you to specify the target format for the generated design. At present, only Verilog filesets are supported.

### Target Development Kit Section

The **Select board** pulldown in this section applies the appropriate development kit pin assignments to the example design.

- This setting is available only when you turn on the **Synthesis** checkbox in the **Example Design Files** section.
- This setting must match the applied development kit present, or else an error message appears.

If the value *None* appears in the **Select board** pulldown, it indicates that the current parameter selections do not match any development kit configurations. You may apply a development kit-specific IP and related parameter settings by selecting one of the presets from the preset library. When you apply a preset, the current IP and other parameter settings are set to match the selected preset. If you want to save your current settings, you should do so before you select a preset. If you do select a preset without saving your prior settings, you can always save the new preset settings under a different name.

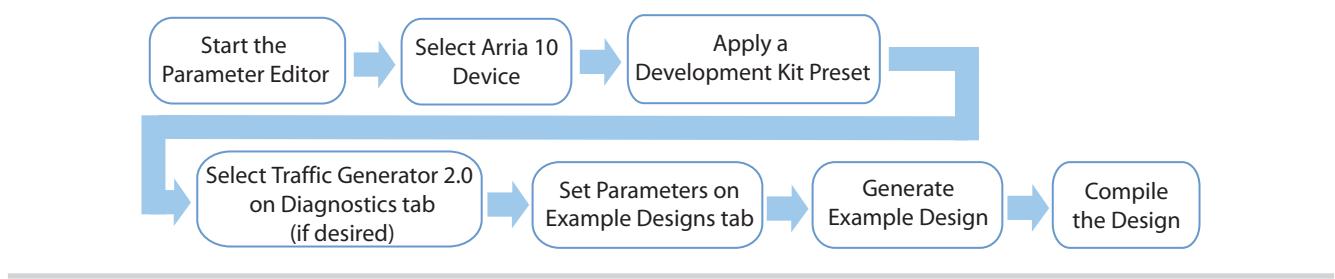
If you want to generate the example design for use on your own board, set **Select board** to *None*, generate the example design, and then add pin location constraints.

**Note:** In the 15.1 release, programming file (**.sof/.pof**) generation is supported for Arria 10 engineering samples only. If you would like to learn more about support for Arria 10 production devices in 15.1, contact your Altera representative or use the support link on [www.altera.com](http://www.altera.com).

## Development Kit Preset Workflow

Arria 10 users can take advantage of development kit presets to create an example design that is automatically configured and ready for hardware testing or simulation. The use of presets greatly simplifies the workflow and reduces the steps necessary to have an EMIF example design working on an Arria 10 development kit.

**Figure 11-6: Generating an Example Design for Use With an Arria 10 Development Kit**



Follow these steps to generate the example design:

1. Under **Memory Interfaces and Controllers** (**Tools > Qsys>IP Catalog>Library>Memory Interfaces and Controllers**), select **Arria 10 External Memory Interfaces**. The parameter editor appears.
2. Specify a top-level name and the folder for your custom IP variation, and specify an Arria 10 device. Click **OK**.
3. Select the appropriate preset for your development kit from the **Presets Library**. Click **Apply**.
4. By default, the example design will be generated with the traditional Traffic Generator. If you would rather use the new EMIF Configurable Traffic Generator 2.0, select **Use configurable Avalon traffic generator 2.0** on the **Diagnostics** tab of the parameter editor.
5. On the **Example Designs** tab in the parameter editor, make the following settings:
  - a. Under **Available Example Designs**, only **EMIF Example Design** is available, and is selected by default.
  - b. Under **Example Design Files**, select **Simulation** or **Synthesis** to have the system create a simulation file set or synthesis file set.
  - c. Under **Generated HDL Format**, only **Verilog** is available, and is selected by default.
  - d. Under **Target Development Kit**, select the desired Arria 10 development kit. The development kit that you select here must match the development kit preset that you selected in step 1, or else an error message appears.

**Note:** The **Target Development Kit** section is available only if you have specified **Synthesis** under **Example Design Files**.

6. Click the **Generate Example Design** button at the top of the parameter editor. The software generates all files necessary to run simulations and hardware tests on the selected development kit.

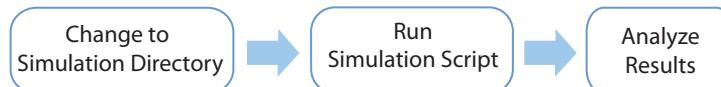
#### Related Information

[Altera Support Center Web Page](#)

## Compiling and Simulating the Design

This section explains how to simulate your EMIF example design.

**Figure 11-7: Procedure for Simulating the Design**



1. Change to the simulation directory.
2. Run the simulation script for the simulator of your choice. Refer to the table below.
3. Analyze the results.

**Table 11-1: Steps to Run Simulation**

Simulator	Working Directory	Instructions
Riviera-PRO	<variation_name>_example_design\sim\aldec	<p>To simulate the example design using the Riviera-PRO simulator, follow these steps:</p> <ol style="list-style-type: none"> <li>At the Linux or Windows shell command prompt, change directory to: &lt;variation_name&gt;_example_design\sim\aldec.</li> <li>Execute the rivierapro_setup.tcl script by typing the following command at the Linux or Windows command prompt: vsim -do rivierapro_setup.tcl .</li> <li>To compile and elaborate the design after the script loads, type: ld_debug.</li> <li>Type run -all to run the simulation.</li> <li>A successful simulation ends with the following message, “--- SIMULATION PASSED ---“</li> </ol>
NCSim	<variation_name>_example_design\sim\cadence\	<p>To simulate the example design using the NCSim simulator, follow these steps:</p> <ol style="list-style-type: none"> <li>At the Linux shell command prompt, change directory to &lt;variation_name&gt;_example_design\sim\cadence\.</li> <li>Run the simulation by typing the following at the command prompt: sh ncsim_setup.sh.</li> <li>A successful simulation ends with the following message, “--- SIMULATION PASSED ---“</li> </ol>
ModelSim	<variation_name>_example_design\sim\mentor\	<p>To simulate the example design using the Modelsim simulator, follow these steps:</p> <ol style="list-style-type: none"> <li>At the Linux or Windows shell command prompt, change directory to &lt;variation_name&gt;_example_design\sim\mentor\.</li> <li>Perform one of the following: <ul style="list-style-type: none"> <li>Execute the msim_setup.tcl script by typing the following command at the Linux or Windows command prompt: vsim -do msim_setup.tcl.</li> <li>Type the following command at the ModelSim command prompt: do msim_setup.tcl.</li> </ul> </li> <li>A successful simulation ends with the following message, “--- SIMULATION PASSED ---“</li> </ol>

Simulator	Working Directory	Instructions
VCS	<variation_name>_example_design\sim\synopsys\vcsmx\	To simulate the example design using the VCS simulator, follow these steps: <ol style="list-style-type: none"><li>1. At the Linux shell command prompt, change directory to &lt;variation_name&gt;_example_design\sim\synopsys\vcsmx\.</li><li>2. Run the simulation by typing the following at the command prompt: sh vcsmx_setup.sh.</li><li>3. A successful simulation ends with the following message, “--- SIMULATION PASSED ---“</li></ol>

## Compiling and Testing the Design in Hardware

This section explains how to compile and test your EMIF example design in hardware.

**Figure 11-8: Procedure for Testing in Hardware**



Follow these steps to compile and test the design in hardware:

1. Before compiling a design, select a method for controlling and monitoring the following pins:
  - **global\_reset\_reset\_n**
  - <variation\_name>**\_status\_local\_cal\_fail**
  - <variation\_name>**\_status\_local\_cal\_success**
  - <variation\_name>**\_tg\_0\_traffic\_gen\_fail**
  - <variation\_name>**\_tg\_0\_traffic\_gen\_pass**
  - <variation\_name>**\_tg\_0\_traffic\_gen\_timeout**
  - Additional pins that may exist if the example design contains more than one traffic generator.
2. Assign pins by one of the following means, as appropriate for your situation:
  - If you have selected a development kit preset, the above pins are automatically assigned to switches and LEDs on the board. See the documentation accompanying your specific development kit for details.
  - If you are using a non-development kit board, assign the above pins to appropriate locations on your board.
  - Alternatively, the above pins can be controlled and monitored using Altera In-System Sources and Probes. To instantiate these in your design, add the following line to your (.qsf) file:  
`set_global_assignment -name VERILOG_MACRO "ALTERA_EMIF_ENABLE_ISSP=1"`
3. Compile the design by clicking **Processing > Start Compilation** in the Quartus Prime software.
4. Configure the FPGA on the board using the generated (.sof) file.
5. Issue a reset on the `global_reset_reset_n` port. If the traffic generator test completes successfully, the <variation\_name>**\_status\_local\_cal\_success** and <variation\_name>**\_tg\_0\_traffic\_gen\_pass** signals will go high, indicating success.
6. If you are using the Traffic Generator 2.0, you can configure and monitor the traffic generator using the EMIF Debug Toolkit. Refer to *Configuring the Traffic Generator 2.0*.

## Configuring the Traffic Generator 2.0

If you have an example design with the Traffic Generator 2.0 enabled, you can configure the traffic pattern using the EMIF Debug Toolkit.

1. Establish a connection to the Traffic Generator by clicking **Create Traffic Generator Connection**.
2. Click **Run Custom Traffic Pattern**.
3. On the **Data** tab, you can set the data to be written on each pin.
  - You can configure each pin individually, or you can use the **All Pins** option to configure all pins simultaneously.
  - All pins can transmit PRBS data, or each pin can transmit its own fixed pattern.
  - The **Seed/Fixed Pattern** has a size in bits equal to the pattern length specified during IP generation (8, 16, or 32). The **Seed/Fixed Pattern** can be specified in decimal or in hexadecimal, using a 0x prefix.
  - If you enable **Test data mask**, the data mask pins are tested by first writing a fixed data pattern to memory, and then running the traffic generator using the specified data mask pattern.
4. On the **Address** tab, you can customize the addresses used by the traffic generator.
  - The **Address Mode** specifies the pattern for generating addresses:
    - In **Sequential** address mode each address is incremented from the previous by the **Sequential address increment**.
    - In **Random** address mode, each address is generated randomly.
    - In **Random Sequential** address mode addresses are randomly generated, then incremented sequentially a number of times equal to the **Number of sequential addresses** setting.
  - **Start address** specifies the starting address used in **Sequential** mode.
  - **Number of sequential addresses** specifies the number of sequential addresses to generate before generating a new random address in **Random sequential** mode.
  - **Sequential address increment** specifies the step size between consecutive addresses in **Sequential** and **Random sequential** modes.
  - **Return to start address** causes the address generator to return to the start address after each loop in **Sequential** address mode.
  - **Mask mode** specifies restrictions on various address components:
    - **Disabled** deactivates masking.
    - **Fixed** restricts the specified address component to the specified value.
    - **Full Cycling** causes the specified address component to be incremented on each new address.
    - **Partial Cycling** causes bank addresses to be cycled for maximum efficiency.
5. On the **Loops** tab, you can set parameters related to the test duration.

- The **Loops** parameter specifies the number of test iterations. Any random address or data generators are not reset between loops.
  - The **Writes/Reads per block** parameter specifies the number of consecutive write/read operations to issue. Typically, the number of writes and reads should be identical in order to verify each write with a corresponding read.
  - The **Write/Read repeats** parameter specifies the number of times to repeat each write or read operation.
  - The **Avalon burst length** parameter specifies the burst length of each Avalon transaction.
6. Click **Ok** to configure the traffic generator and start the traffic pattern.
7. Click **Tasks > Generate All Reports** to view the results of the test.

#### Related Information

- [EMIF Configurable Traffic Generator 2.0 Reference](#)
- [Configurable Traffic Generator 2.0 Parameters](#)
- [Configurable Traffic Generator 2.0 Configuration Options](#)
- [Performing Your Own Tests Using Traffic Generator 2.0](#)

### The Traffic Generator 2.0 Report

The traffic generator report provides information about the configuration of the Traffic Generator 2.0 and the result of the most recent run of traffic.

#### Understanding the Traffic Generator 2.0 Report

The traffic generator report contains the following information:

- A **Pass flag** value of 1 indicates that the run completed with no errors.
- A **Fail flag** value of 1 indicates that the run encountered one or more errors.
- The **Failure Count** indicates the number of read transactions where data did not match the expected value.
- The **First failure** address indicates the address corresponding to the first data mismatch.
- The **Version** indicates the version number of the traffic generator.
- The **Number of data generators** indicates the number of data pins at the memory interface.
- The **Number of byte enable generators** indicates the number of byte enable and data mask pins at the memory interface.
- The **Rank Address**, **Bank address**, and **Bank group width** values indicate the number of bits in the Avalon address corresponding to each of those components.
- The **Data/Byte enable pattern length** indicates the number of bits in the fixed pattern used on each data-byte enable pin.
- The **PNF** (pass not fail) value indicates the persistent pass/fail status for each bit in the Avalon data. It is also presented on a per-memory-pin basis for each beat within a memory burst.
- **Fail Expected Data** is the data that was expected on the first failing transaction (if applicable).
- **Fail Read Data** is the data that was received on the first failing transaction (if applicable).

### EMIF Configurable Traffic Generator 2.0 Reference

The EMIF Configurable Traffic Generator 2.0 can assist in debugging and stress-testing your external memory interface. The traffic generator 2.0 supports Arria 10 and later device families.

## Key Features

The traffic generator 2.0 has the following key features:

- Is a standalone, soft-logic device that resides in the FPGA core.
- Is independent of the FPGA architecture and the external memory protocol in use.
- Offers configuration options for the generation of reads and writes, addressing, data, and data mask.
- Supports many DDR3 and DDR4 features, including various data widths and transfer rates, error-correcting code (ECC), and Ping Pong PHY.

## Configurable Traffic Generator Parameters

Parameters for the EMIF Configurable Traffic Generator 2.0 reside on the **Diagnostics** tab of the parameter editor. The flow for generating your EMIF IP with the configurable traffic generator is the same as when generating the EMIF IP regularly.

**Table 11-2: Configurable Traffic Generator Parameters**

Parameter	Description
Use configurable Avalon traffic generator 2.0	Specifies to use the configurable traffic generator in the example design.
Data Pattern Length	The value you select specifies the length of the data pattern in the data generators, and sets the $n+1$ PRBS coefficient. for example, a value of 8 for PRBS-7)
Byte Enable Pattern Length	The value you select specifies the length of the byte enable pattern in the byte enable generators, and sets the $n+1$ PRBS coefficient. for example, a value of 8 for PRBS-7)
Bypass the default traffic pattern	Specifies not to use the default traffic patterns generated by the traffic generator. The default patterns include single read/write, byte-enabled read/write, and block read/write.
Bypass the user-configured traffic stage	Specifies to skip the stage that uses the user-defined test bench file to configure the traffic generator in simulation.
Bypass the traffic generator repeated-writes/repeated-reads test pattern	Bypasses the traffic generator's repeat test stage, which causes every write and read to be repeated several times.
Bypass the traffic generator stress pattern	Bypasses a test stage intended to stress test signal integrity and memory interface calibration.
Export Traffic Generator 2.0 configuration interface	Instantiates a port for traffic generator configuration. Use this port if the traffic generator is to be configured by user logic,

## Configurable Traffic Generator 2.0 Configuration Options

The EMIF Configurable Traffic Generator 2.0 offers a range of configuration options for fast debugging of your external memory interface. You configure the traffic generator by modifying address-mapped registers through the simulation test bench file or by creating a configuration test stage.

### Configuration Syntax

The test bench includes an example test procedure.

The syntax for writing to a configuration register is as follows:

```
tg_send_cfg_write_<index>(<Register Name>, <Value to be written>);
```

The *index* value represents the index of the memory interface (which is usually 0, but can be 0/1 in ping-pong PHY mode).

The *register name* values are listed in the tables of configuration options, and can also be found in the **altera\_emif\_avl\_tg\_defs.sv** file, in the same directory as the test bench.

The final configuration command must be a write of any value to TG\_START, which starts the traffic generator for the specified interface.

## Configuration Options

Configuration options are divided into read/write, address, data, and data mask generation categories.

**Table 11-3: Configuration Options for Read/Write Generation**

Toolkit GUI Tab	Toolkit GUI Parameter	Register Name	Description
Loops	--	TG_START	Tells the system to perform a write to this register to initiate traffic generation test.
	Loops	TG_LOOP_COUNT	Specifies the number of read/write loops to perform before completion of the test stage. A loop is a single iteration of writes and reads. Upon completion, the base address is either incremented (SEQ or RAND_SEQ) or replaced by a newly generated random address (RAND or RAND_SEQ).
	Writes per block	TG_WRITE_COUNT	Specifies the number of writes to be performed in each loop of the test.
	Reads per block	TG_READ_COUNT	Specifies the number of reads to be performed in each loop of the test. This register must have the same value as TG_WRITE_COUNT.
	--	TG_WRITE_REPEAT_COUNT	Specifies the number of times each write transaction is repeated.
	--	TG_READ_REPEAT_COUNT	Specifies the number of times each read transaction is repeated.
	Reads per block	TG_BURST_LENGTH	Configures the length of each write/read burst to memory.
--	--	TG_CLEAR_FIRST_FAIL	Clears the record of first failure occurrence. New failure information such as expected data, read data, and fail address, is written to the corresponding registers following the next failure.

Toolkit GUI Tab	Toolkit GUI Parameter	Register Name	Description
--	--	TG_TEST_BYTEN	Toggles the byte-enable (data mask enable) register within the traffic generator which allows the current test to use byte-enable signals.
--	--	TG_DATA_MODE	Specifies the source of data used for data signal generation during the test. Set to 0 to use pseudo-random data. Set to 1 to use user-specified values stored in the static data generators.
--	--	TG_BYTEN_MODE	Specifies the source of data used for byte-enable signal generation during the test. Set to 0 to use pseudo-random data. Set to 1 to use user-specified values stored in the static data generators.

**Table 11-4: Configuration Options for Address Generation**

Toolkit GUI Tab	Toolkit GUI Parameter	Register Name	Description
Address	Start Address	TG_SEQ_START_ADDR_WR_L	Specifies the sequential start address (lower 32 bits).
	Start address	TG_SEQ_START_ADDR_WR_H	Specifies the sequential start address (upper 32 bits)
	Address mode	TG_ADDR_MODE_WR	Specifies how write addresses are generated by writing the value in parentheses to the register address. Values are: randomize the address for every write (0), increment sequentially from a specified address (1), increment sequentially from a random address (2), or perform one hot addressing (3).
	Number of sequential address	TG_RAND_SEQ_ADDRS_WR	Specifies the number of times to increment sequentially on the random base address before generating a new random write address.
	Return to start address	TG_RETURN_TO_START_ADDR	Return to start address in deterministic sequential address mode (if 1 is written to TG_ADDR_MODE_WR) after every loop of transactions.
	Rank   Mask Mode	TG_RANK_MASK_EN	Specifies the rank masking mode by writing the value in parentheses to the register address. Values are: disable rank masking (0), maintain a static rank mask (1), cycle through rank masks incrementally (2).

Toolkit GUI Tab	Toolkit GUI Parameter	Register Name	Description
Bank Address   Mask Mode	Bank Address   Mask Mode	TG_BANK_MASK_EN	Specifies the bank masking mode by writing the value in parentheses to the register address. Values are: disable bank masking (0), maintain a static bank mask (1), cycle through bank masks incrementally (2), cycle through only three consecutive bank masks (3).
	Row   Mask Mode	TG_ROW_MASK_EN	Specifies the mode for row masking by writing the value in parentheses to the register address. Values are: disable row masking (0), maintain a static row mask (1), cycle through row masks incrementally (2).
	Bank Group   Mask Mode	TG_BG_MASK_EN	Specifies the mode for bank group masking by writing the value in parentheses to the register address. Values are: disable bank group masking (0), maintain a static bank group mask (1), cycle through bank group masks incrementally (2).
	Rank   Mask Value	TG_RANK_MASK	Specifies the initial rank to be masked into the generated traffic generator address.
	Bank Address   Mask Value	TG_BANK_MASK	Specifies the initial bank to be masked into the generated traffic generator address.
	Row   Mask Value	TG_ROW_MASK	Specifies the initial row to be masked into the generated traffic generator address.
	Bank Group   Mask Value	TG_BG_MASK	Specifies the initial bank group to be masked into the generated traffic generator address.
	Sequential Address Increment	TG_SEQ_ADDR_INCR	Specifies the increment to use when sequentially incrementing the address. This value is used by both deterministic sequential addressing and random sequential addressing. (Refer to TG_ADDR_MODE_WR)
	Start Address	TG_SEQ_START_ADDR_RD_L	Specifies the sequential start read address (lower 32 bits).
		TG_SEQ_START_ADDR_RD_H	Specifies the sequential start read address (upper 32 bits).
Address Mode		TG_ADDR_MODE_RD	Similar to TG_ADDR_MODE_WR but for reads.

Toolkit GUI Tab	Toolkit GUI Parameter	Register Name	Description
	Number of sequential address	TG_RAND_SEQ_ADDRS_RD	Specifies the number of times to increment the random sequential read address.

**Table 11-5: Configuration Options for Data Generation**

Toolkit GUI Tab	Toolkit GUI Parameter	Register Name	Description
Data	Seed/Fixed Pattern	TG_DATA_SEED	Specifies an initial value to the data generator corresponding to the index value.
	<b>PRBS and Fixed Pattern</b> radio buttons	TG_DATA_MODE	Specifies whether to treat the initial value of the data generator of corresponding index as a seed for generating pseudo-random data (value of 0) or to keep the initial value static (value of 1).

**Table 11-6: Configuration Options for Data Mask Generation**

Toolkit GUI Tab	Toolkit GUI Parameter	Register Name	Description
Data	Seed/Fixed Pattern	TG_BYTEEN_SEED	Specifies an initial value to the byte-enable generator corresponding to the index value.
	<b>PRBS and Fixed Pattern</b> radio buttons	TG_BYTEEN_MODE	Specifies whether to treat the initial value of the byte-enable generator of corresponding index as a seed and generate pseudo-random data (value of 0) or to keep the initial value static (value of 1).

## Test Information

In the test bench file, register reads are encoded in a similar syntax to register writes.

The following example illustrates the syntax of a register read:

```
integer <Variable Name>;
tg_send_cfg_read_<index>(<Register Name>, <Variable Name>);
```

In hardware, you can probe the registers storing the test information (such as pnf\_per\_bit\_persist, first\_fail\_read\_address, first\_fail\_read\_data, and first\_fail\_expected\_data).

**Table 11-7: Test Information Read-Accessible Through Register Addresses**

Register Name	Description
TG_PASS	Returns a high value if the traffic generator passes at the end of all the test stages.

Register Name	Description
TG_FAIL	Returns a high value if the traffic generator fails at the end of all the test stages.
TG_FAIL_COUNT_L	Reports the failure count (lower 32 bits).
TG_FAIL_COUNT_H	Reports the failure count(upper 32 bits).
TG_FIRST_FAIL_ADDR_L	Reports the address of the first failure (lower 32 bits).
TG_FIRST_FAIL_ADDR_H	Reports the address of the first failure (upper 32 bits).
TG_FIRST_FAIL_IS_READ	First failure is Read Failure.
TG_FIRST_FAIL_IS_WRITE	First failure is Write Failure.
TG_VERSION	Reports the traffic generator version number.
TG_NUM_DATA_GEN	Reports the number of data generators.
TG_NUM_BYTEEN_GEN	Reports the number of byte-enable generators.
TG_RANK_ADDR_WIDTH	Reports the rank address width.
TG_BANK_ADDR_WIDTH	Reports the bank address width.
TG_ROW_ADDR_WIDTH	Reports the row address width.
TG_BANK_GROUP_WIDTH	Reports the bank group width.
TG_RDATA_WIDTH	Reports the width of all data and PNF signals within the traffic generator.
TG_DATA_PATTERN_LENGTH	Reports the length of the static pattern to be loaded into static per-pin data generators.
TG_BYTEEN_PATTERN_LENGTH	Reports the length of the static pattern to be loaded into static per-pin byte-enable generators.
TG_MIN_ADDR_INCR	Reports the minimum address increment permitted for sequential and random-sequential address generation.
TG_ERROR_REPORT	Reports error bits. Refer to <i>Error Report Register Bits</i> for details.
TG_PNF	Read the persistent PNF per bit as an array of 32-bit entries.
TG_FAIL_EXPECTED_DATA	Reports the first failure expected data. This is read as an array of 32-bit entries.
TG_FAIL_READ_DATA	Reports the first failure read data. This is read as an array of 32-bit entries.

The configuration error report register contains information on common misconfigurations of the traffic generator. The bit corresponding to a given configuration error is set high when that configuration error is detected. Ensure that all bits in this register are low, to avoid test failures or unexpected behavior due to improper configuration.

**Table 11-8: Error Report Register Bits**

Bit Index (LSB = 0x0)	Bit Name	Description of Error
0	ERR_MORE_READS_THAN_WRITES	You have requested more read transactions per loop than write transactions per loop.
1	ERR_BURSTLENGTH_GT_SEQ_ADDR_INCR	The configured burst length is greater than the configured sequential address increment when address generation is in sequential or random sequential mode.
2	ERR_ADDR_DIVISIBLE_BY_GT_SEQ_ADDR_INCR	The configured sequential address increment is less than the minimum required address increment when address generation is in sequential or random sequential mode.
3	ERR_SEQ_ADDR_INCR_NOT_DIVISIBLE	The configured sequential address increment is not a multiple of the minimum required address increment when address generation is in sequential or random sequential mode.
4	ERR_READ_AND_WRITE_START_ADDRS_DIFFER	The configured start addresses for reads and writes are different when address generation is in sequential mode.
5	ERR_ADDR_MODES_DIFFERENT	The configured address modes for reads and writes are different.
6	ERR_NUMBER_OF RAND SEQ_ADDRS_DIFFERENT	The configured number of random sequential addresses for reads and writes are different when address generation is in random sequential mode.
7	ERR_REPEATS_SET_TO_ZERO	The number of read and/or write repeats is set to 0.
8	ERR_BOTH_BURST_AND_REPEAT_MODE_ACTIVE	The burst length is set greater than 1 and read/write requests are set greater than 1.
9-31	Reserved	--

## Performing Your Own Tests Using Traffic Generator 2.0

If you want, you can create your own configuration test stage for the EMIF Configurable Traffic Generator.

The general flow of a configuration test stage, including the default test stages, is as follows:

- Configure the number of loops to be completed by the traffic generator (TG\_LOOP\_COUNT).
- Configure the number of writes and reads to be complete per loop (TG\_WRITE\_COUNT and TG\_READ\_COUNT respectively).
- Choose the burst length of each write and read (TG\_BURST\_LENGTH).
- Select starting write address by writing to the lower and upper bits of the address register (TG\_SEQ\_START\_ADDR\_WR\_L and TG\_SEQ\_START\_ADDR\_WR\_H, respectively).
- Select write address generation mode (TG\_ADDR\_MODE\_WR).

- Select starting read address by writing to the lower and upper bits of the address register (TG\_SEQ\_START\_ADDR\_RD\_L and TG\_SEQ\_START\_ADDR\_RD\_H, respectively).
- Select read address generation mode (TG\_ADDR\_MODE\_RD).
- If applicable, select sequential address increment (TG\_SEQ\_ADDR\_INCR).
- Write initial values/seeds to the data and byte-enable generators (TG\_DATA\_SEED and TG\_BYTEEN\_SEED).
- Select generation mode of the data and byte-enable generators (TG\_DATA\_MODE and TG\_BYTEEN\_MODE).
- Initiate test (TG\_START).

## Simulation

For a comprehensive example of how to write your own configuration test for simulation, refer to the test bench file, located at **<example\_design\_directory>/sim/ed\_sim/altera\_emif\_tg\_avl\_2\_>/sim/altera\_emif\_avl\_tg\_2\_tb.sv**

To iterate over the data generators or byte-enable generators, you must read the number of data generators and number of byte-enable generators. These values are mapped to read-accessible registers TG\_NUM\_DATA\_GEN and TG\_NUM\_BYTEEN\_GEN, respectively. The following example illustrates how one would configure the data generators to continuously output the pattern *0x5a*, using the simulation test bench:

```
integer num_data_generators;  
...  
    tg_send_cfg_read_0(TG_NUM_DATA_GEN, num_data_generators);  
    tg_send_cfg_write_0(TG_DATA_MODE, 32'h1);  
    for (i = 0; i < num_data_generators; i = i + 1) begin  
        tg_send_cfg_write_0(TG_DATA_SEED + i, 32'h5A);  
  
    end
```

## Hardware

Configuration test stages in hardware must be inserted into the RTL, and will resemble the single read/write, byte-enable, and block read/write stages in the default test pattern. In most cases, you can modify one of the existing stages to create the desired custom test stage. The stages are linear, finite, state machines that write predetermined values to the configuration address-mapped registers. As always, the last state in configuration is a write to address *0x0* or *TG\_START*. The state machine then waits for the traffic generator to return a signal signifying its completion of the test stage.

Refer to the aforementioned default test stages as examples of hardware test stages. The default test stages are contained within the following files:

- **<example\_design\_directory>/qii/ed\_synth/altera\_emif\_tg\_avl\_2\_>/synth/altera\_emif\_avl\_tg\_2\_rw\_stage.sv**
- **<example\_design\_directory>/qii/ed\_synth/altera\_emif\_tg\_avl\_2\_>/synth/altera\_emif\_avl\_tg\_2\_bytetable\_test\_stage.sv**

You can also configure the configurable traffic generator in real time using the EMIF Debug Toolkit. The configuration settings available in the Toolkit interface are detailed in the *Configurable Traffic Generator 2.0 Configuration Options* topic.

## UniPHY-Based Example Designs

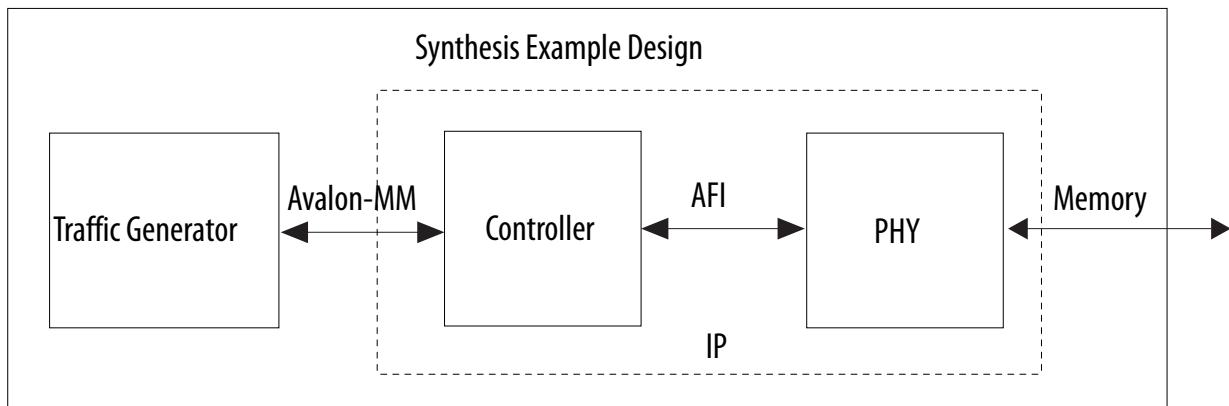
For UniPHY-based interfaces, two independent example designs are created, each containing independent RTL files and other project files. You should compile or simulate these designs separately, and the files should not be mixed. Nonetheless, the designs are related, as the simulation example design builds upon the design of the synthesis example design.

### Synthesis Example Design

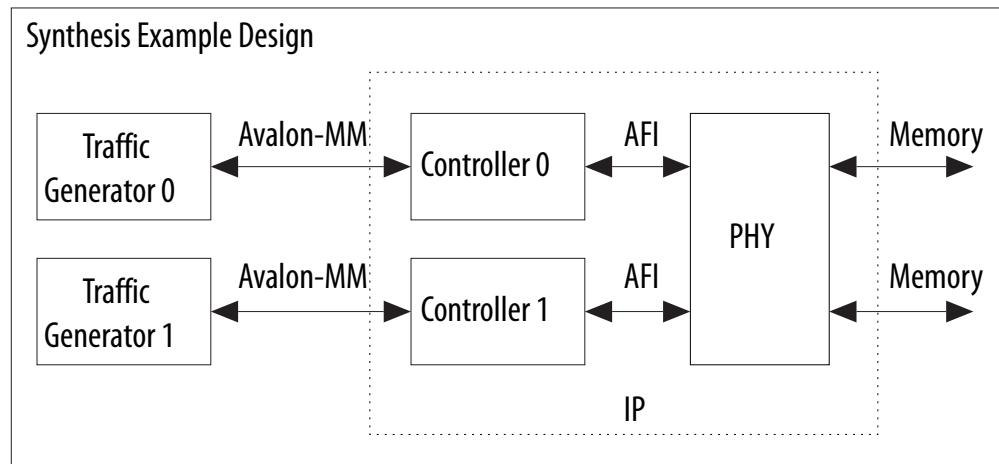
The synthesis example design contains the major blocks shown in the figure below.

- A traffic generator, which is a synthesizable Avalon-MM example driver that implements a pseudo-random pattern of reads and writes to a parameterized number of addresses. The traffic generator also monitors the data read from the memory to ensure it matches the written data and asserts a failure otherwise.
- An instance of the UniPHY memory interface, which includes a memory controller that moderates between the Avalon-MM interface and the AFI interface, and the UniPHY, which serves as an interface between the memory controller and external memory devices to perform read and write operations.

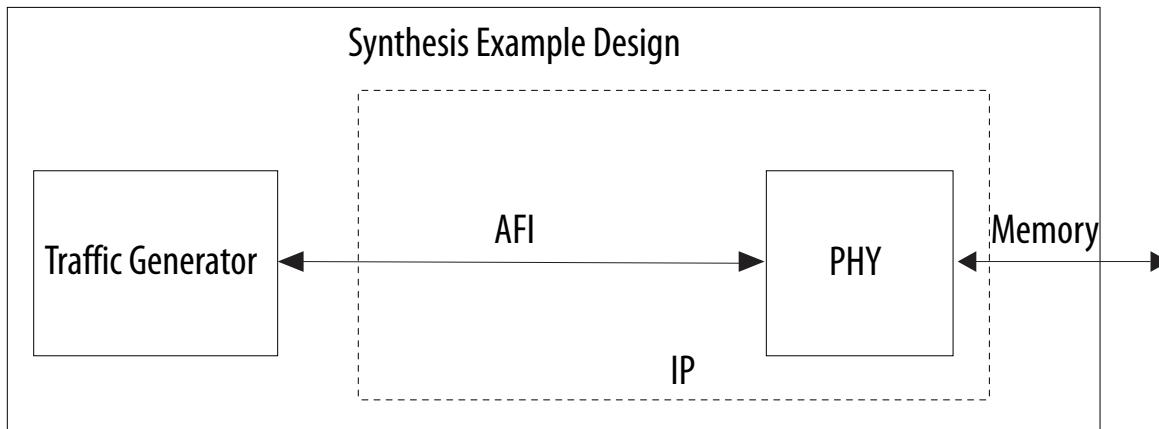
**Figure 11-9: Synthesis Example Design**



If you are using the Ping Pong PHY feature, the synthesis example design includes two traffic generators issuing commands to two independent memory devices through two independent controllers and a common PHY, as shown in the following figure.

**Figure 11-10: Synthesis Example Design for Ping Pong PHY**

If you are using RLDRAM 3, the traffic generator in the synthesis example design communicates directly with the PHY using AFI, as shown in the following figure.

**Figure 11-11: Synthesis Example Design for RLDRAM 3 Interfaces**

You can obtain the synthesis example design by generating your IP core. The files related to the synthesis example design reside at `<variation_name>_example_design / example_project`. The synthesis example design includes a Quartus Prime project file (`<variation_name>_example_design / example_project /<variation_name>_example.qpf`). The Quartus Prime project file can be compiled in the Quartus Prime software, and can be run on hardware.

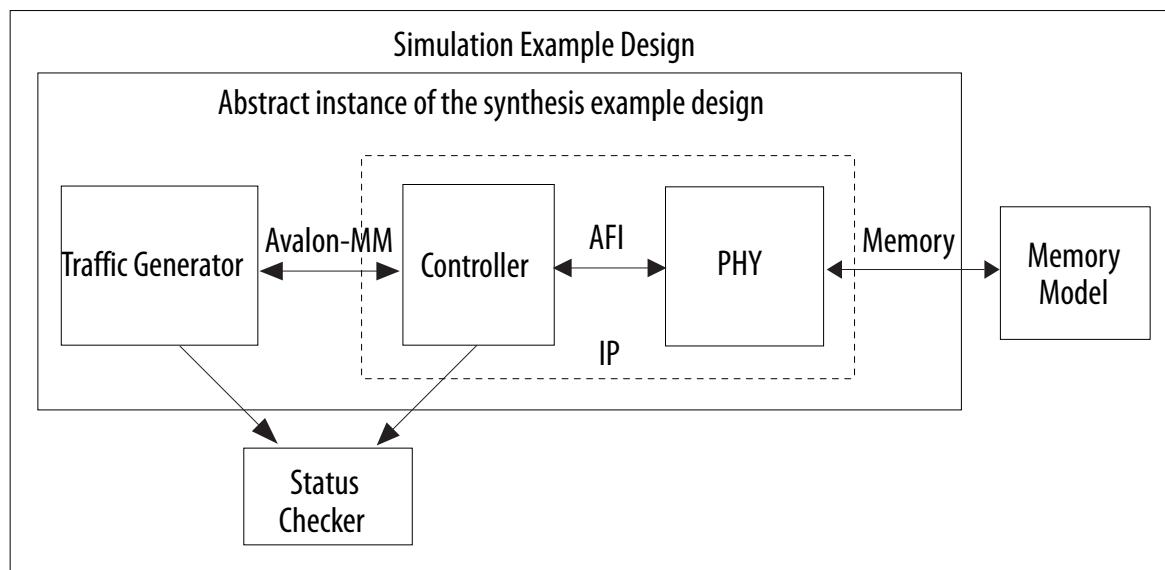
**Note:** If one or more of the **PLL Sharing Mode**, **DLL Sharing Mode**, or **OCT Sharing Mode** parameters are set to any value other than **No Sharing**, the synthesis example design will contain two traffic generator/memory interface instances. The two traffic generator/memory interface instances are related only by shared PLL/DLL/OCT connections as defined by the parameter settings. The traffic generator/memory interface instances demonstrate how you can make such connections in your own designs.

## Simulation Example Design

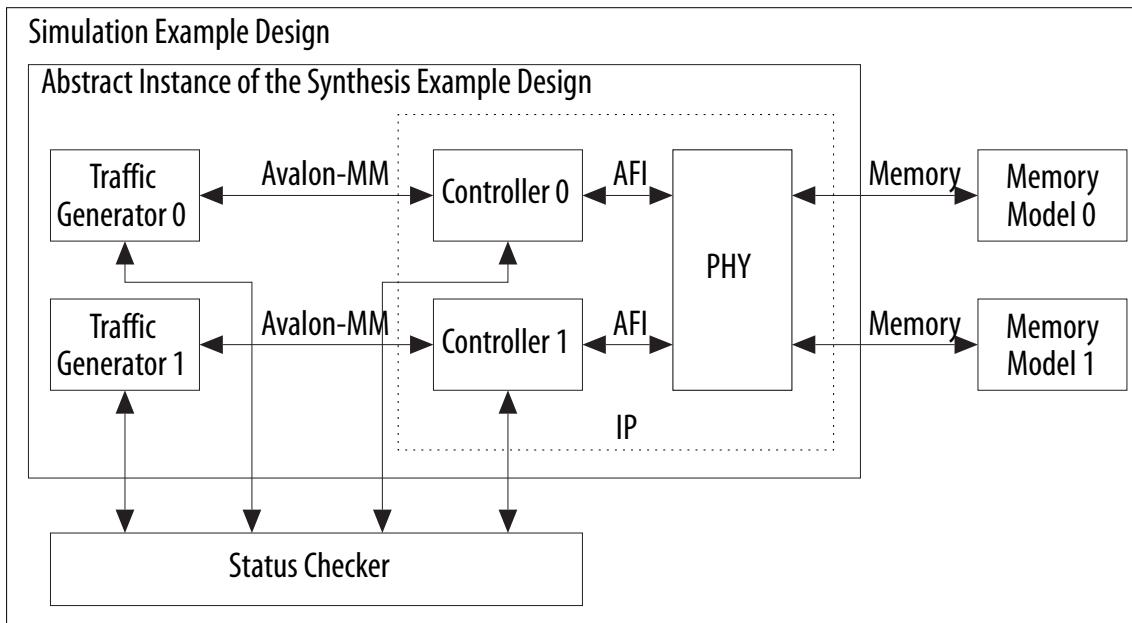
The simulation example design contains the major blocks shown in the following figure.

- An instance of the synthesis example design. As described in the previous section, the synthesis example design contains a traffic generator and an instance of the UniPHY memory interface. These blocks default to abstract simulation models where appropriate for rapid simulation.
- A memory model, which acts as a generic model that adheres to the memory protocol specifications. Frequently, memory vendors provide simulation models for specific memory components that you can download from their websites.
- A status checker, which monitors the status signals from the UniPHY IP and the traffic generator, to signal an overall pass or fail condition.

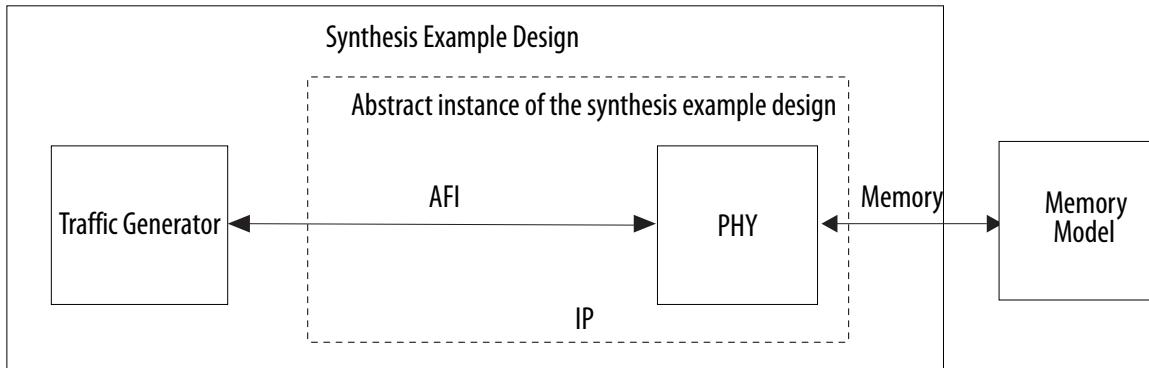
Figure 11-12: Simulation Example Design



If you are using the Ping Pong PHY feature, the simulation example design includes two traffic generators issuing commands to two independent memory devices through two independent controllers and a common PHY, as shown in the following figure.

**Figure 11-13: Simulation Example Design for Ping Pong PHY**

If you are using RLDRAM 3, the traffic generator in the simulation example design communicates directly with the PHY using AFI, as shown in the following figure.

**Figure 11-14: Simulation Example Design for RLDRAM 3 Interfaces**

You can obtain the simulation example design by generating your IP core. The files related to the simulation example design reside at `<variation_name>_example_design/simulation`. After obtaining the generated files, you must still generate the simulation example design RTL for your desired HDL language. The file `<variation_name>_example_design/simulation/README.txt` contains details about how to generate the IP and to run the simulation in ModelSim-AE/SE.

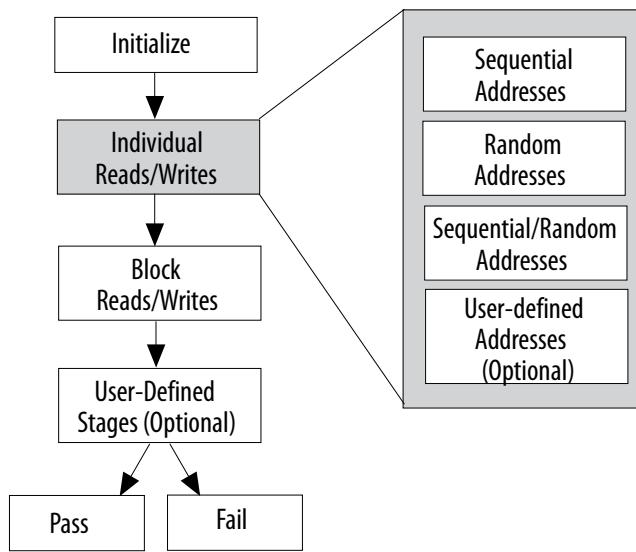
## Traffic Generator and BIST Engine

The traffic generator and built-in self test (BIST) engine for Avalon-MM memory interfaces generates Avalon-MM traffic on an Avalon-MM master interface. The traffic generator creates read and write traffic, stores the expected read responses internally, and compares the expected responses to the read

responses as they arrive. If all reads report their expected response, the pass signal is asserted; however, if any read responds with unexpected data a fail signal occurs.

Each operation generated by the traffic generator is a single write or block of writes followed by a single read or block of reads to the same addresses, which allows the driver to precisely determine the data that should be expected when the read data is returned by the memory interface. The traffic generator comprises a traffic generation block, the Avalon-MM interface and a read comparison block. The traffic generation block generates addresses and write data, which are then sent out over the Avalon-MM interface. The read comparison block compares the read data received from the Avalon-MM interface to the write data from the traffic generator. If at any time the data received is not the expected data, the read comparison block records the failure, finishes reading all the data, and then signals that there is a failure and the traffic generator enters a fail state. If all patterns have been generated and compared successfully, the traffic generator enters a pass state.

**Figure 11-15: Example Driver Operations**



Within the traffic generator, there are the following main states:

- Generation of individual read and writes
- Generation of block read and writes
- The pass state
- The fail state

Within each of the generation states there are the following substates:

- Sequential address generation
- Random address generation
- Mixed sequential and random address generation

For each of the states and substates, the order and number of operations generated for each substate is parameterizable—you can decide how many of each address pattern to generate, or can disable certain patterns entirely if you want. The sequential and random interleave substate takes in additions to the number of operations to generate. An additional parameter specifies the ratio of sequential to random addresses to generate randomly.

## Read and Write Generation

The traffic generator block can generate individual or block reads and writes.

### Individual Read and Write Generation

During the traffic generator's individual read and write generation state, the traffic generation block generates individual write followed by individual read Avalon-MM transactions, where the address for the transactions is chosen according to the specific substate. The width of the Avalon-MM interface is a global parameter for the driver, but each substate can have a parameterizable range of burst lengths for each operation.

### Block Read and Write Generation

During the traffic generator's block read and write generation state, the traffic generator block generates a parameterizable number of write operations followed by the same number of read operations. The specific addresses generated for the blocks are chosen by the specific substates. The burst length of each block operation can be parameterized by a range of acceptable burst lengths.

## Address and Burst Length Generation

The traffic generator block can perform sequential or random addressing.

### Sequential Addressing

The sequential addressing substate defines a traffic pattern where addresses are chosen in sequential order starting from a user definable address. The number of operations in this substate is parameterizable.

### Random Addressing

The random addressing substate defines a traffic pattern where addresses are chosen randomly over a parameterizable range. The number of operations in this substate is parameterizable.

### Sequential and Random Interleaved Addressing

The sequential and random interleaved addressing substate defines a traffic pattern where addresses are chosen to be either sequential or random based on a parameterizable ratio. The acceptable address range is parameterizable as is the number of operations to perform in this substate.

## Traffic Generator Signals

The following table lists the signals used by the traffic generator.

**Table 11-9: Traffic Generator Signals**

Signal	Signal Type
clk	Input clock to traffic generator. For 28nm devices, use the AFI clock. For 20nm devices, use the emif_usr_clk.
reset_n	Active low reset input. For 28nm devices, typically connected to afi_reset_n. For 20nm devices, typically connected to emif_usr_reset_n.
avl_ready	Refer to Avalon Interface Specification.
avl_write_req	Refer to Avalon Interface Specification.

Signal	Signal Type
avl_read_req	Refer to Avalon Interface Specification.
avl_addr	Refer to Avalon Interface Specification.
avl_size	Refer to Avalon Interface Specification.
avl_wdata	Refer to Avalon Interface Specification.
avl_rdata	Refer to Avalon Interface Specification.
avl_rdata_valid	Refer to Avalon Interface Specification.
pnf_per_bit	Output. Bitwise pass/fail for last traffic generator read compare of AFI interface. No errors produces value of all 1s.
pnf_per_bit_persist	Output. Cumulative bitwise pass/fail for all previous traffic generator read compares of AFI interface. No errors produces value of all 1s.
pass	Active high output when traffic generator tests complete successfully.
fail	Active high output when traffic generator test does not complete successfully.
test_complete	Active high output when traffic generator test completes.

For information about the Avalon signals and the Avalon interface, refer to *Avalon Interface Specifications*.

#### Related Information

##### [Avalon Interface Specifications](#)

## Traffic Generator Add-Ons

Some optional components that can be useful for verifying aspects of the controller and PHY operation are generated in conjunction with certain user-specified options. These add-on components are self-contained, and are not part of the controller or PHY, nor the traffic generator.

## User Refresh Generator

The user refresh generator sends refresh requests to the memory controller when user refresh is enabled. The memory controller returns an acknowledgement signal and then issues the refresh command to the memory device.

The user refresh generator is created when you turn on **Enable User Refresh** on the **Controller Settings** tab of the parameter editor.

## Traffic Generator Timeout Counter

The traffic generator timeout counter uses the Avalon interface clock.

When a test fails due to driver failure or timeout, the `fail` signal is asserted. When a test has failed, the traffic generator must be reset with the `reset_n` signal.

## Creating and Connecting the UniPHY Memory Interface and the Traffic Generator in Qsys

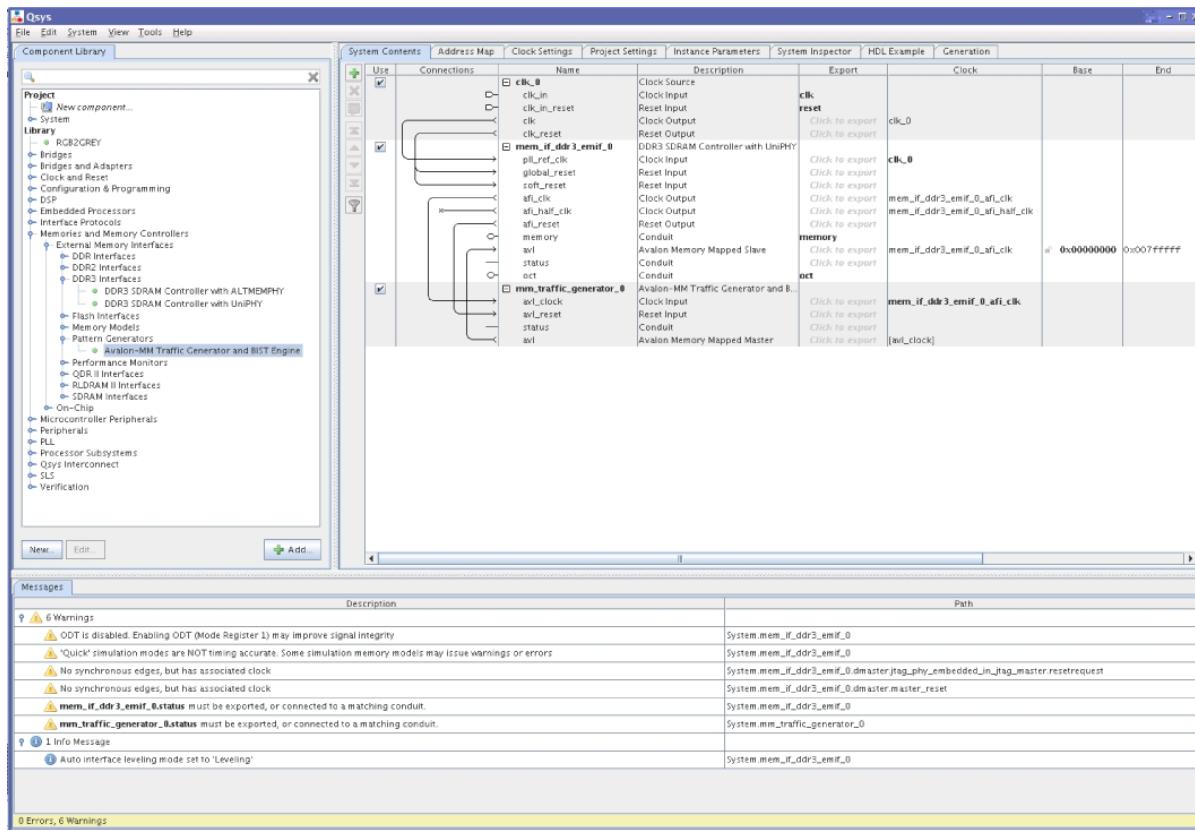
The traffic generator can be used in Qsys as a stand-alone component for use within a larger system.

To create the system in Qsys, perform the following steps:

1. Start Qsys.
2. On the **Project Settings** tab, select the required device from the **Device Family** list.
3. In the **Component Library**, choose a UniPHY memory interface to instantiate. For example, under **Library > Memories and Memory Controllers > External Memory Interfaces**, select **DDR3 SDRAM Controller with UniPHY**.
4. Configure the parameters for your instantiation of the memory interface.
5. In the Component Library, find the example driver and instantiate it in the system. For example, under **Library > Memories and Memory Controllers > Pattern Generators**, select **Avalon-MM Traffic Generator and BIST Engine**.
6. Configure the parameters for your instantiation of the example driver.

**Note:** The Avalon specification stipulates that Avalon-MM master interfaces issue byte addresses, while Avalon-MM slave interfaces accept word addresses. The default for the Avalon-MM Traffic Generator and BIST Engine is to issue word addresses. When using Qsys, you must enable the **Generate per byte address** setting in the traffic generator.

7. Connect the interfaces as illustrated in the following figure. At this point, you can generate synthesis RTL, Verilog or VHDL simulation RTL, or a simulation testbench system.



## Notes on Configuring UniPHY IP in Qsys

This section includes notes and tips on configuring the UniPHY IP in Qsys.

- The address ranges shown for the Avalon-MM slave interface on the UniPHY component should be interpreted as byte addresses that an Avalon-MM master would address, despite the fact that this range is modified by configuring the word addresses width of the Avalon-MM slave interface on the UniPHY controller.
- The `afi_clock` clock source is the associated clock to the Avalon-MM slave interface on the memory controller. This is the ideal clock source to use for all IP components connected on the same Avalon network. Using another clock would cause Qsys to automatically instantiate clock-crossing logic, potentially degrading performance.
- The `afi_clock` clock rate is determined by the **Rate on Avalon-MM interface** setting on the UniPHY **PHY Settings** tab. The `afi_half_clock` clock interface has a rate which is further halved. For example, if **Rate on Avalon-MM interface** is set to **Half**, the `afi_clock` rate is half of the memory clock frequency, and the `afi_half_clock` is one quarter of the memory clock frequency.
- The `global_reset` input interface can be used to reset the UniPHY memory interface and the PLL contained therein. The `soft_reset` input interface can be used to reset the UniPHY memory interface but allow the PLL to remain locked. You can use the `soft_reset` input to reset the memory but to maintain the AFI clock output to other components in the system.

- Do not connect a reset request from a system component (such as a Nios II processor) to the UniPHY `global_reset_n` port. Doing so would reset the UniPHY PLL, which would propagate as a reset condition on `afi_reset` back to the requester; the resulting reset loop could freeze the system.
- Qsys generates an interconnect fabric for each Avalon network. The interconnect fabric is capable of burst and width adaptation. If your UniPHY memory controller is configured with an Avalon interface data width which is wider than an Avalon-MM master interface connected to it, you must enable the byte enable signal on the Avalon-MM slave interface, by checking the **Enable Avalon-MM byte-enable signal** checkbox on the **Controller Settings** tab in the parameter editor.
- If you have a point-to-point connection from an Avalon-MM master to the Avalon-MM slave interface on the memory controller, and if the Avalon data width and burst length settings match, then the Avalon interface data widths may be multiples of either a power of two or nine. Otherwise, you must enable **Generate power-of-2 data bus widths for Qsys or SOPC Builder** on the **Controller Settings** tab of the parameter editor.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"><li>• Modified step 6 of <i>Compiling and Testing the Design in Hardware</i>.</li><li>• Added <i>Configuring the Traffic Generator 2.0</i>.</li><li>• Added <i>The Traffic Generator 2.0 Report</i>.</li><li>• Changed heading from <i>EMIF Configurable Traffic Generator 2.0 for Arria 10 EMIF IP</i> to <i>EMIF Configurable Traffic Generator 2.0 Reference</i>.</li><li>• Added <i>Bypass the traffic generator repeated-writes/ repeated-reads test pattern</i>, <i>Bypass the traffic generator stress pattern</i>, and <i>Export Traffic Generator 2.0 configuration interface</i> to <i>Configurable Traffic Generator Parameters</i> table. Removed <i>Number of Traffic Generator 2.0 configuration interfaces</i>.</li><li>• Added <code>TG_WRITE_REPEAT_COUNT</code>, <code>TG_READ_REPEAT_COUNT</code>, <code>TG_DATA_MODE</code>, and <code>TG_BYTEN_MODE</code> to <i>Configuration Options for Read/Write Generation</i> table.</li><li>• Added <i>Error Report Register Bits</i> table to <i>Test Information</i> section.</li><li>• Corrected paths in <i>Simulation</i> and <i>Hardware</i> sections of <i>Performing Your Own Tests Using Traffic Generator 2.0</i> topic.</li></ul>
November 2015	2015.11.02	<ul style="list-style-type: none"><li>• Added <i>EMIF Configurable Traffic Generator 2.0 for Arria 10 EMIF IP</i>.</li><li>• Added <i>Arria 10 EMIF IP Example Design Quick Start Guide</i>.</li><li>• Changed order of sections in chapter.</li><li>• Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li></ul>
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Removed occurrence of MegaWizard Plug-In Manager.

Date	Version	Changes
December 2013	2013.12.16	Removed references to SOPC Builder.
November 2012	1.3	<ul style="list-style-type: none"><li>Added block diagrams of simulation and synthesis example designs for RLDRAM 3 and Ping Pong PHY.</li><li>Changed chapter number from 7 to 9.</li></ul>
June 2012	1.2	Added Feedback icon.
November 2011	1.1	<ul style="list-style-type: none"><li>Added <i>Synthesis Example Design</i> and <i>Simulation Example Design</i> sections.</li><li>Added <i>Creating and Connecting the UniPHY Memory Interface and the Traffic Generator in Qsys</i>.</li><li>Revised Example Driver section as <i>Traffic Generator and BIST Engine</i>.</li></ul>

# Introduction to UniPHY IP

# 12

2016.05.02

EMI\_RM



Subscribe



Send Feedback

The UniPHY IP is an interface between a memory controller and memory devices and performs read and write operations to the memory. The UniPHY IP creates the datapath between the memory device and the memory controller and user logic in various Altera devices.

The Altera® DDR2, DDR3, and LPDDR2 SDRAM controllers with UniPHY, QDR II and QDR II+ SRAM controllers with UniPHY, RLDRAM II controller with UniPHY, and RLDRAM 3 PHY-only IP provide low latency, high-performance, feature-rich interfaces to industry-standard memory devices. The DDR2, QDR II and QDR II+, and RLDRAM II controllers with UniPHY offer full-rate and half-rate interfaces, while the DDR3 controller with UniPHY and the RLDRAM 3 PHY-only IP offer half-rate and quarter-rate interfaces, and the LPDDR2 controller with UniPHY offers a half-rate interface.

When you generate your external memory interface IP core, the system creates an example top-level project, consisting of an example driver, and your controller custom variation. The controller instantiates an instance of the UniPHY datapath.

The example top-level project is a fully-functional design that you can simulate, synthesize, and use in hardware. The example driver is a self-test module that issues read and write commands to the controller and checks the read data to produce the pass, fail, and test-complete signals.

If the UniPHY datapath does not match your requirements, you can create your own memory interface datapath using the ALTDLL, ALTDQ\_DQS, ALTDQ\_DQS2, ALTDQ, or ALTDQS IP cores, available in the Quartus® II software, but you are then responsible for all aspects of the design including timing analysis and design constraints.

## Release Information

The following table provides information about this release of the DDR2 and DDR3 SDRAM, QDR II and QDR II+ SRAM, and RLDRAM II controllers with UniPHY, and the RLDRAM 3 PHY-only IP.

Table 12-1: Release Information

Item	Protocol			
	DDR2, DDR3, LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Version	13.1	13.1	13.1	13.1

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

Item	Protocol			
	DDR2, DDR3, LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Release Date	November 2013	November 2013	November 2013	November 2013
Ordering Code	IP-DDR2/UNI IP-DDR3/UNI IP-SDRAM/ LPDDR2	IP-QDRII/UNI	IP-RLDII/UNI	—

Altera verifies that the current version of the Quartus Prime software compiles the previous version of each MegaCore function. The *MegaCore IP Library Release Notes and Errata* report any exceptions to this verification. Altera does not verify compilation with MegaCore function versions older than one release.

#### Related Information

[MegaCore IP Library Release Notes](#)

## Device Support Levels

The following terms define the device support levels for Altera IP cores.

#### Altera IP Core Device Support Levels

- **Preliminary support**—Altera verifies the IP core with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. You can use it in production designs with caution.
- **Final support**—Altera verifies the IP core with final timing models for this device family. The IP core meets all functional and timing requirements for the device family and can be used in production designs.

## Device Family and Protocol Support

The following table shows the level of support offered by each of the UniPHY-based external memory interface protocols for Altera device families.

Table 12-2: Device Family Support

Device Family	Support Level					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Arria® II GX	No support	No support	No support	Final	No support	No support
Arria II GZ	Final	Final	No support	Final	Final	No support

Device Family	Support Level					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Arria V	Refer to the <i>What's New in Altera IP</i> page of the Altera website.				No support	
Arria V GZ	Refer to the <i>What's New in Altera IP</i> page of the Altera website.		No support	Refer to the <i>What's New in Altera IP</i> page of the Altera website.		
Cyclone V	Refer to the <i>What's New in Altera IP</i> page of the Altera website.				No support	
Stratix® III	Final	Final (Only Vcc = 1.1V supported)	No support	Final	Final (Only Vcc = 1.1V supported)	No support
Stratix IV	Final	Final	No support	Final	Final	No support
Stratix V	Refer to the <i>What's New in Altera IP</i> page of the Altera website.		No support	Refer to the <i>What's New in Altera IP</i> page of the Altera website.		
Other device families	No support	No support	No support	No support	No support	No support

For information about features and supported clock rates for external memory interfaces, refer to the *External Memory Specification Estimator*.

#### Related Information

- [What's New in Altera IP](#)
- [External Memory Interface Spec Estimator](#)

## UniPHY-Based External Memory Interface Features

The following table summarizes key feature support for Altera's UniPHY-based external memory interfaces.

**Table 12-3: Feature Support**

Key Feature	Protocol					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
High-performance controller II (HPC II)	Yes	Yes	Yes	—	—	—
Half-rate core logic and user interface	Yes	Yes	Yes	Yes	Yes	Yes

Key Feature	Protocol					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Full-rate core logic and user interface	Yes	—	—	Yes	Yes	—
Quarter-rate core logic and user interface	—	Yes <sup>(1)</sup>	—	—	—	Yes
Dynamically generated Nios II-based sequencer	Yes	Yes	Yes	Yes	Yes	Yes
Choice of RTL-based or dynamically generated Nios® II-based sequencer	—	—	—	Yes <sup>(2) (3) (12)</sup>	Yes (12)	—
Available Efficiency Monitor and Protocol Checker <sup>(14)</sup>	Yes	Yes	Yes	—	Yes	Yes
DDR3L support	—	Yes <sup>(13)</sup>	—	—	—	—
UDIMM and RDIMM in any form factor	Yes	Yes <sup>(4) (5)</sup>	—	—	—	—
Multiple components in a single-rank UDIMM or RDIMM layout	Yes	Yes	—	—	—	—
LRDIMM	—	Yes	—	—	—	—
Burst length (half-rate)	8	—	8 or 16	4	4 or 8	2, 4, or 8
Burst length (full-rate)	4	—	—	2 or 4	2, 4, or 8	—
Burst length (quarter-rate)	—	8	—	—	—	2, 4, or 8
Burst length of 8 and burst chop of 4 (on the fly)	—	Yes	—	—	—	—
With leveling	240 MHz and above <sup>(10)</sup>	Yes <sup>(9) (10)</sup>	—	—	—	Yes
Without leveling	Below 240 MHz	—	Yes	—	—	—
Maximum data width	144 bits <sup>(6)</sup>	144 bits <sup>(6)</sup>	32 bits	72 bits	72 bits	72 bits

Key Feature	Protocol					
	DDR2	DDR3	LPDDR2	QDR II	RLDRAM II	RLDRAM 3
Reduced controller latency	—	—	—	Yes <sup>(2)(7)</sup>	Yes <sup>(2)(7)</sup>	—
Read latency	—	—	—	1.5 (QDR II) 2 or 2.5 (QDR II+)	—	—
ODT (in memory device)	—	Yes	—	QDR II+ only	Yes	Yes
x36 emulation mode	—	—	—	Yes <sup>(8)(10)</sup>	—	—

Notes:

1. For Arria V, Arria V GZ, and Stratix V devices only.
2. Not available in Arria II GX devices.
3. Nios II-based sequencer not available for full-rate interfaces.
4. For DDR3, the DIMM form is not supported in Arria II GX, Arria II GZ, Arria V, or Cyclone V devices.
5. Arria II GZ uses leveling logic for discrete devices in DDR3 interfaces to achieve high speeds, but that leveling cannot be used to implement the DIMM form in DDR3 interfaces.
6. For any interface with data width above 72 bits, you must use software timing analysis of your complete design to determine the maximum clock rate.
7. The maximum achievable clock rate when reduced controller latency is selected must be attained through Quatrus Prime software timing analysis of your complete design.
8. Emulation mode allows emulation of a larger memory-width interface using multiple smaller memory-width interfaces. For example, an x36 QDR II or QDR II+ interface can be emulated using two x18 interfaces.
9. The leveling delay on the board between first and last DDR3 SDRAM component laid out as a DIMM must be less than 0.69 tCK.
10. Leveling is not available for Arria V or Cyclone V devices.
11. x36 emulation mode is not supported in Arria V, Arria V GZ, Cyclone V, or Stratix V devices.
12. The RTL-based sequencer is not available for QDR II or RLDRAM II interfaces on Arria V devices.
13. For Arria V, Arria V GZ, Cyclone V, and Stratix V.
14. The Efficiency Monitor and Protocol Checker is not available for QDR II and QDR II+ SRAM, or for the MAX 10 device family, or for Arria V or Cyclone V designs using the Hard Memory Controller.

## System Requirements

For system requirements and installation instructions, refer to *Altera Software Installation and Licensing*.

The DDR2, DDR3, and LPDDR2 SDRAM controllers with UniPHY, QDR II and QDR II+ SRAM controllers with UniPHY, RLDRAM II controller with UniPHY, and RLDRAM 3 PHY-only IP are part of the MegaCore IP Library, which Altera distributes with the Quartus Prime software.

### Related Information

[Altera Software Installation and Licensing Manual](#)

## MegaCore Verification

Altera has carried out extensive random, directed tests with functional test coverage using industry-standard models to ensure the functionality of the external memory controllers with UniPHY. Altera's functional verification of the external memory controllers with UniPHY use modified Denali models, with certain assertions disabled.

## Resource Utilization

The following topics provide resource utilization data for the external memory controllers with UniPHY for supported device families.

### DDR2, DDR3, and LPDDR2 Resource Utilization in Arria V Devices

The following table shows typical resource usage of the DDR2, DDR3, and LPDDR2 SDRAM controllers with UniPHY in the current version of Quartus Prime software for Arria V devices.

**Table 12-4: Resource Utilization in Arria V Devices**

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controller
<b>Controller</b>						
DDR2 (Half rate)	8	2286	1404	4	6560	0
	64	2304	1379	17	51360	0
DDR2 (Fullrate)	32	0	0	0	0	1
DDR3 (Half rate)	8	2355	1412	4	6560	0
	64	2372	1440	17	51360	0
DDR3 (Full rate)	32	0	0	0	0	1
LPDDR2 (Half rate)	8	2230	1617	4	6560	0
	32	2239	1600	10	25760	0
<b>PHY</b>						
DDR2 (Half rate)	8	1652	2015	34	141312	0
	64	1819	2089	34	174080	0
DDR2 (Fullrate)	32	1222	1415	34	157696	1

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controller
DDR3 (Half rate)	8	1653	1977	34	141312	0
	64	1822	2090	34	174080	0
DDR3 (Full rate)	32	1220	1428	34	157696	0
LPDDR2 (Half rate)	8	2998	3187	35	150016	0
	32	3289	3306	35	174592	0

**Total**

DDR2 (Half rate)	8	4555	3959	39	148384	0
	64	4991	4002	52	225952	0
DDR2 (Fullrate)	32	1776	1890	35	158208	1
DDR3 (Half rate)	8	4640	3934	39	148384	0
	64	5078	4072	52	225952	0
DDR3 (Full rate)	32	1774	1917	35	158208	1
LPDDR2 (Half rate)	8	5228	4804	39	156576	0
	32	5528	4906	45	200352	0

**DDR2 and DDR3 Resource Utilization in Arria II GZ Devices**

The following table shows typical resource usage of the DDR2 and DDR3 SDRAM controllers with UniPHY in the current version of Quartus Prime software for Arria II GZ devices.

**Table 12-5: Resource Utilization in Arria II GZ Devices**

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
----------	---------------------	---------------------	-----------------	-----------	------------	--------------	---------------

**Controller**

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
DDR2 (Half rate)	8	1,781	1,092	10	2	0	4,352
	16	1,784	1,092	10	4	0	8,704
	64	1,818	1,108	10	15	0	34,560
	72	1,872	1,092	10	17	0	39,168
DDR2 (Full rate)	8	1,851	1,124	10	2	0	2,176
	16	1,847	1,124	10	2	0	4,352
	64	1,848	1,124	10	8	0	17,408
	72	1,852	1,124	10	9	0	19,574
DDR3 (Half rate)	8	1,869	1,115	10	2	0	4,352
	16	1,868	1,115	10	4	0	8,704
	64	1,882	1,131	10	15	0	34,560
	72	1,888	1,115	10	17	0	39,168
<b>PHY</b>							
DDR2 (Half rate)	8	2,560	2,042	183	22	0	157,696
	16	2,730	2,262	183	22	0	157,696
	64	3,606	3,581	183	22	0	157,696
	72	3,743	3,796	183	22	0	157,696
DDR2 (Full rate)	8	2,494	1,934	169	22	0	157,696
	16	2,652	2,149	169	22	0	157,696
	64	3,519	3,428	169	22	0	157,696
	72	3,646	3,642	169	22	0	157,696

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
DDR3 (Half rate)	8	2,555	2,032	187	22	0	157,696
	16	3,731	2,251	187	22	0	157,696
	64	3,607	3,572	187	22	0	157,696
	72	3,749	3,788	187	22	0	157,696
<b>Total</b>							
DDR2 (Half rate)	8	4,341	3,134	193	24	0	4,374
	16	4,514	3,354	193	26	0	166,400
	64	5,424	4,689	193	37	0	192,256
	72	5,615	4,888	193	39	0	196,864
DDR2 (Full rate)	8	4,345	3,058	179	24	0	159,872
	16	4,499	3,273	179	24	0	162,048
	64	5,367	4,552	179	30	0	175,104
	72	5,498	4,766	179	31	0	177,280
DDR3 (Half rate)	8	4,424	3,147	197	24	0	162,048
	16	5,599	3,366	197	26	0	166,400
	64	5,489	4,703	197	37	0	192,256
	72	5,637	4,903	197	39	0	196,864

## DDR2 and DDR3 Resource Utilization in Stratix III Devices

The following table shows typical resource usage of the DDR2 and DDR3 SDRAM controllers with UniPHY in the current version of Quartus Prime software for Stratix III devices.

**Table 12-6: Resource Utilization in Stratix III Devices**

Protocol	Memory Width (Bits)	Combinational ALUTs	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
<b>Controller</b>							
DDR2 (Half rate)	8	1,807	1,058	0	4	0	4,464
	16	1,809	1,058	0	6	0	8,816
	64	1,810	1,272	10	14	0	32,256
	72	1,842	1,090	10	17	0	39,168
DDR2 (Full rate)	8	1,856	1,093	0	4	0	2,288
	16	1,855	1,092	0	4	0	4,464
	64	1,841	1,092	0	10	0	17,520
	72	1,834	1,092	0	11	0	19,696
DDR3 (Half rate)	8	1,861	1,083	0	4	0	4,464
	16	1,863	1,083	0	6	0	8,816
	64	1,878	1,295	10	14	0	32,256
	72	1,895	1,115	10	17	0	39,168
<b>PHY</b>							
DDR2 (Half rate)	8	2,591	2,100	218	6	1	157,696
	16	2,762	2,320	218	6	1	157,696
	64	3,672	3,658	242	6	1	157,696
	72	3,814	3,877	242	6	1	157,696
DDR2 (Full rate)	8	2,510	1,986	200	6	1	157,696
	16	2,666	2,200	200	6	1	157,696
	64	3,571	3,504	224	6	1	157,696
	72	3,731	3,715	224	6	1	157,696

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
DDR3 (Half rate)	8	2,591	2,094	224	6	1	157,696
	16	2,765	2,314	224	6	1	157,696
	64	3,680	3,653	248	6	1	157,696
	72	3,819	3,871	248	6	1	157,696
<b>Total</b>							
DDR2 (Half rate)	8	4,398	3,158	218	10	1	162,160
	16	4,571	3,378	218	12	1	166,512
	64	5,482	4,930	252	20	1	189,952
	72	5,656	4,967	252	23	1	196,864
DDR2 (Full rate)	8	4,366	3,079	200	10	1	159,984
	16	4,521	3,292	200	10	1	162,160
	64	5,412	4,596	224	16	1	175,216
	72	5,565	4,807	224	17	1	177,392
DDR3 (Half rate)	8	4,452	3,177	224	10	1	162,160
	16	4,628	3,397	224	12	1	166,512
	64	5,558	4,948	258	20	1	189,952
	72	5,714	4,986	258	23	1	196,864

## DDR2 and DDR3 Resource Utilization in Stratix IV Devices

The following table shows typical resource usage of the DDR2 and DDR3 SDRAM controllers with UniPHY in the current version of Quartus Prime software for Stratix IV devices.

**Table 12-7: Resource Utilization in Stratix IV Devices**

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
<b>Controller</b>							
DDR2 (Half rate)	8	1,785	1,090	10	2	0	4,352
	16	1,785	1,090	10	4	0	8,704
	64	1,796	1,106	10	15	0	34,560
	72	1,798	1,090	10	17	0	39,168
DDR2 (Full rate)	8	1,843	1,124	10	2	0	2,176
	16	1,845	1,124	10	2	0	4,352
	64	1,832	1,124	10	8	0	17,408
	72	1,834	1,124	10	9	0	19,584
DDR3 (Half rate)	8	1,862	1,115	10	2	0	4,352
	16	1,874	1,115	10	4	0	8,704
	64	1,880	1,131	10	15	0	34,560
	72	1,886	1,115	10	17	0	39,168
<b>PHY</b>							
DDR2 (Half rate)	8	2,558	2,041	183	6	1	157,696
	16	2,728	2,262	183	6	1	157,696
	64	3,606	3,581	183	6	1	157,696
	72	3,748	3,800	183	6	1	157,696
DDR2 (Full rate)	8	2,492	1,934	169	6	1	157,696
	16	2,652	2,148	169	6	1	157,696
	64	3,522	3,428	169	6	1	157,696
	72	3,646	3,641	169	6	1	157,696

Protocol	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Mem ALUTs	M9K Blocks	M144K Blocks	Memory (Bits)
DDR3 (Half rate)	8	2,575	2,031	187	6	1	157,696
	16	2,732	2,251	187	6	1	157,696
	64	3,602	3,568	187	6	1	157,696
	72	3,750	3,791	187	6	1	157,696
<b>Total</b>							
DDR2 (Half rate)	8	4,343	3,131	193	8	1	162,048
	16	4,513	3,352	193	10	1	166,400
	64	5,402	4,687	193	21	1	192,256
	72	5,546	4,890	193	23	1	196,864
DDR2 (Full rate)	8	4,335	3,058	179	8	1	159,872
	16	4,497	3,272	179	8	1	162,048
	64	5,354	4,552	179	14	1	175,104
	72	5,480	4,765	179	15	1	177,280
DDR3 (Half rate)	8	4,437	3,146	197	8	1	162,048
	16	4,606	3,366	197	10	1	166,400
	64	5,482	4,699	197	21	1	192,256
	72	5,636	4,906	197	23	1	196,864

## DDR2 and DDR3 Resource Utilization in Arria V GZ and Stratix V Devices

The following table shows typical resource usage of the DDR2 and DDR3 SDRAM controllers with UniPHY in the current version of Quartus Prime software for Arria V GZ and Stratix V devices.

**Table 12-8: Resource Utilization in Arria V GZ and Stratix V Devices**

Protocol	Memory Width (Bits)	Combinational LCs	Logic Registers	M20K Blocks	Memory (Bits)
----------	---------------------	-------------------	-----------------	-------------	---------------

### Controller

Protocol	Memory Width (Bits)	Combinational LCs	Logic Registers	M20K Blocks	Memory (Bits)
DDR2 (Half rate)	8	1,787	1,064	2	4,352
	16	1,794	1,064	4	8,704
	64	1,830	1,070	14	34,304
	72	1,828	1,076	15	38,400
DDR2 (Full rate)	8	2,099	1,290	2	2,176
	16	2,099	1,290	2	4,352
	64	2,126	1,296	7	16,896
	72	2,117	1,296	8	19,456
DDR3 (Quarter rate)	8	2,101	1,370	4	8,704
	16	2,123	1,440	7	16,896
	64	2,236	1,885	28	69,632
	72	2,102	1,870	30	74,880
DDR3 (Half rate)	8	1,849	1,104	2	4,352
	16	1,851	1,104	4	8,704
	64	1,853	1,112	14	34,304
	72	1,889	1,116	15	38,400
<b>PHY</b>					
DDR2 (Half rate)	8	2,567	1,757	13	157,696
	16	2,688	1,809	13	157,696
	64	3,273	2,115	13	157,696
	72	3,377	2,166	13	157,696

Protocol	Memory Width (Bits)	Combinational LCs	Logic Registers	M20K Blocks	Memory (Bits)
DDR2 (Full rate)	8	2,491	1,695	13	157,696
	16	2,578	1,759	13	157,696
	64	3,062	2,137	13	157,696
	72	3,114	2,200	13	157,696
DDR3 (Quarter rate)	8	2,209	2,918	18	149,504
	16	2,355	3,327	18	157,696
	64	3,358	5,228	18	182,272
	72	4,016	6,318	18	198,656
DDR3 (Half rate)	8	2,573	1,791	13	157,696
	16	2,691	1,843	13	157,696
	64	3,284	2,149	13	157,696
	72	3,378	2,200	13	157,696
<b>Total</b>					
DDR2 (Half rate)	8	4,354	2,821	15	162,048
	16	4,482	2,873	17	166,400
	64	5,103	3,185	27	192,000
	72	5,205	3,242	28	196,096
DDR2 (Full rate)	8	4,590	2,985	15	159,872
	16	4,677	3,049	15	162,048
	64	5,188	3,433	20	174,592
	72	5,231	3,496	21	177,152

Protocol	Memory Width (Bits)	Combinational LCs	Logic Registers	M20K Blocks	Memory (Bits)
DDR3 (Quarter rate)	8	4,897	4,844	23	158,720
	16	5,065	5,318	26	175,104
	64	6,183	7,669	47	252,416
	72	6,705	8,744	49	274,048
DDR3 (Half rate)	8	4,422	2,895	15	162,048
	16	4,542	2,947	17	166,400
	64	5,137	3,261	27	192,000
	72	5,267	3,316	28	196,096

## QDR II and QDR II+ Resource Utilization in Arria V Devices

The following table shows typical resource usage of the QDR II and QDR II+ SRAM controllers with UniPHY in the current version of Quartus Prime software for Arria V devices.

Table 12-9: Resource Utilization in Arria V Devices

PHY Rate	Memory Width (Bits)	Combinational ALUTs	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controller
<b>Controller</b>						
Half	9	98	120	0	0	0
	18	96	156	0	0	0
	36	94	224	0	0	0
<b>PHY</b>						
Half	9	234	257	0	0	0
	18	328	370	0	0	0
	36	522	579	0	0	0
<b>Total</b>						

PHY Rate	Memory Width (Bits)	Combinational ALUTs	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controller
Half	9	416	377	0	0	0
	18	542	526	0	0	0
	36	804	803	0	0	0

## QDR II and QDR II+ Resource Utilization in Arria II GX Devices

The following table shows typical resource usage of the QDR II and QDR II+ SRAM controllers with UniPHY in the current version of Quartus Prime software for Arria II GX devices.

**Table 12-10: Resource Utilization in Arria II GX Devices**

PHY Rate	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Memory (Bits)	M9K Blocks
Half	9	620	701	0	0
	18	921	1122	0	0
	36	1534	1964	0	0
Full	9	584	708	0	0
	18	850	1126	0	0
	36	1387	1962	0	0

## QDR II and QDR II+ Resource Utilization in Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices

The following table shows typical resource usage of the QDR II and QDR II+ SRAM controllers with UniPHY in the current version of Quartus Prime software for Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V devices.

**Table 12-11: Resource Utilization in Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices**

PHY Rate	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Memory (Bits)	M9K Blocks
Half	9	602	641	0	0
	18	883	1002	0	0
	36	1457	1724	0	0

PHY Rate	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Memory (Bits)	M9K Blocks
Full	9	586	708	0	0
	18	851	1126	0	0
	36	1392	1962	0	0

## RLDRAM II Resource Utilization in Arria V Devices

The following table shows typical resource usage of the RLDARAM II controller with UniPHY in the current version of Quartus Prime software for Arria V devices.

**Table 12-12: Resource Utilization in Arria V Devices (Part 1 of 2)**

PHY Rate	Memory Width (Bits)	Combina-tional ALUTs	Logic Registers	M10K Blocks	Memory (Bits)	Hard Memory Controller
<b>Controller</b>						
Half	9	353	303	1	288	0
	18	350	324	2	576	0
	36	350	402	4	1152	0
<b>PHY</b>						
Half	9	295	474	0	0	0
	18	428	719	0	0	0
	36	681	1229	0	0	0
<b>Total</b>						
Half	9	705	777	1	288	0
	18	871	1043	2	576	0
	36	1198	1631	4	1152	0

## RLDRAM II Resource Utilization in Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices

The following table shows typical resource usage of the RLDARAM II controller with UniPHY in the current version of Quartus Prime software for Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V devices.

**Table 12-13: Resource Utilization in Arria II GZ, Arria V GZ, Stratix III, Stratix IV, and Stratix V Devices<sup>(1)</sup>**

PHY Rate	Memory Width (Bits)	Combinational ALUTS	Logic Registers	Memory (Bits)	M9K Blocks
Half	9	829	763	288	1
	18	1145	1147	576	2
	36	1713	1861	1152	4
Full	9	892	839	288	1
	18	1182	1197	576	1
	36	1678	1874	1152	2

Note to Table:

1. Half-rate designs use the same amount of memory as full-rate designs, but the data is organized in a different way (half the width, double the depth) and the design may need more M9K resources.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Removed occurrences of MegaWizard Plug-In Manager.
December 2013	2013.12.16	<ul style="list-style-type: none"> <li>• Removed references to ALTMEMPHY.</li> <li>• Removed references to HardCopy.</li> </ul>
November 2012	2.1	<ul style="list-style-type: none"> <li>• Added RLDRAM 3 support.</li> <li>• Added LRDIMM support.</li> <li>• Added Arria V GZ support.</li> <li>• Changed chapter number from 8 to 10.</li> </ul>
June 2012	2.0	<ul style="list-style-type: none"> <li>• Added LPDDR2 support.</li> <li>• Moved <i>Protocol Support Matrix</i> to Volume 1.</li> <li>• Added Feedback icon.</li> </ul>

Date	Version	Changes
November 2011	1.1	<ul style="list-style-type: none"><li>• Combined Release Information, Device Family Support, Features list, and Unsupported Features list for DDR2, DDR3, QDR II, and RLDRAM II.</li><li>• Added Protocol Support Matrix.</li><li>• Combined Resource Utilization information for DDR2, DDR3, QDR II, and RLDRAM II.</li><li>• Updated data for 11.1.</li></ul>



2016.05.02

EMI\_RM



Subscribe



Send Feedback

Altera defines read and write latencies in terms of memory device clock cycles, which are always full-rate. There are two types of latencies that exists while designing with memory controllers—read and write latencies, which have the following definitions:

- Read latency—the amount of time it takes for the read data to appear at the local interface after initiating the read request.
- Write latency—the amount of time it takes for the write data to appear at the memory interface after initiating the write request.

Latency of the memory interface depends on its configuration and traffic patterns, therefore you should simulate your system to determine precise latency values. The numbers presented in this chapter are typical values meant only as guidelines.

Latency found in simulation may differ from latency found on the board, because functional simulation does not consider board trace delays and differences in process, voltage, and temperature. For a given design on a given board, the latency found may differ by one clock cycle (for full-rate designs), or two clock cycles (for quarter-rate or half-rate designs) upon resetting the board. The same design can yield different latencies on different boards.

**Note:** For a half-rate controller, the local side frequency is half of the memory interface frequency. For a full-rate controller, the local side frequency is equal to the memory interface frequency.

## DDR2 SDRAM LATENCY

The following table shows the DDR2 SDRAM latency in full-rate memory clock cycles.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

**Table 13-1: DDR2 SDRAM Controller Latency (In Full-Rate Memory Clock Cycles) (1)(2)**

Latency in Full-Rate Memory Clock Cycles							
Rate	Controller Address & Command	PHY Address & Command	Memory Maximum Read	PHY Read Return	Controller Read Return	Round Trip	Round Trip Without Memory
Half	10	EWL: 3	3-7	6	4	EWL: 26–30	EWL: 23
		OWL: 4				OWL: 27–31	OWL: 24
Full	5	0	3-7	4	10	22–26	19

Notes to Table:

1. EWL = Even write latency
2. OWL = Odd write latency

## DDR3 SDRAM LATENCY

The following table shows the DDR3 SDRAM latency in full-rate memory clock cycles.

**Table 13-2: DDR3 SDRAM Controller Latency (In Full-Rate Memory Clock Cycles) (1)(2)(3)(4)**

Latency in Full-Rate Memory Clock Cycles							
Rate	Controller Address & Command	PHY Address & Command	Memory Maximum Read	PHY Read Return	Controller Read Return	Round Trip	Round Trip Without Memory
Quarter	20	EWER : 8	5–11	EWER: 16	8	EWER: 57–63	EWER: 52
		EWOR: 8		EWOR: 17		EWOR: 58–64	EWOR: 53
		OWER: 11		OWER: 17		OWER: 61–67	OWER: 56
		OWOR: 11		OWOR: 14		OWOR: 58–64	OWOR: 53

Latency in Full-Rate Memory Clock Cycles							
Rate	Controller Address & Command	PHY Address & Command	Memory Maximum Read	PHY Read Return	Controller Read Return	Round Trip	Round Trip Without Memory
Half	10	EWER: 3	5–11	EWER: 7	4	EWER: 29–35	EWER: 24
		EWOR: 3		EWOR: 6		EWOR: 28–34	EWOR: 23
		OWER: 4		OWER: 6		OWER: 29–35	OWER: 24
		OWOR: 4		OWOR: 7		OWOR: 30–36	OWOR: 25
Full	5	0	5–11	4	10	24–30	19

Notes to Table:

1. EWER = Even write latency and even read latency
2. EWOR = Even write latency and odd read latency
3. OWER = Odd write latency and even read latency
4. OWOR = Odd write latency and odd read latency

## LPDDR2 SDRAM LATENCY

The following table shows the LPDDR2 SDRAM latency in full-rate memory clock cycles.

**Table 13-3: LPDDR2 SDRAM Controller Latency (In Full-Rate Memory Clock Cycles) <sup>(1)(2)(3)(4)</sup>**

Latency in Full-Rate Memory Clock Cycles							
Rate	Controller Address & Command	PHY Address & Command	Memory Maximum Read	PHY Read Return	Controller Read Return	Round Trip	Round Trip Without Memory
Half	10	EWER: 3	5–11	EWER: 7	4	EWER: 29–35	EWER: 24
		EWOR: 3		EWOR: 6		EWOR: 28–34	EWOR: 23
		OWER: 4		OWER: 6		OWER: 29–35	OWER: 24
		OWOR: 4		OWOR: 7		OWOR: 30–36	OWOR: 25
Full	5	0	5–11	4	10	24–30	19

Notes to Table:

1. EWER = Even write latency and even read latency
2. EWOR = Even write latency and odd read latency
3. OWER = Odd write latency and even read latency
4. OWOR = Odd write latency and odd read latency

## QDR II and QDR II+ SRAM Latency

The following table shows the latency in full-rate memory clock cycles.

**Table 13-4: QDR II Latency (In Full-Rate Memory Clock Cycles) <sup>(1)</sup>**

Latency in Full-Rate Memory Clock Cycles							
Rate	Controller Address & Command	PHY Address & Command	Memory Maximum Read	PHY Read Return	Controller Read Return	Round Trip	Round Trip Without Memory
Half 1.5 RL	2	5.5	1.5	7.0	0	16	14.5
Half 2.0 RL	2	5.5	2.0	6.5	0	16	14.0
Half 2.5 RL	2	5.5	2.5	6.0	0	16	13.5
Full 1.5 RL	2	1.5	1.5	4.0	1	10	8.5

Latency in Full-Rate Memory Clock Cycles							
Rate	Controller Address & Command	PHY Address & Command	Memory Maximum Read	PHY Read Return	Controller Read Return	Round Trip	Round Trip Without Memory
Full 2.0 RL	2	1.5	2.0	4.5	1	11	9.0
Full 2.5 RL	2	1.5	2.5	4.0	1	11	8.5

Note to Table:

1. RL = Read latency

## RLDRAM II Latency

The following table shows the latency in full-rate memory clock cycles.

**Table 13-5: RLDRAm II Latency (In Full-Rate Memory Clock Cycles) <sup>(1)(2)</sup>**

Latency in Full-Rate Memory Clock Cycles							
Rate	Controller Address & Command	PHY Address & Command	Memory Maximum Read	PHY Read Return	Controller Read Return	Round Trip	Round Trip Without Memory
Half	4	EWL: 1	3–8	EWL: 4	0	EWL: 12–17	EWL: 9
		OWL: 2		OWL: 4		OWL: 13–18	OWL: 10
Full	2	1	3–8	4	0	10–15	7

Notes to Table:

1. EWL = Even write latency
2. OWL = Odd write latency

## RLDRAM 3 Latency

The following table shows the latency in full-rate memory clock cycles.

**Table 13-6: RLDRAM 3 Latency (In Full-Rate Memory Clock Cycles)**

Latency in Full-Rate Memory Clock Cycles						
Rate	PHY Address & Command	Memory Maximum Read	PHY Read Return	Controller Read Return	Round Trip	Round Trip Without Memory
Quarter	7	3–16	18	0	28–41	25
Half	4	3–16	6	0	13–26	10

## Variable Controller Latency

The variable controller latency feature allows you to take advantage of lower latency for variations designed to run at lower frequency. When deciding whether to vary the controller latency from the default value of 1, be aware of the following considerations:

- Reduced latency can help achieve a reduction in resource usage and clock cycles in the controller, but might result in lower fMAX.
- Increased latency can help achieve greater fMAX, but might consume more clock cycles in the controller and result in increased resource usage.

If you select a latency value that is inappropriate for the target frequency, the system displays a warning message in the text area at the bottom of the parameter editor.

You can change the controller latency by altering the value of the **Controller Latency** setting in the **Controller Settings** section of the **General Settings** tab of the QDR II and QDR II+ SRAM controller with UniPHY parameter editor.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Maintenance release.
December 2013	2013.12.16	Updated latency data for QDR II/II+.
November 2012	2.1	<ul style="list-style-type: none"> <li>Added latency information for RLDRAM 3.</li> <li>Changed chapter number from 9 to 11.</li> </ul>

Date	Version	Changes
June 2012	2.0	<ul style="list-style-type: none"><li>Added latency information for LPDDR2.</li><li>Added Feedback icon.</li></ul>
November 2011	1.0	<ul style="list-style-type: none"><li>Consolidated latency information from <i>11.0 DDR2 and DDR3 SDRAM Controller with UniPHY User Guide</i>, <i>QDR II and QDR II+ SRAM Controller with UniPHY User Guide</i>, and <i>RDRAM II Controller with UniPHY IP User Guide</i>.</li><li>Updated data for 11.1.</li></ul>

# Timing Diagrams for UniPHY IP

# 14

2016.05.02

EMI\_RM



Subscribe



Send Feedback

The following topics contain timing diagrams for UniPHY-based external memory interface IP for supported protocols.

## DDR2 Timing Diagrams

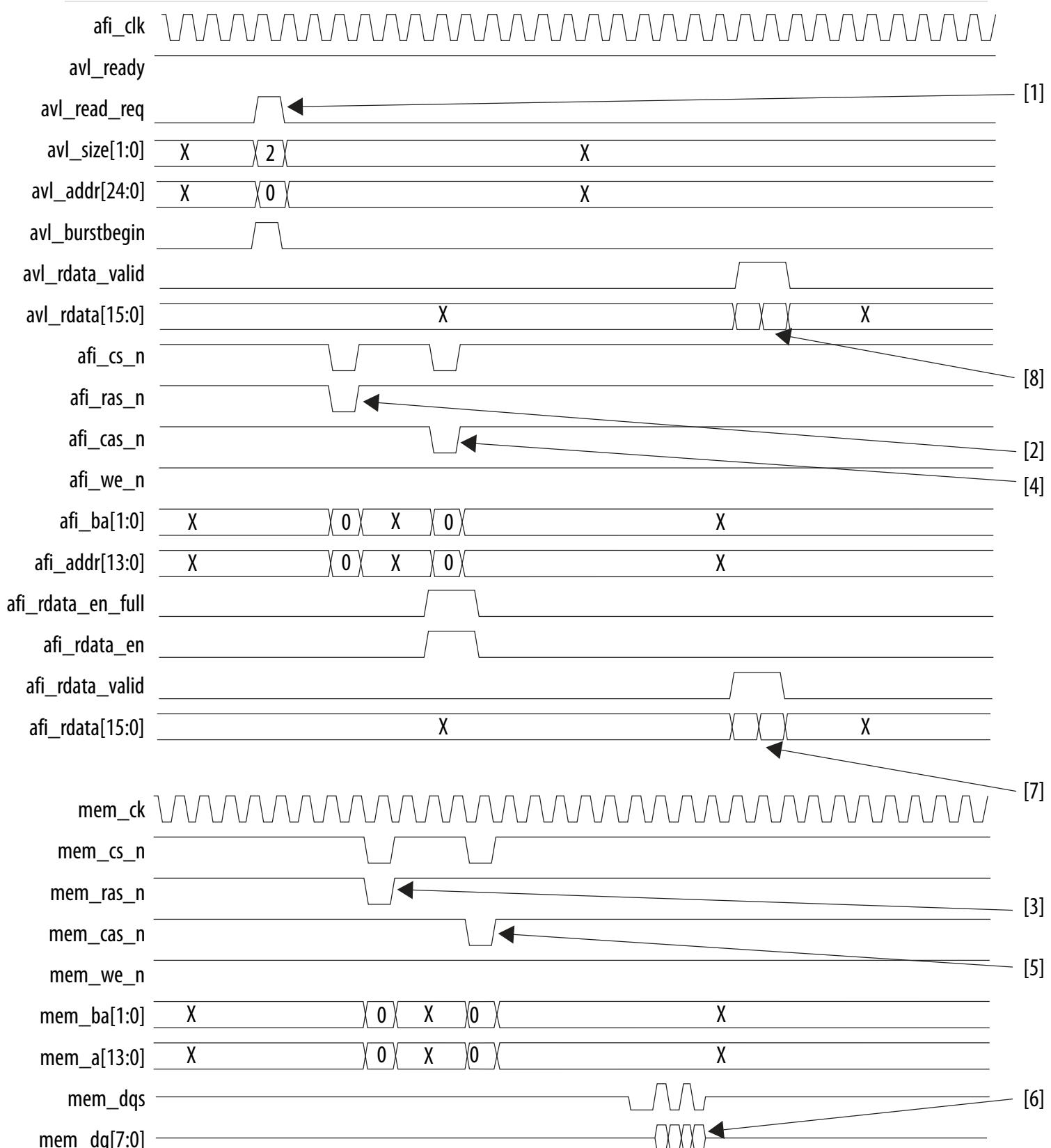
This topic contains timing diagrams for UniPHY-based external memory interface IP for DDR2 protocols.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

The following figures present timing diagrams based on a Stratix III device:

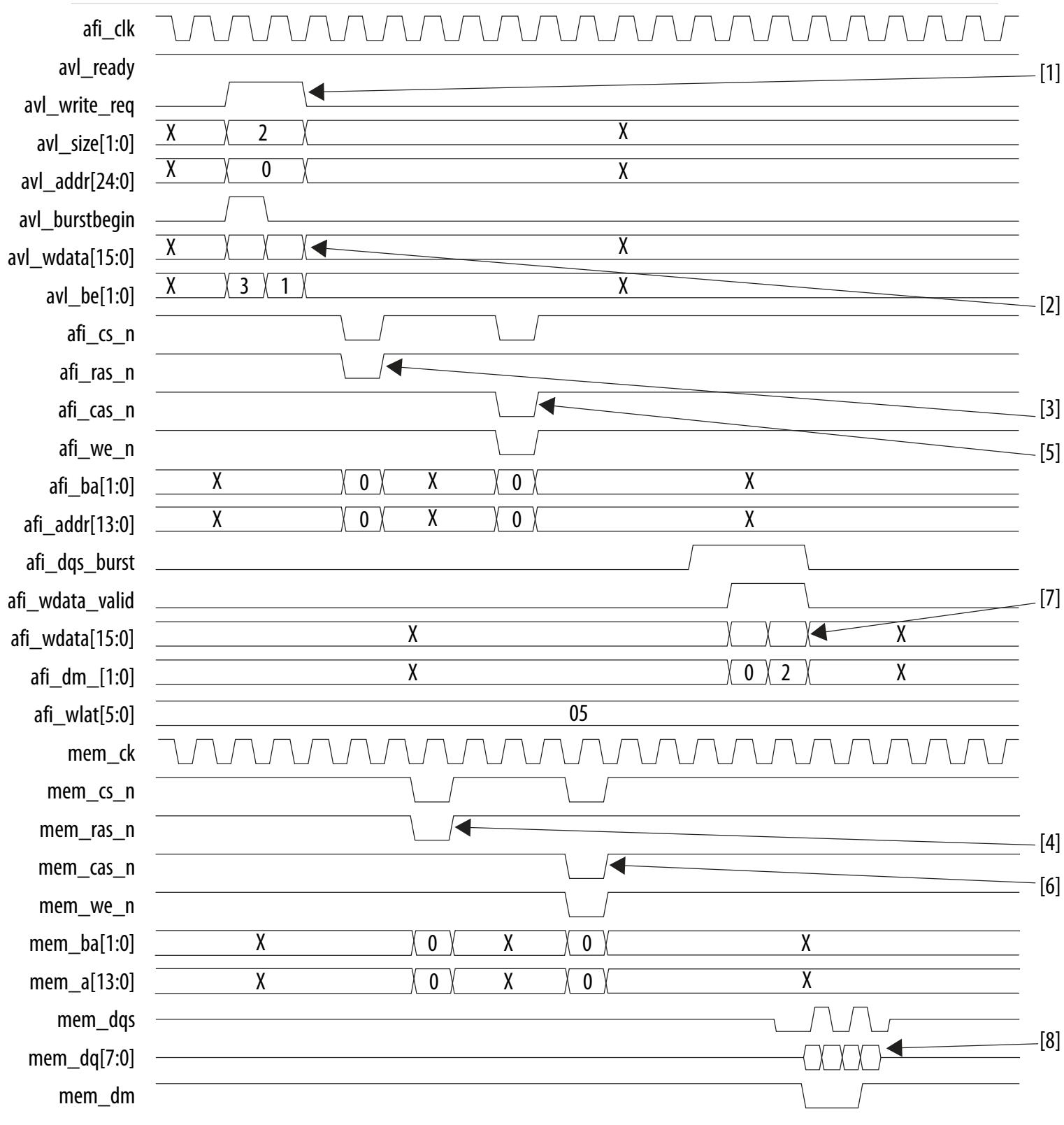
**Figure 14-1: Full-Rate DDR2 SDRAM Read**



Notes for the above Figure:

1. Controller receives read command.
2. Controller issues activate command to PHY.
3. PHY issues activate command to memory.
4. Controller issues read command to PHY.
5. PHY issues read command to memory.
6. PHY receives read data from memory.
7. Controller receives read data from PHY.
8. User logic receives read data from controller.

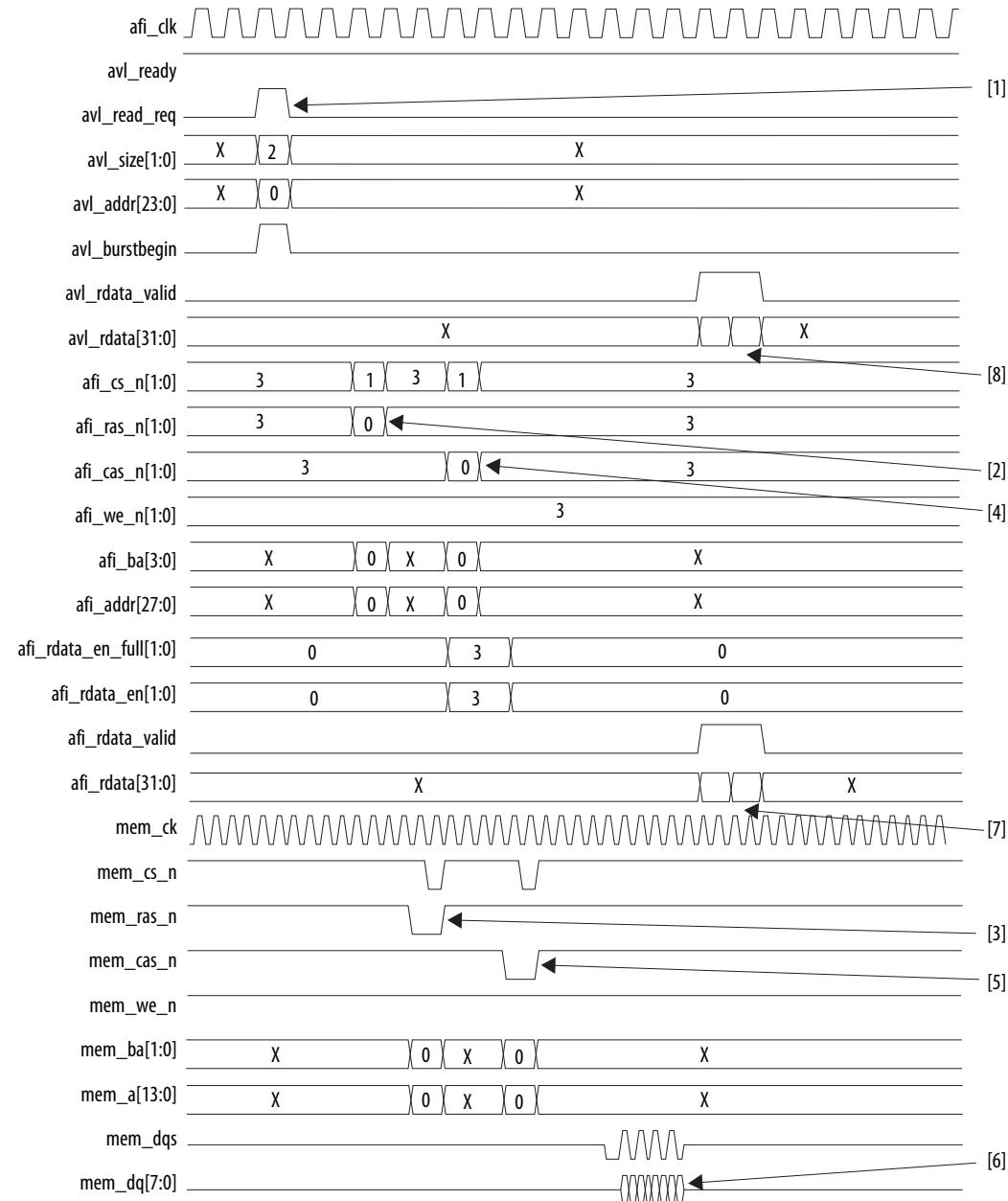
Figure 14-2: Full-Rate DDR2 SDRAM Write



Notes for the above Figure:

1. Controller receives write command.
2. Controller receives write data.
3. Controller issues activate command to PHY.
4. PHY issues activate command to memory.
5. Controller issues write command to PHY.
6. PHY issues write command to memory.
7. Controller sends write data to PHY.
8. PHY sends write data to memory.

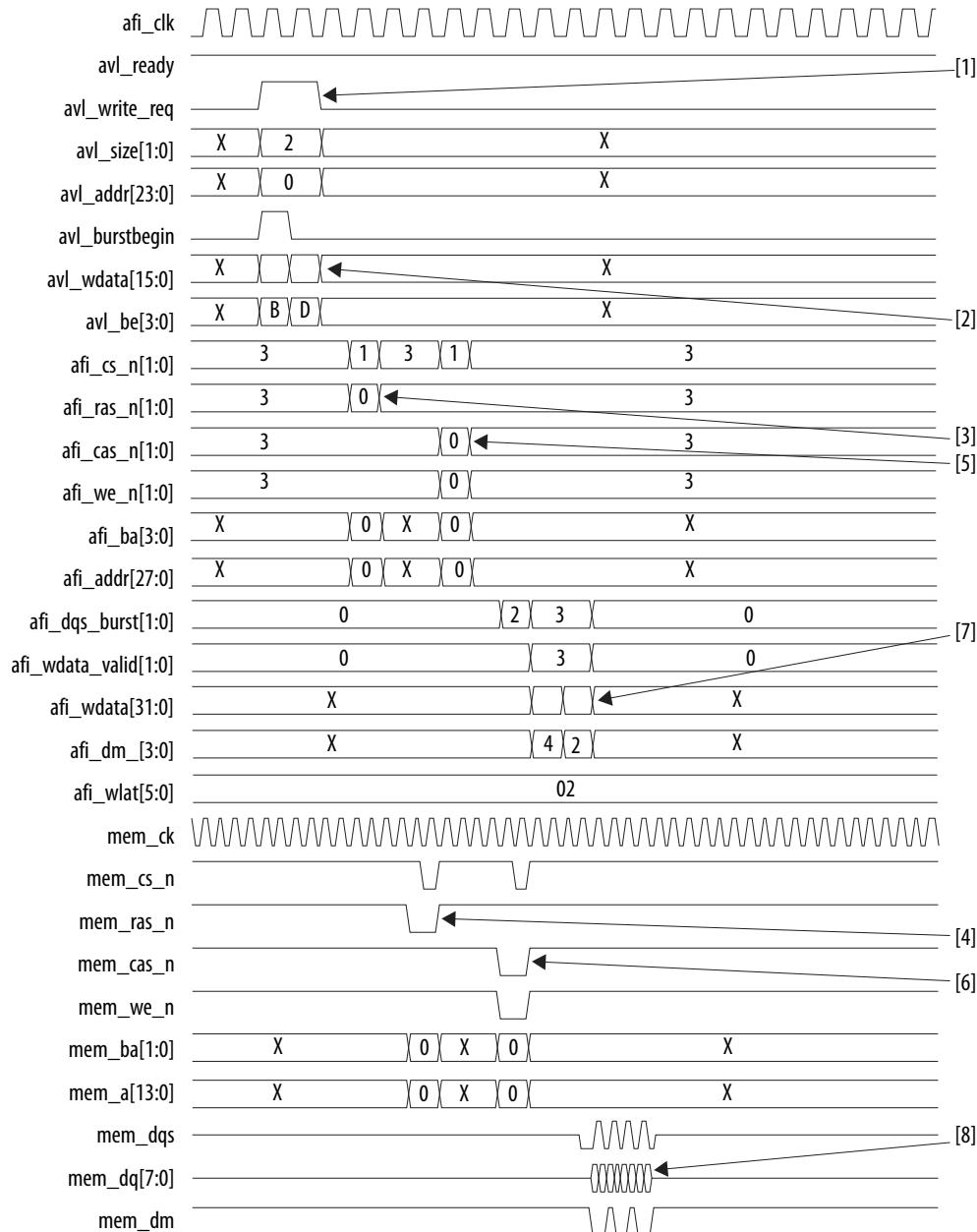
Figure 14-3: Half-Rate DDR2 SDRAM Read



Notes for the above Figure:

1. Controller receives read command.
2. Controller issues activate command to PHY.
3. PHY issues activate command to memory.
4. Controller issues read command to PHY.
5. PHY issues read command to memory.
6. PHY receives read data from memory.
7. Controller receives read data from PHY.
8. User logic receives read data from controller.

**Figure 14-4: Half-Rate DDR2 SDRAM Write**



Notes for the above Figure:

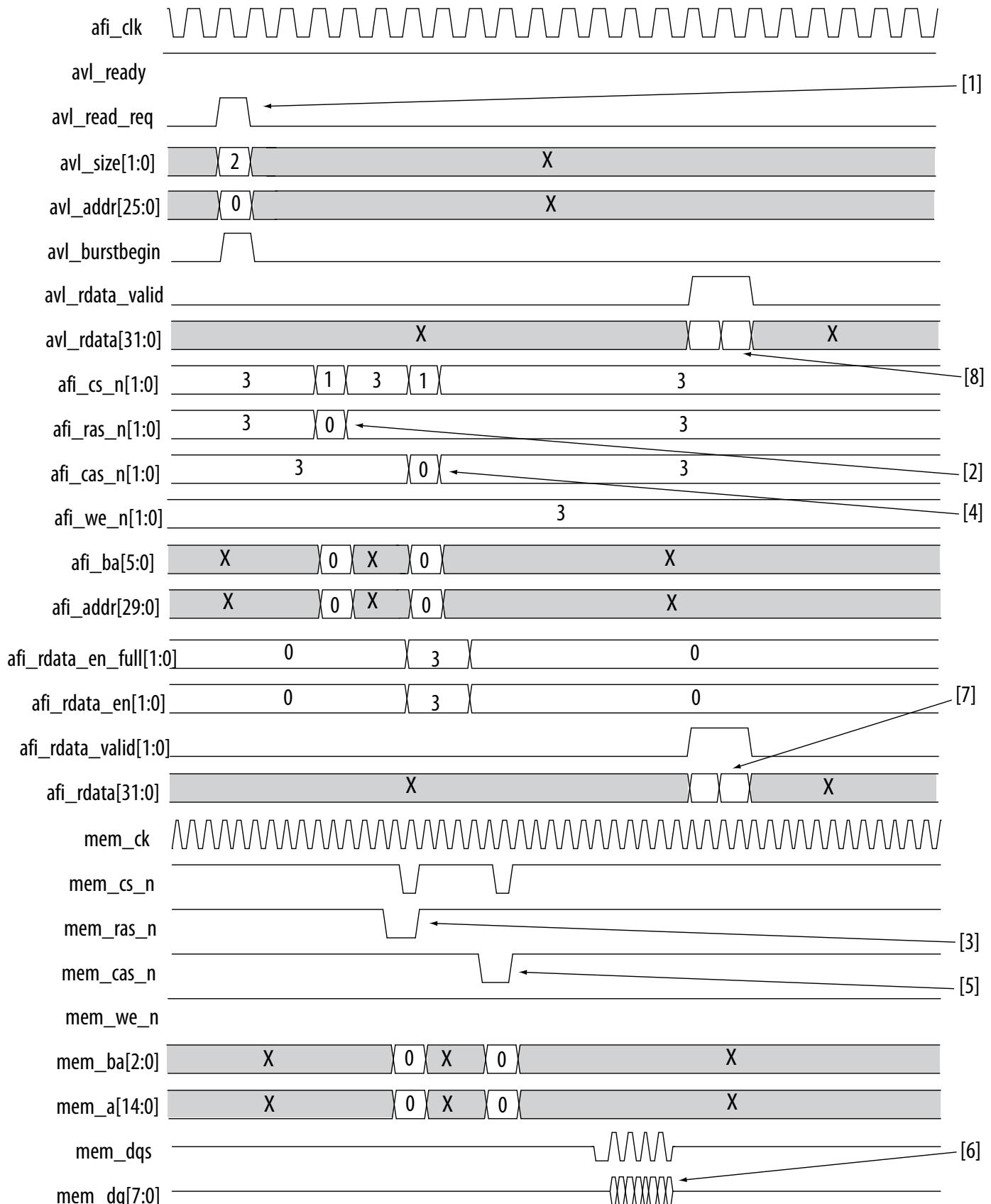
1. Controller receives write command.
2. Controller receives write data.
3. Controller issues activate command to PHY.
4. PHY issues activate command to memory.
5. Controller issues write command to PHY.
6. PHY issues write command to memory.
7. Controller sends write data to PHY.
8. PHY sends write data to memory.

## DDR3 Timing Diagrams

This topic contains timing diagrams for UniPHY-based external memory interface IP for DDR3 protocols.

The following figures present timing diagrams based on a Stratix III device

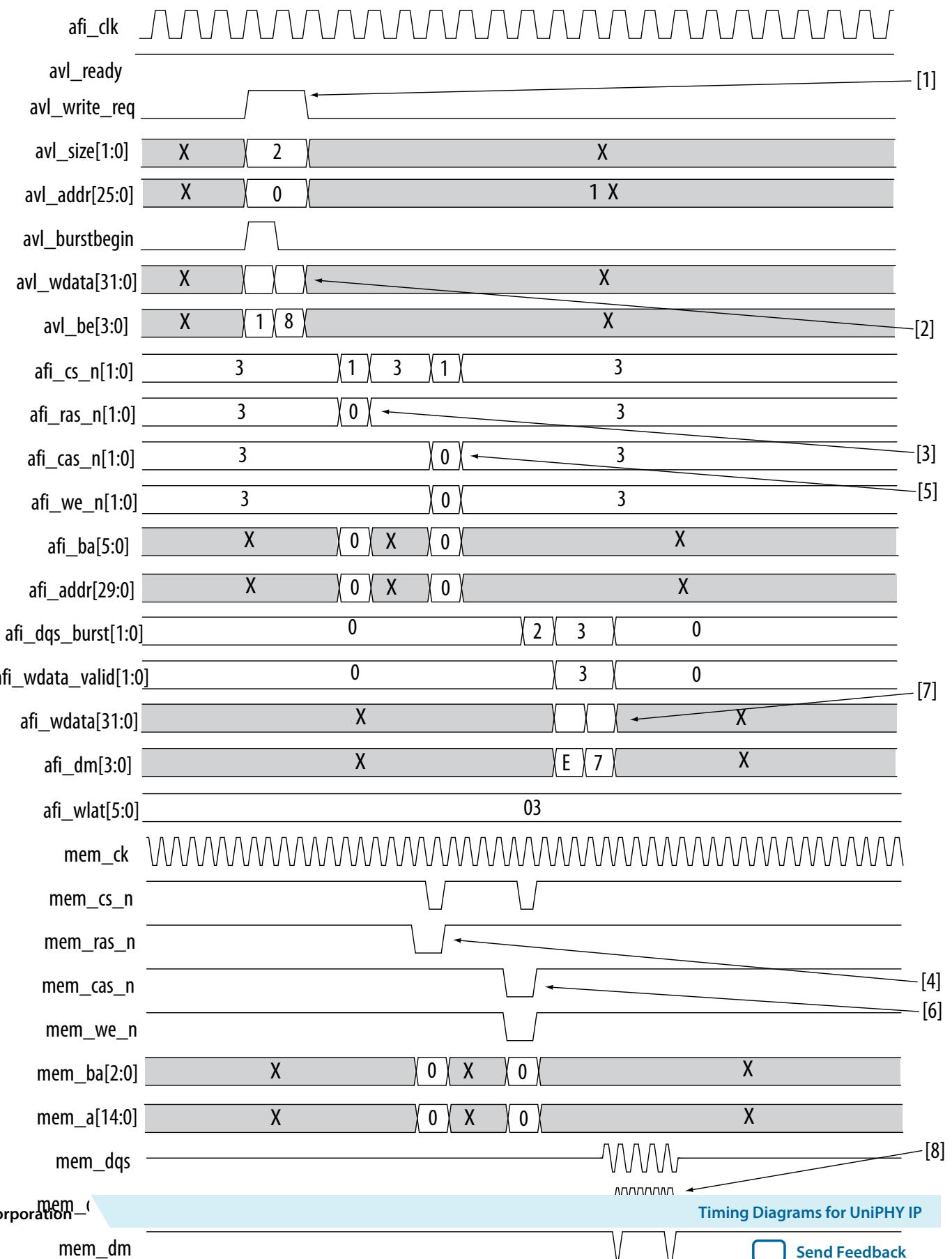
Figure 14-5: Half-Rate DDR3 SDRAM Read



Notes for the above Figure:

1. Controller receives read command.
2. Controller issues activate command to PHY.
3. PHY issues activate command to memory.
4. Controller issues read command to PHY.
5. PHY issues read command to memory.
6. PHY receives read data from memory.
7. Controller receives read data from PHY.
8. User logic receives read data from controller.

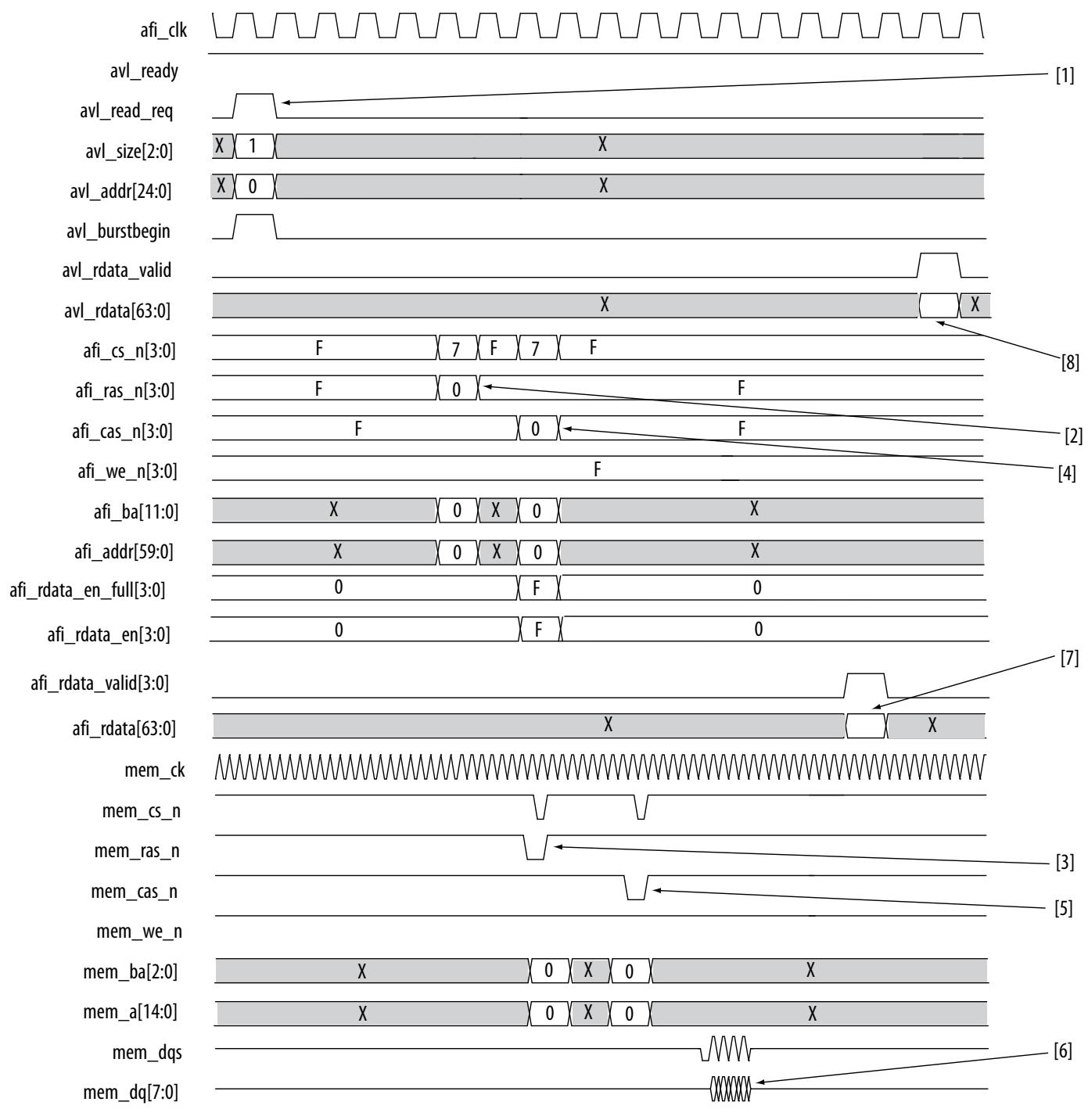
Figure 14-6: Half-Rate DDR3 SDRAM Writes



Notes for the above Figure:

1. Controller receives write command.
2. Controller receives write data.
3. Controller issues activate command to PHY.
4. PHY issues activate command to memory.
5. Controller issues write command to PHY.
6. PHY issues write command to memory.
7. Controller sends write data to PHY.
8. PHY sends write data to memory.

Figure 14-7: Quarter-Rate DDR3 SDRAM Reads

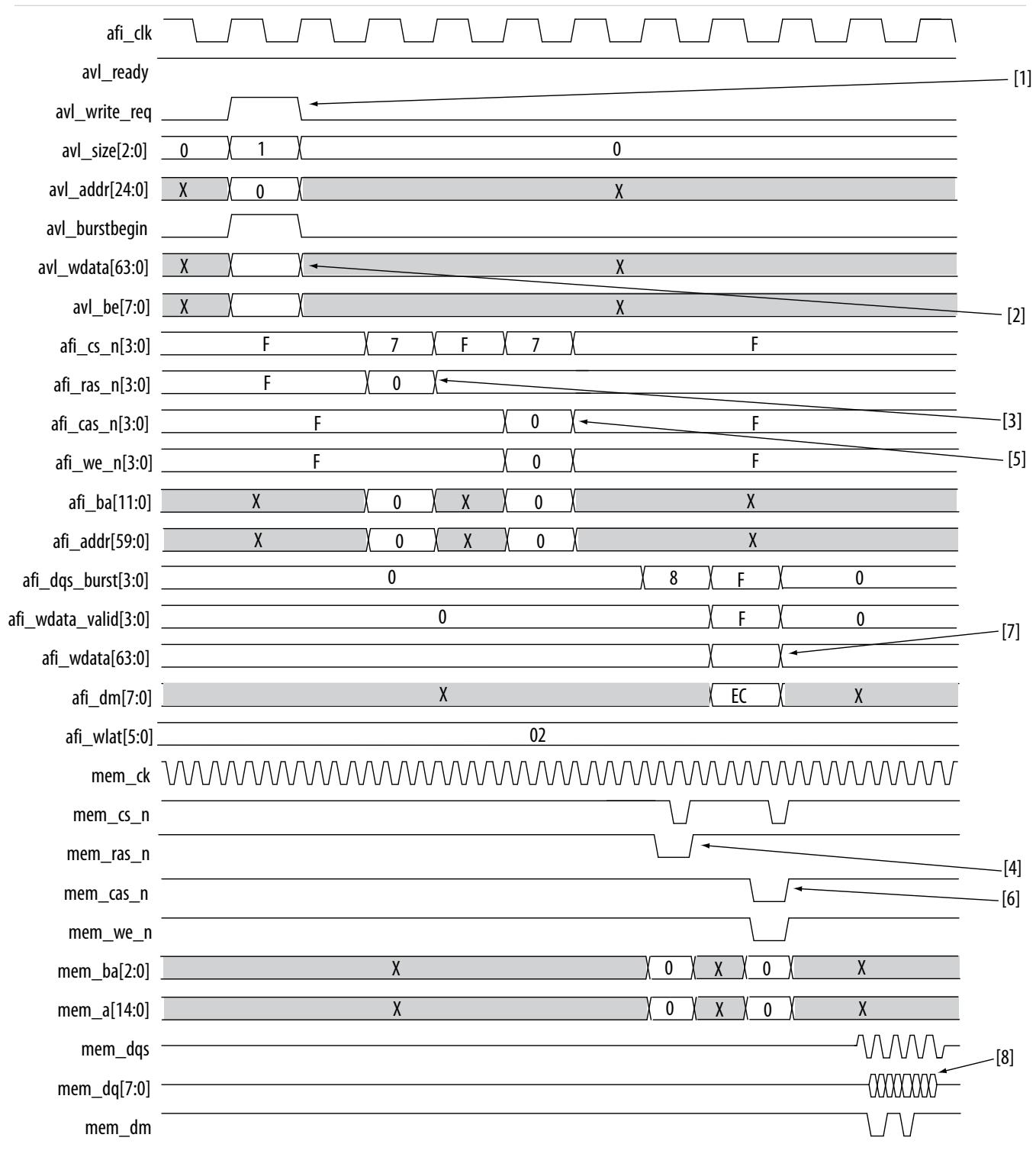


Notes for the above Figure:

1. Controller receives read command.
2. Controller issues activate command to PHY.
3. PHY issues activate command to memory.
4. Controller issues read command to PHY.

5. PHY issues read command to memory.
6. PHY receives read data from memory
7. Controller receives read data from PHY
8. User logic receives read data from controller.

Figure 14-8: Quarter-Rate DDR3 SDRAM Writes



Notes for the above Figure:

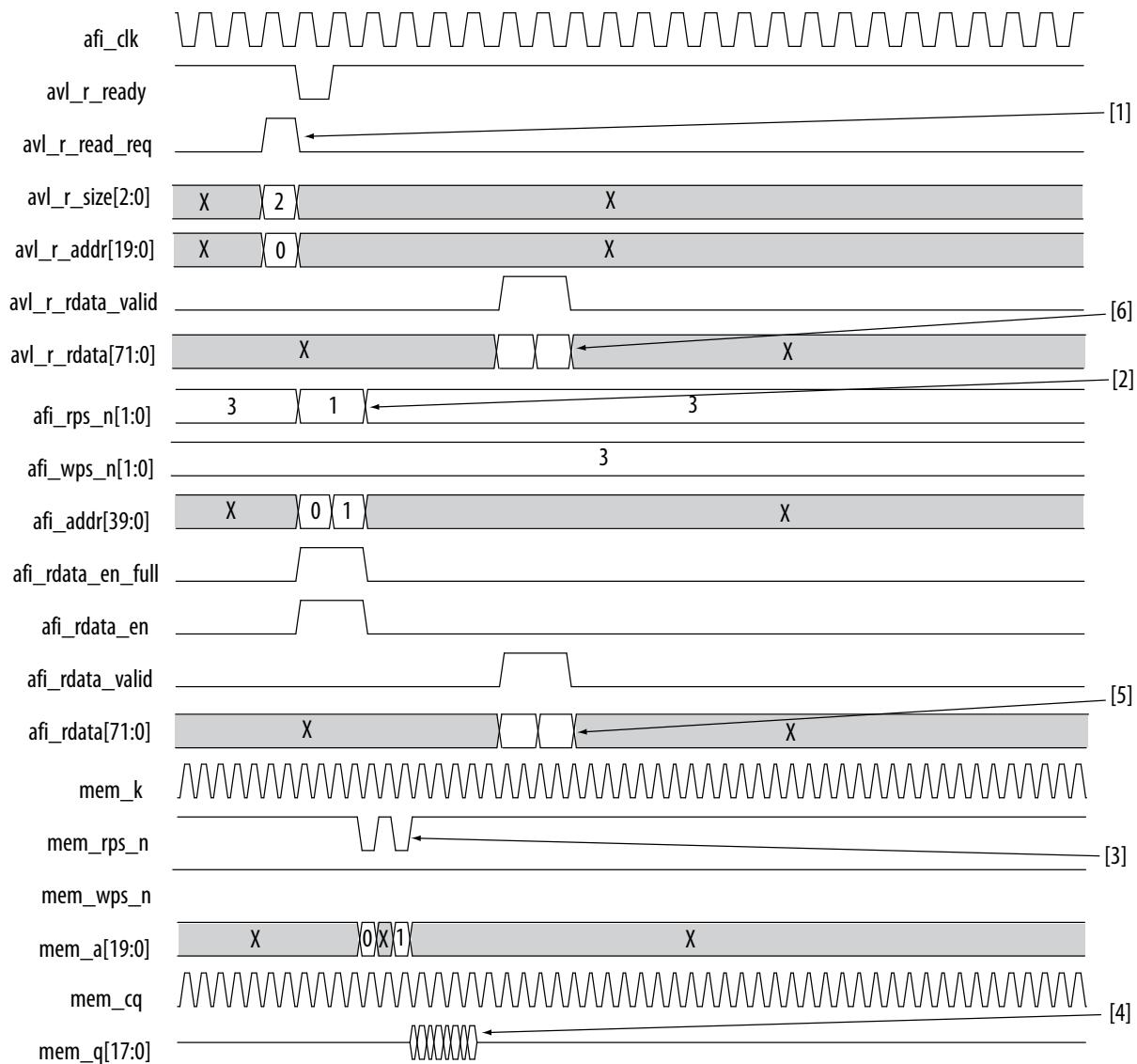
1. Controller receives write command.
2. Controller receives write data.
3. Controller issues activate command to PHY
4. PHY issues activate command to memory.
5. Controller issues write command to PHY
6. PHY issues write command to memory
7. Controller sends write data to PHY
8. PHY sends write data to memory.

## QDR II and QDR II+ Timing Diagrams

This topic contains timing diagrams for UniPHY-based external memory interface IP for QDR II and QDR II+ protocols.

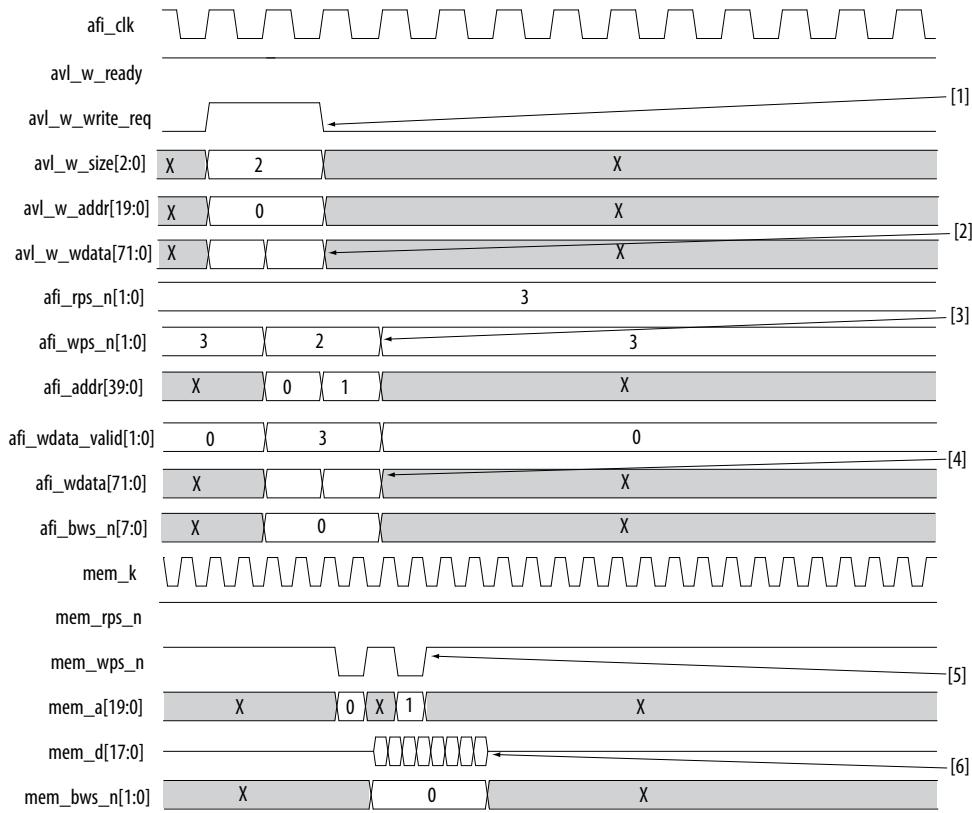
The following figures present timing diagrams, based on a Stratix III device:

**Figure 14-9: Half-Rate QDR II and QDR II+ SRAM Read**



Notes for the above Figure:

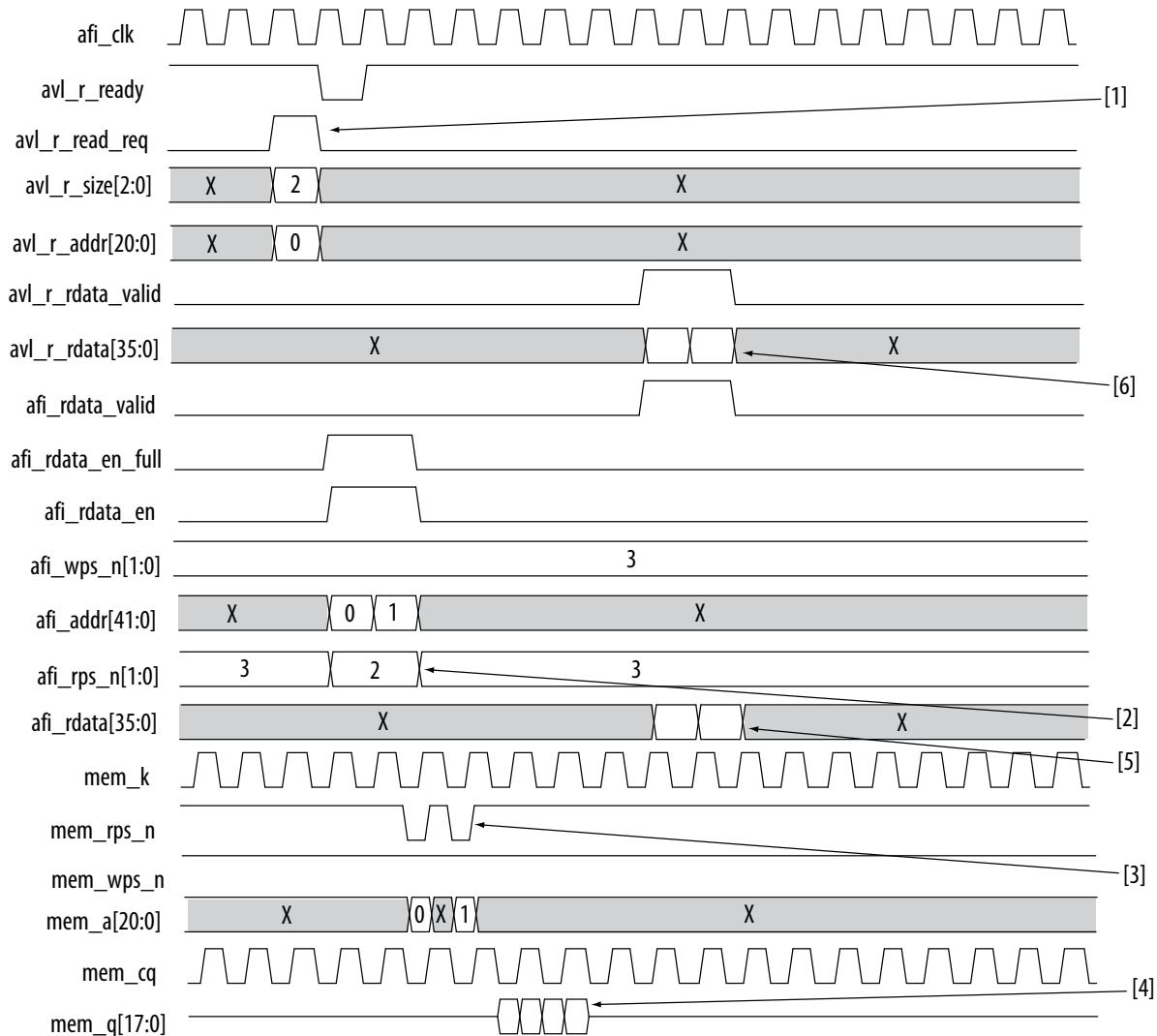
1. Controller receives read command.
2. Controller issues two read commands to PHY.
3. PHY issues two read commands to memory.
4. PHY receives read data from memory.
5. Controller receives read data from PHY.
6. User logic receives read data from controller.

**Figure 14-10: Half-Rate QDR II and QDR II+ SRAM Write**

Notes for the above Figure:

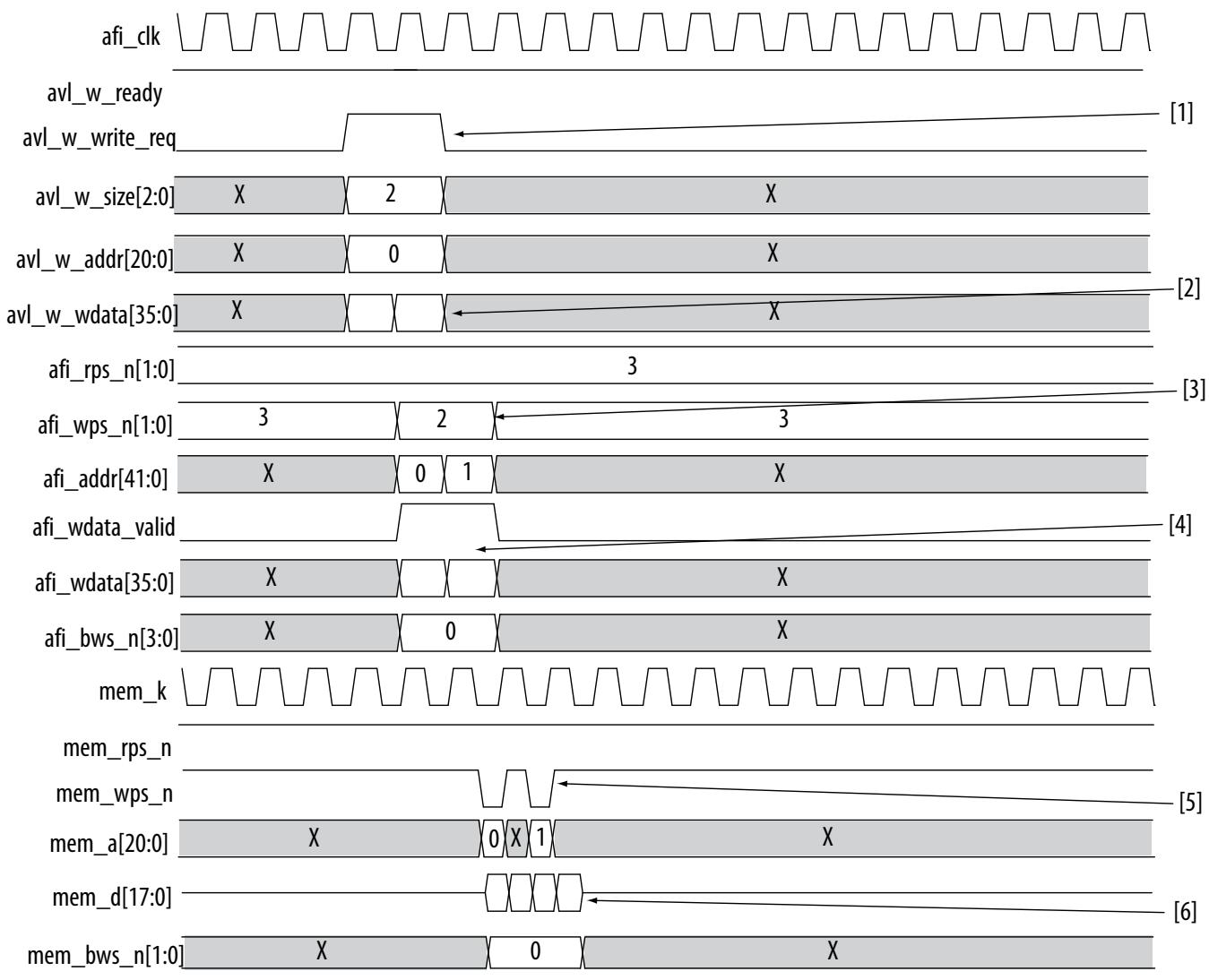
1. Controller receives write command.
2. Controller receives write data.
3. Controller issues two write commands to PHY.
4. Controller sends write data to PHY.
5. PHY issues two write commands to memory.
6. PHY sends write data to memory.

Figure 14-11: Full-Rate QDR II and QDR II+ SRAM Read



Notes for the above Figure:

1. Controller receives read command.
2. Controller issues two read commands to PHY.
3. PHY issues two read commands to memory.
4. PHY receives read data from memory.
5. Controller receives read data from PHY.
6. User logic receives read data from controller.

**Figure 14-12: Full-Rate QDR II and QDR II+ SRAM Write**

Notes for the above Figure:

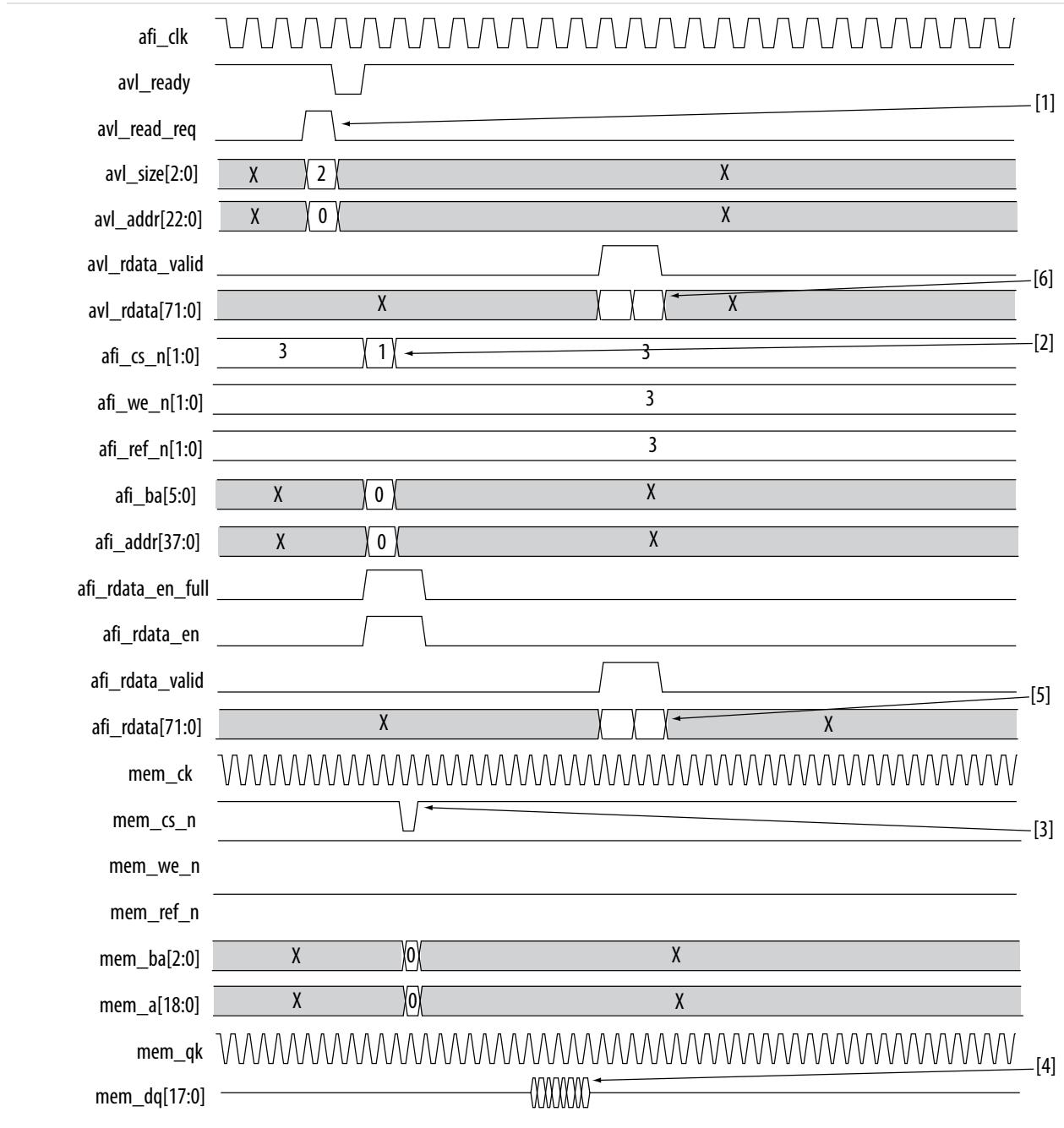
1. Controller receives write command.
2. Controller receives write data.
3. Controller issues two write commands to PHY.
4. Controller sends write data to PHY.
5. PHY issues two write commands to memory.
6. PHY sends write data to memory.

## RLDRAM II Timing Diagrams

This topic contains timing diagrams for UniPHY-based external memory interface IP for RLDRAM protocols.

The following figures present timing diagrams, based on a Stratix III device:

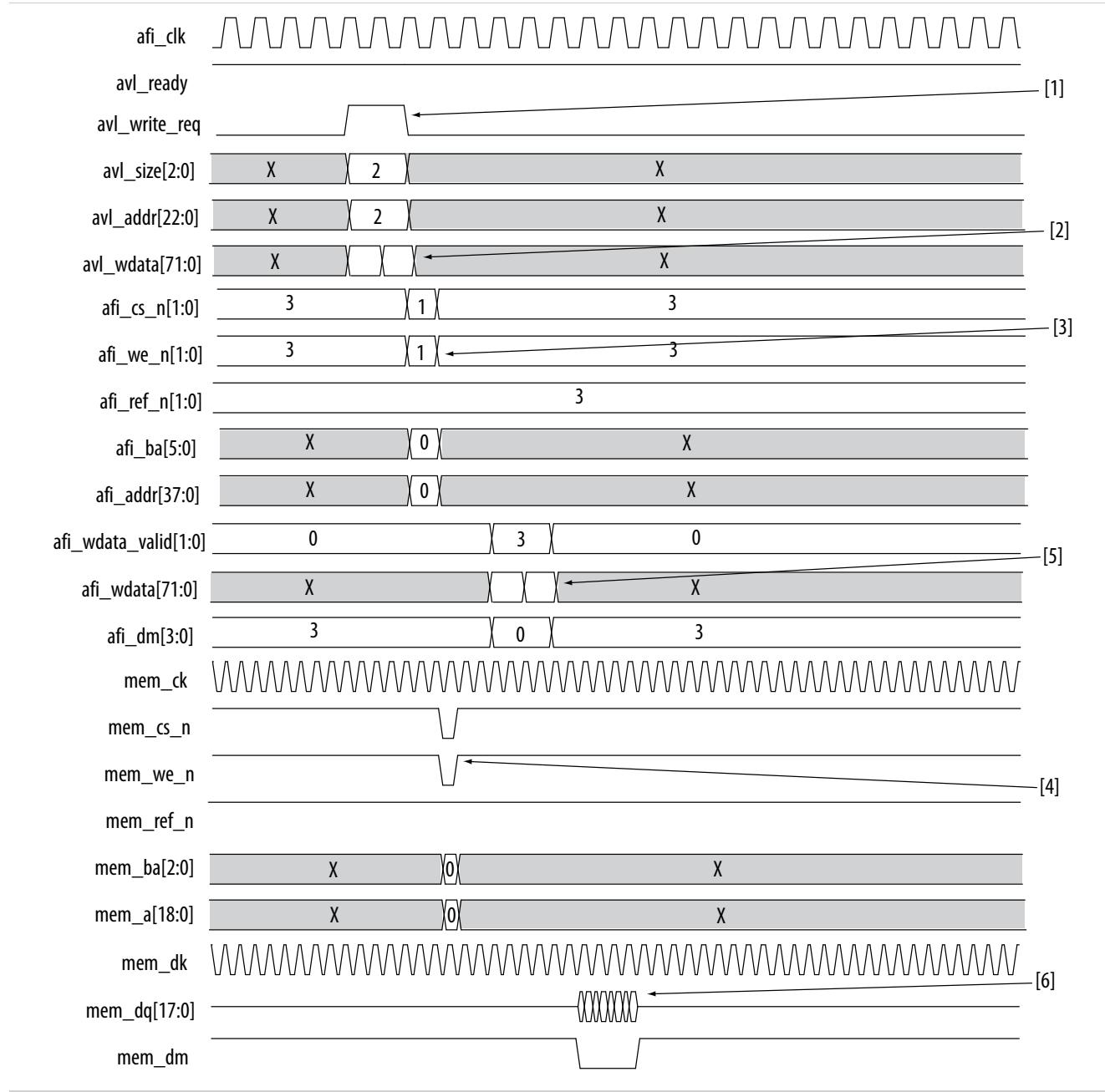
**Figure 14-13: Half-Rate RLDRAM II Read**



Notes for the above Figure:

1. Controller receives read command.
2. Controller issues read command to PHY.
3. PHY issues read command to memory.
4. PHY receives read data from memory.
5. Controller receives read data from PHY.
6. User logic receives read data from controller.

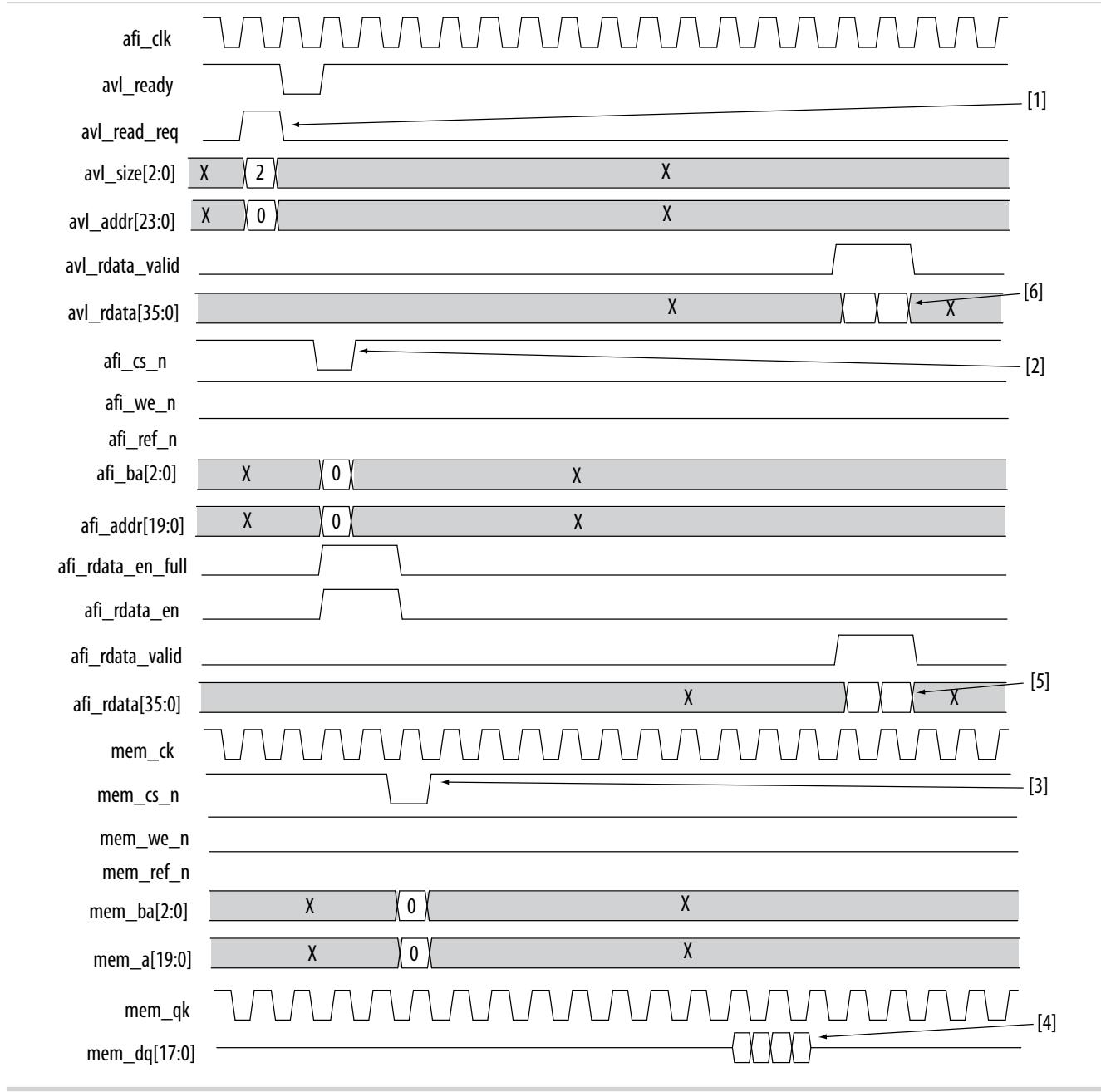
**Figure 14-14: Half-Rate RLDRAM II Write**



Notes for the above Figure:

1. Controller receives write command.
2. Controller receives write data.
3. Controller issues write command to PHY.
4. PHY issues write command to memory.
5. Controller sends write data to PHY.
6. PHY sends write data to memory.

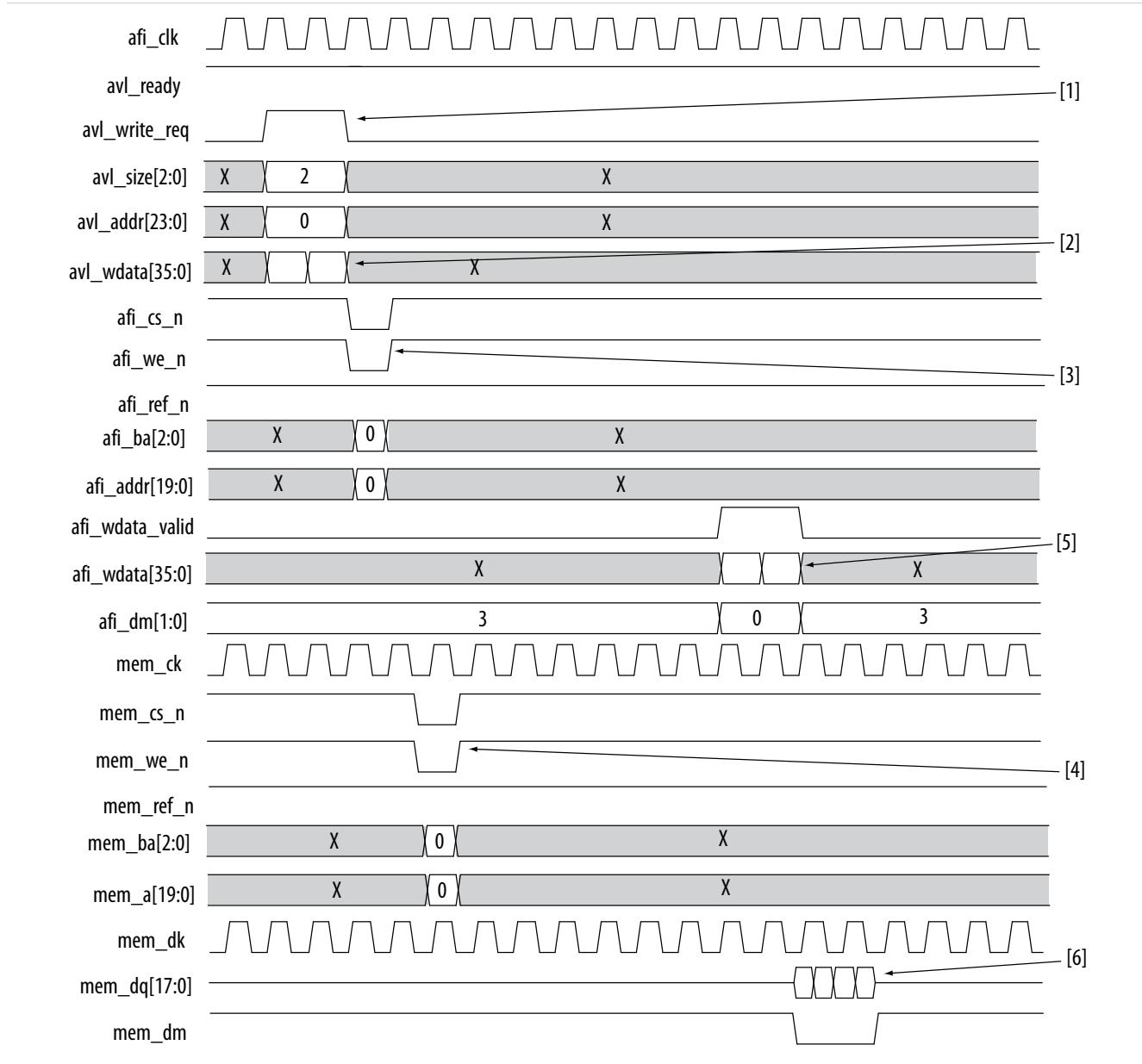
**Figure 14-15: Full-Rate RLDRAM II Read**



Notes for the above Figure:

1. Controller receives read command.
2. Controller issues read command to PHY.
3. PHY issues read command to memory.
4. PHY receives read data from memory.
5. Controller receives read data from PHY.
6. User logic receives read data from controller.

**Figure 14-16: Full-Rate RLDRAM II Write**



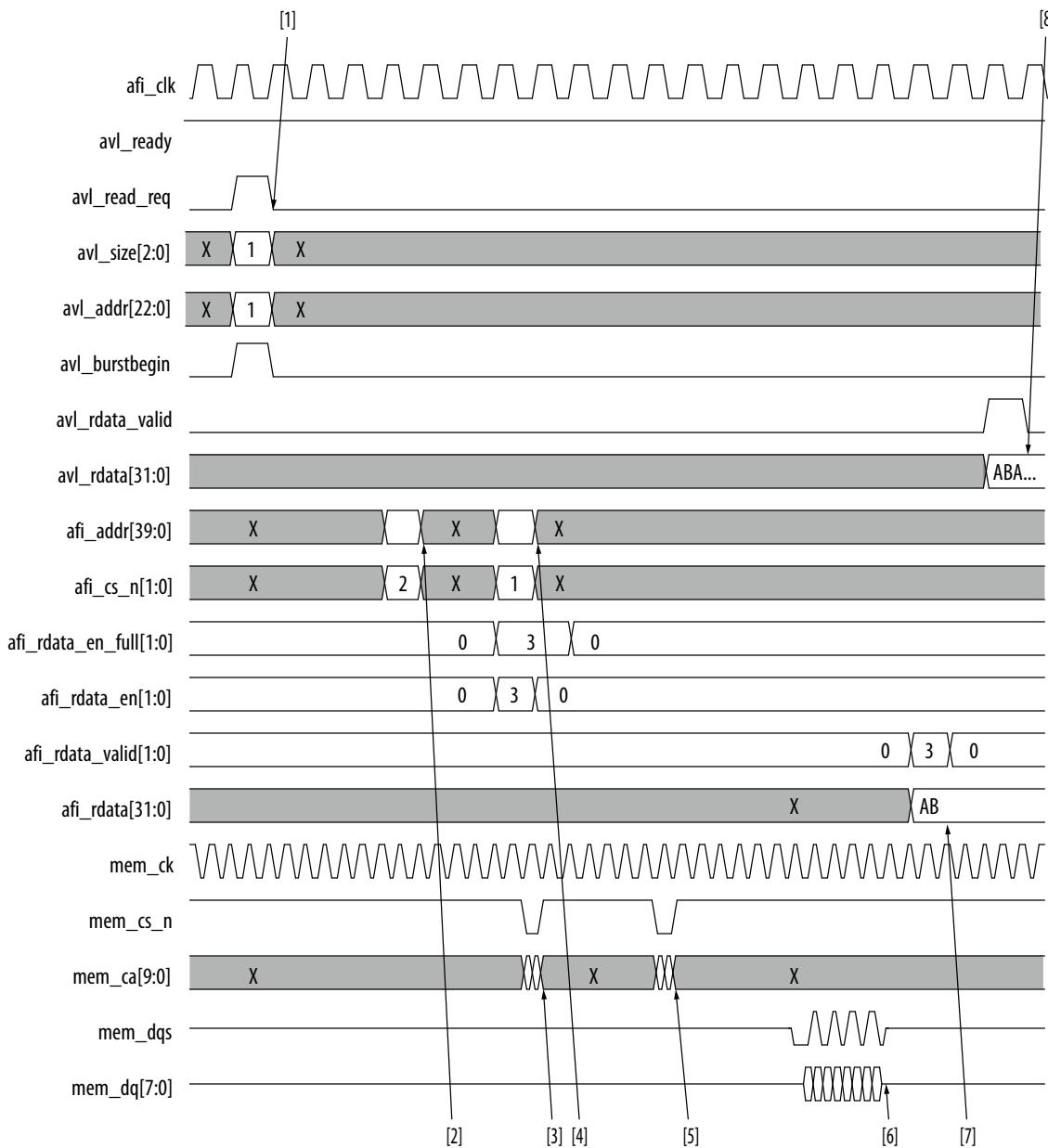
Notes for the above Figure:

1. Controller receives write command.
2. Controller receives write data.
3. Controller issues write command to PHY.
4. PHY issues write command to memory.
5. Controller sends write data to PHY.
6. PHY sends write data to memory.

## LPDDR2 Timing Diagrams

This topic contains timing diagrams for UniPHY-based external memory interface IP for LPDDR2 protocols.

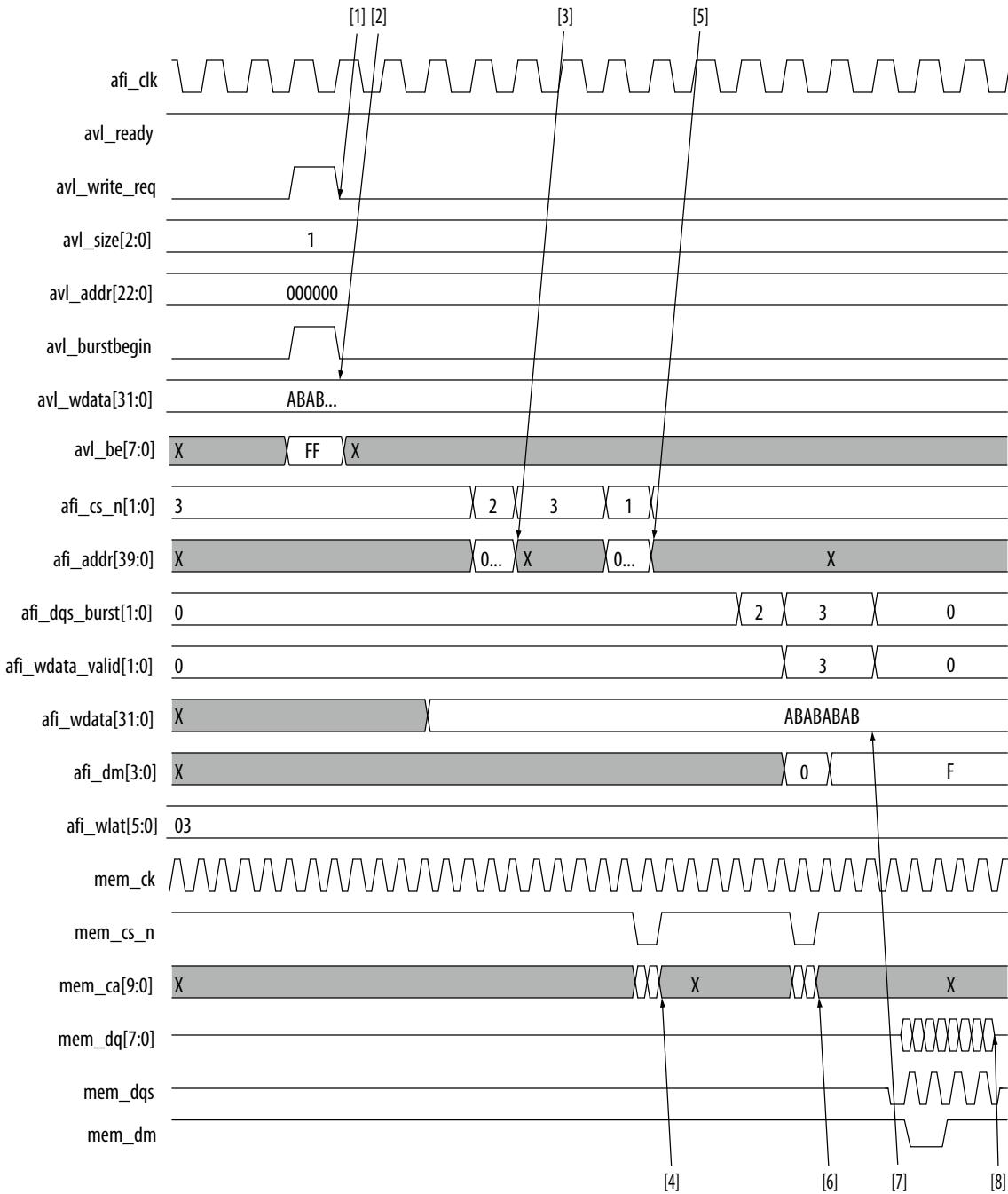
Figure 14-17: Half-Rate LPDDR2 Read



Notes for the above Figure:

1. Controller receives read command.
2. Controller issues activate command to PHY.
3. PHY issues activate command to memory.
4. Controller issues read command to PHY.
5. PHY issues read command to memory.
6. PHY receives read data from memory.
7. Controller receives read data from PHY.
8. User logic receives read data from controller.

Figure 14-18: Half-Rate LPDDR2 Write

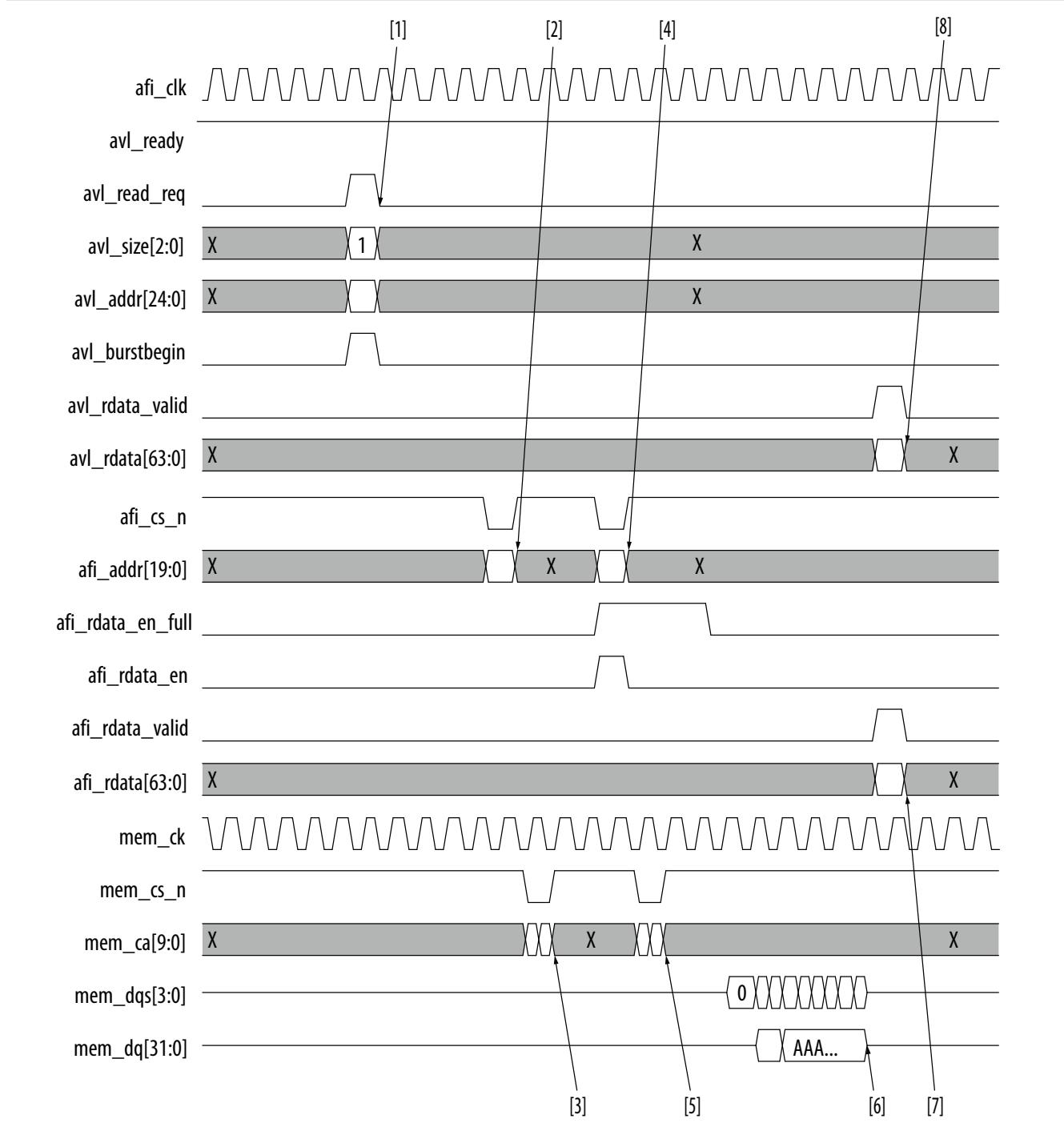


Notes for the above Figure:

1. Controller receives write command.
2. Controller receives write data.
3. Controller issues activate command to PHY.
4. PHY issues activate command to memory.
5. Controller issues write command to PHY.

6. PHY issues write command to memory.
7. Controller sends write data to PHY.
8. PHY sends write data to memory.

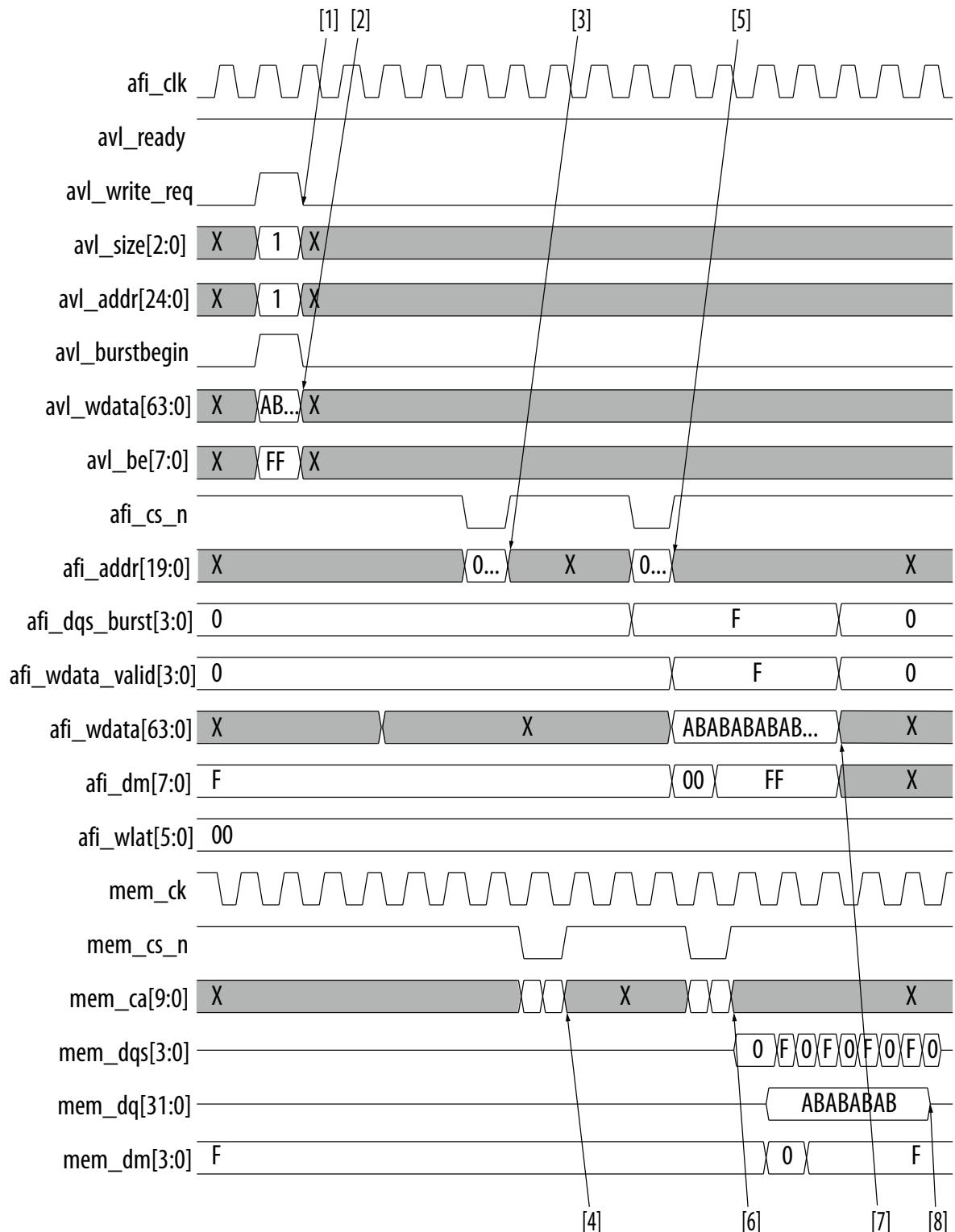
Figure 14-19: Full-Rate LPDDR2 Read



Notes for the above Figure:

1. Controller receives read command.
2. Controller issues activate command to PHY.
3. PHY issues activate command to memory.
4. Controller issues read command to PHY.
5. PHY issues read command to memory.
6. PHY receives read data from memory.
7. Controller receives read data from PHY.
8. User logic receives read data from controller.

Figure 14-20: Full-Rate LPDDR2 Write



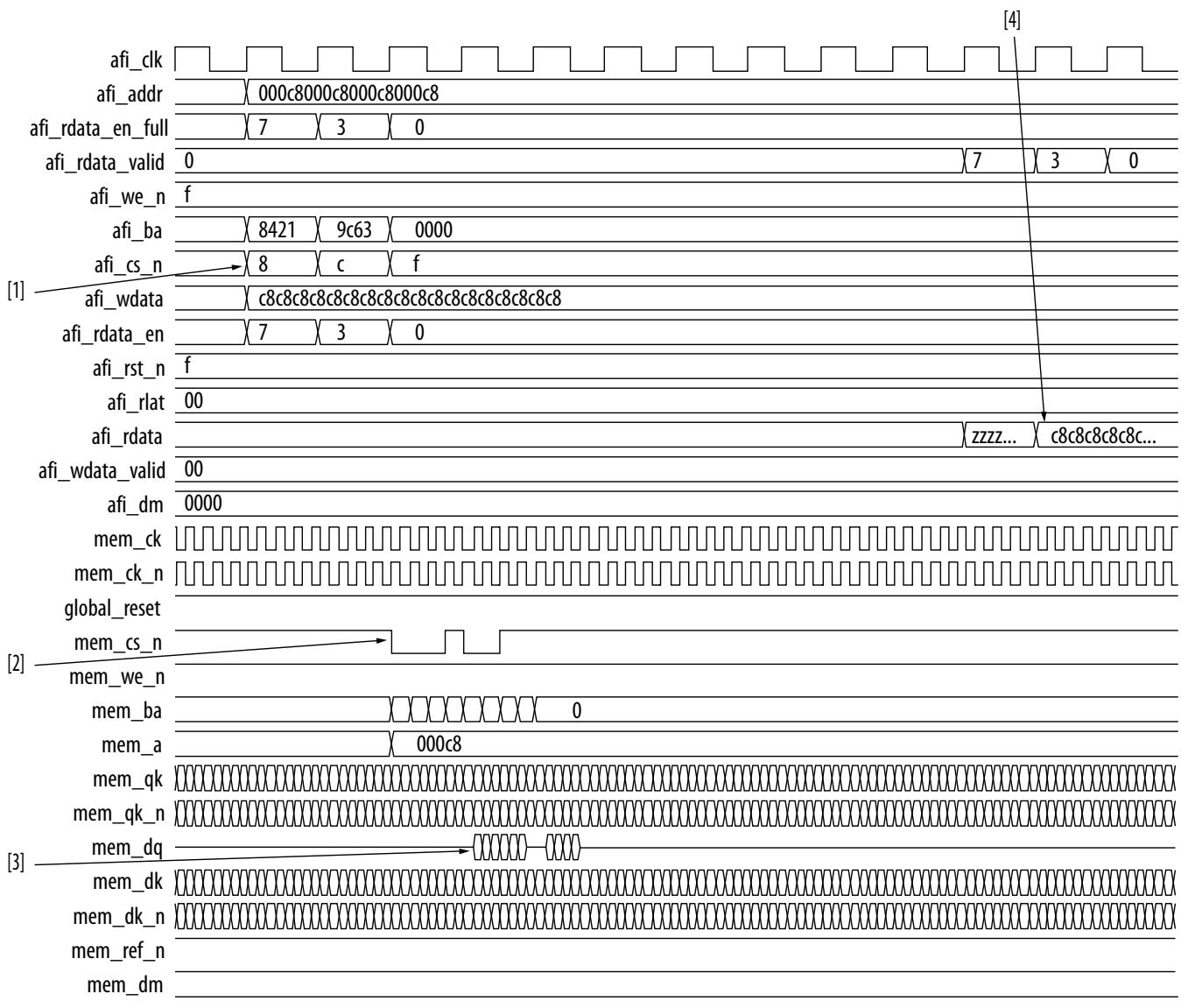
Notes for the above Figure:

1. Controller receives write command.
2. Controller receives write data.
3. Controller issues activate command to PHY.
4. PHY issues activate command to memory.
5. Controller issues write command to PHY.
6. PHY issues write command to memory.
7. Controller sends write data to PHY.
8. PHY sends write data to memory.

## RLDRAM 3 Timing Diagrams

This topic contains timing diagrams for UniPHY-based external memory interface IP for RLDRAm 3 protocols.

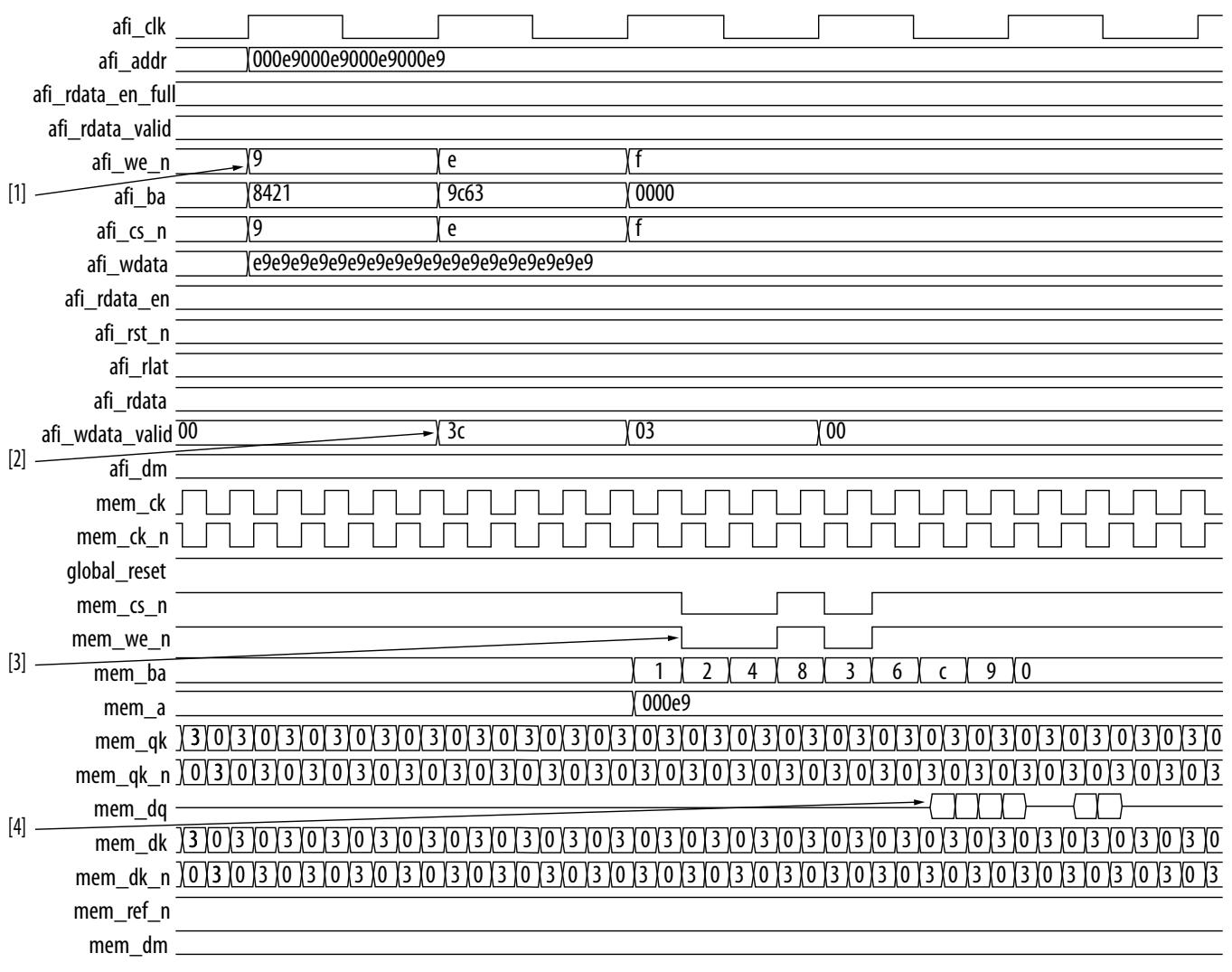
**Figure 14-21: Quarter-Rate RLDRAM 3 Read**



Notes for the above Figure:

1. Controller issues read command to PHY.
  2. PHY issues read command to memory.
  3. PHY receives data from memory.
  4. Controller receives read data from PHY.

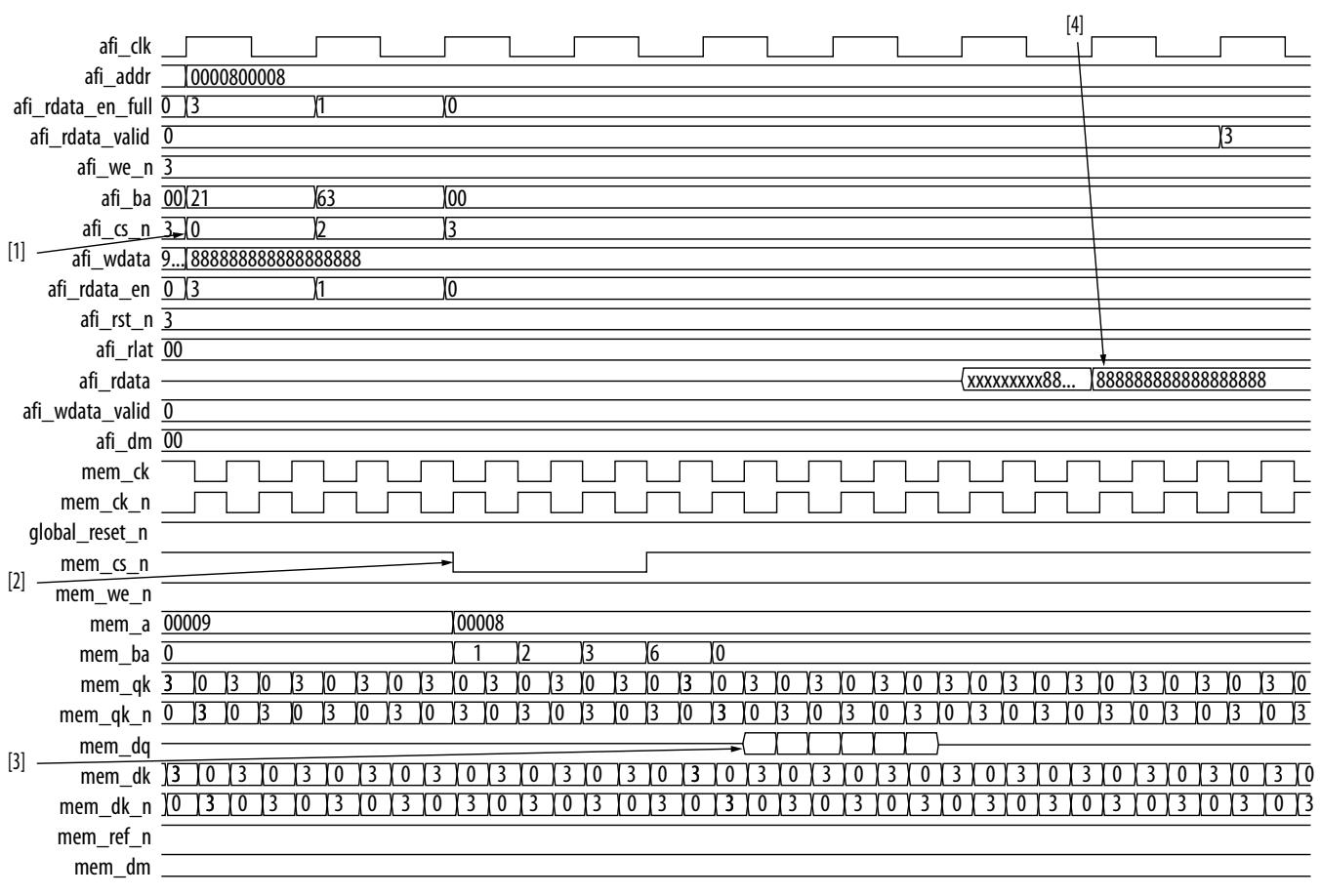
**Figure 14-22: Quarter-Rate RLDRAM 3 Write**



Notes for the above Figure:

1. Controller issues write command to PHY.
  2. Data ready from controller for PHY.
  3. PHY issues write command to memory.
  4. PHY sends read data to memory.

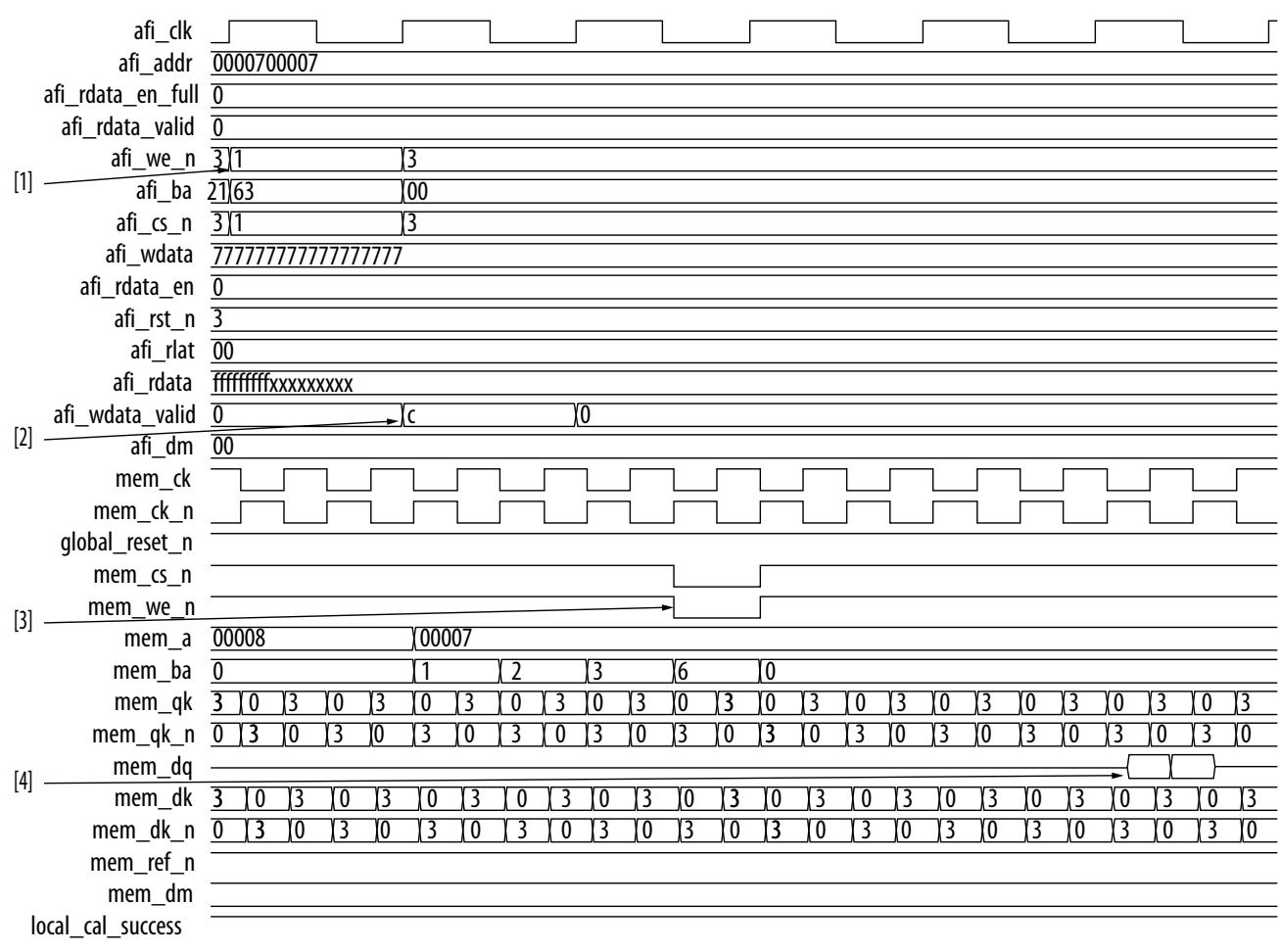
**Figure 14-23: Half-rate RLDRAM 3 Read**



### Notes for the above Figure:

1. Controller issues read command to PHY.
  2. PHY issues read command to memory.
  3. PHY receives data from memory.
  4. Controller receives read data from PHY.

**Figure 14-24: Half-Rate RLDRAM 3 Write**



Notes for the above Figure:

1. Controller issues write command to PHY.
  2. Data ready from controller for PHY.
  3. PHY issues write command to memory.
  4. PHY sends read data to memory.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Maintenance release.
May 2015	2015.05.04	Maintenance release.

Date	Version	Changes
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Maintenance release.
December 2013	2013.12.16	Updated timing diagrams for LPDDR2.
November 2012	2.1	<ul style="list-style-type: none"><li>Added timing diagrams for RLDRAM 3.</li><li>Changed chapter number from 10 to 12.</li></ul>
June 2012	2.0	<ul style="list-style-type: none"><li>Added timing diagrams for LPDDR2.</li><li>Added Feedback icon.</li></ul>
November 2011	1.1	<ul style="list-style-type: none"><li>Consolidated timing diagrams from <i>11.0 DDR2 and DDR3 SDRAM Controller with UniPHY User Guide</i>, <i>QDR II and QDR II+ SRAM Controller with UniPHY User Guide</i>, and <i>RLDRAM II Controller with UniPHY IP User Guide</i>.</li><li>Added Read and Write diagrams for DDR3 quarter-rate.</li></ul>

# External Memory Interface Debug Toolkit

# 15

2016.05.02

EMI\_RM



Subscribe



Send Feedback

The EMIF Toolkit lets you diagnose and debug calibration problems and produce margining reports for your external memory interface.

The toolkit is compatible with UniPHY-based external memory interfaces that use the Nios II-based sequencer, with toolkit communication enabled, and with Arria 10 EMIF IP. Toolkit communication is on by default in versions 10.1 and 11.0 of UniPHY IP; for version 11.1 and later, toolkit communication is on whenever debugging is enabled on the Diagnostics tab of the IP core interface.

The EMIF Toolkit can communicate with several different memory interfaces on the same device, but can communicate with only one memory device at a time.

**Note:** The EMIF Debug Toolkit does not support MAX 10 devices.

## User Interface

The EMIF toolkit provides a graphical user interface for communication with connections.

All functions provided in the toolkit are also available directly from the `quartus_sh` TCL shell, through the `external_memif_toolkit` TCL package. The availability of TCL support allows you to create scripts to run automatically from TCL. You can find information about specific TCL commands by running `help -pkg external_memif_toolkit` from the `quartus_sh` TCL shell.

If you want, you can begin interacting with the toolkit through the GUI, and later automate your workflow by creating TCL scripts. The toolkit GUI records a history of the commands that you run. You can see the command history on the History tab in the toolkit GUI.

## Communication

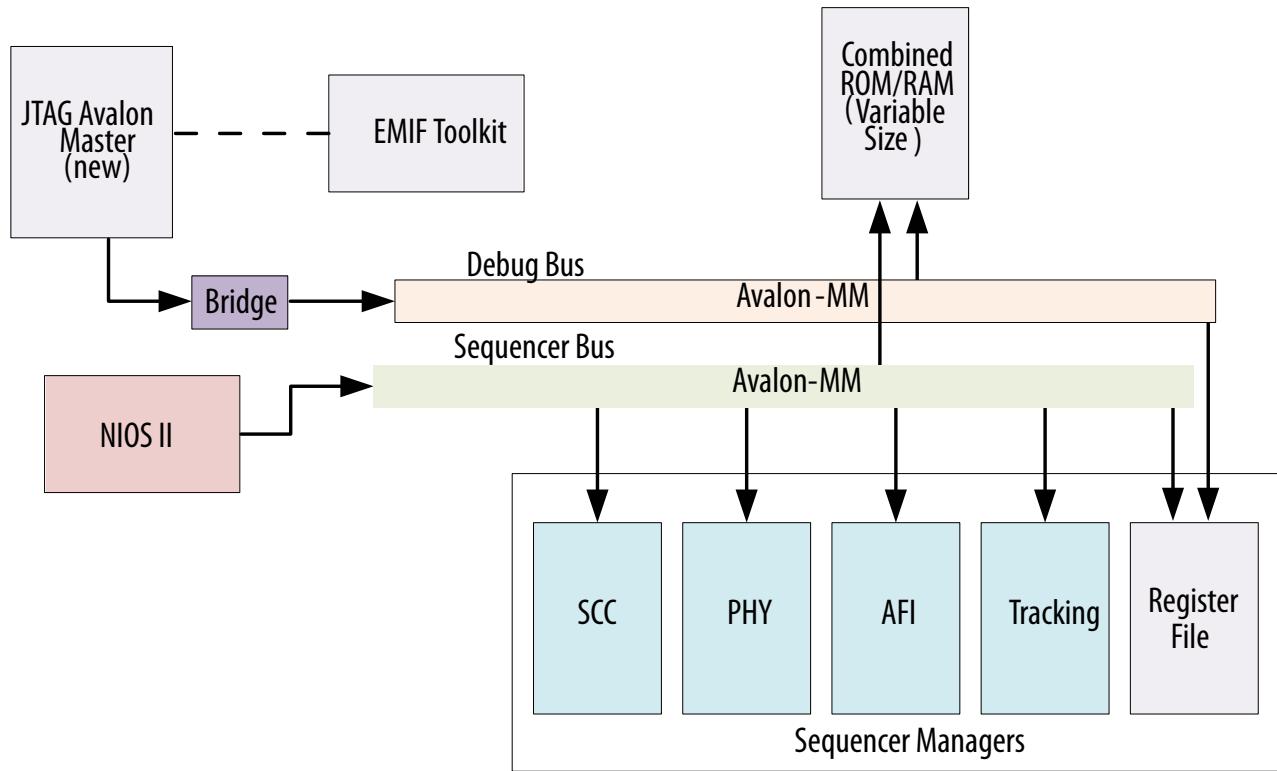
Communication between the EMIF Toolkit and external memory interface connections varies, depending on the connection type and version. In versions 10.1 and 11.0 of the EMIF IP, communication is achieved using direct communication to the Nios II-based sequencer. In version 11.1 and later, communication is achieved using a JTAG Avalon-MM master attached to the sequencer bus.

The following figure shows the structure of UniPHY-based IP version 11.1 and later, with JTAG Avalon-MM master attached to sequencer bus masters.

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

Figure 15-1: UniPHY IP Version 11.1 and Later, with JTAG Avalon-MM Master



## Calibration and Report Generation

The EMIF Toolkit uses calibration differently, depending on the version of external memory interface in use. For versions 10.1 and 11.0 interfaces, the EMIF Toolkit causes the memory interface to calibrate several times, to produce the data from which the toolkit generates its reports. In version 11.1 and later, report data is generated during calibration, without need to repeat calibration. For version 11.1 and later, generated reports reflect the result of the previous calibration, without need to recalibrate unless you choose to do so.

## Setup and Use

Before using the EMIF Toolkit, you should compile your design and program the target device with the resulting SRAM Object File (**.sof**). For designs compiled in the Quartus II software version 12.0 or earlier, debugging information is contained in the JTAG Debugging Information file (**.jdi**); however, for designs compiled in the Quartus II software version 12.1 or later, all debugging information resides in the **.sof** file.

You can run the toolkit using all your project files, or using only the Quartus Prime Project File (**.qpf**), Quartus Prime Settings File (**.qsf**), and **.sof** file; the **.jdi** file is also required for designs compiled prior to version 12.1. To ensure that all debugging information is correctly synchronized for designs compiled prior to version 12.1, ensure that the **.sof** and **.jdi** files that you used are generated during the same run of the Quartus Prime Assembler.

After you have programmed the target device, you can run the EMIF Toolkit and open your project. You can then use the toolkit to create connections to the external memory interface.

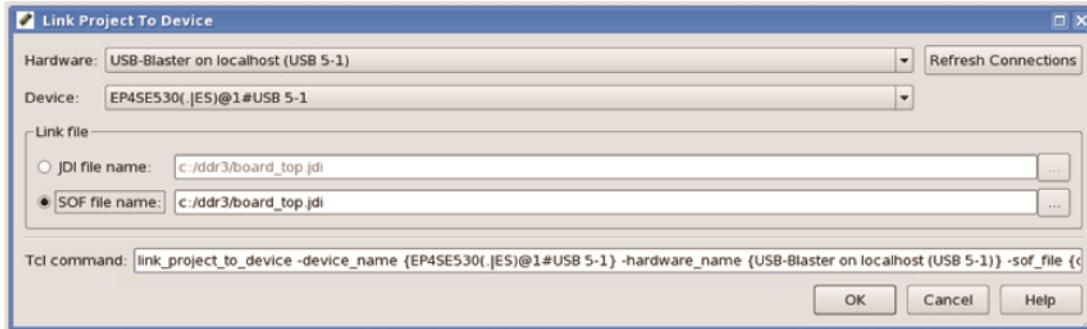
## General Workflow

To use the EMIF Toolkit, you must link your compiled project to a device, and create a communication channel to the connection that you want to examine.

### Linking the Project to a Device

1. To launch the toolkit, select External Memory Interface Toolkit from the Tools menu in the Quartus Prime software.
2. After you have launched the toolkit, open your project and click the **Initialize connections** task in the **Tasks** window, to initialize a list of all known connections.
3. To link your project to a specific device on specific hardware, perform the following steps:
  - a. Click the **Link Project to Device** task in the **Tasks** window.
  - b. Select the desired hardware from the **Hardware** dropdown menu in the **Link Project to Device** dialog box.
  - c. Select the desired device on the hardware from the **Device** dropdown menu in the **Link Project to Device** dialog box.
  - d. Select the correct **Link file type**, depending on the version of software in which your design was compiled:
    - If your design was compiled in the Quartus II software version 12.0 or earlier, select **JDI** as the **Link file type**, verify that the **.jdi** file is correct for your **.sof** file, and click **Ok**.
    - If your design was compiled in the Quartus II software version 12.1 or later, or the Quartus Prime software, select **SOF** as the **Link file type**, verify that the **.sof** file is correct for your programmed device, and click **Ok**.

**Figure 15-2: Link Project to Device Dialog Box**



When you link your project to the device, the toolkit verifies all connections on the device against the information in the JDI or SOF file, as appropriate. If the toolkit detects any mismatch between the JDI file and the device connections, an error message is displayed.

For designs compiled using the Quartus II software version 12.1 or later, or the Quartus Prime software, the SOF file contains a design hash to ensure the SOF file used to program the device matches the SOF file specified for linking to a project. If the hash does not match, an error message appears.

If the toolkit successfully verifies all connections, it then attempts to determine the connection type for each connection. Connections of a known type are listed in the Linked Connections report, and are available for the toolkit to use.

## Establishing Communication to Connections

After you have completed linking the project, you can establish communication to the connections.

1. In the Tasks window,

- Click **Establish Memory Interface Connection** to create a connection to the external memory interface.
- Click **Establish Efficiency Monitor Connection** to create a connection to the efficiency monitor.
- Click **Establish Traffic Generator Connection** to create a connection to the Traffic Generator 2.0 (if enabled, Arria 10 only).

2. To create a communication channel to a connection, select the desired connection from the displayed pulldown menu of connections, and click **Ok**. The toolkit establishes a communication channel to the connection, creates a report folder for the connection, and creates a folder of tasks for the connection.

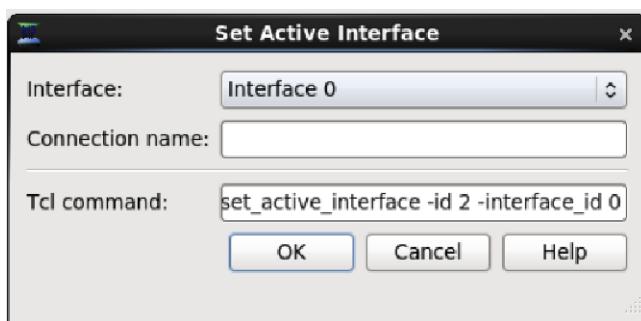
**Note:** By default, the connection and the reports and tasks folders are named according to the hierarchy path of the connection. If you want, you can specify a different name for the connection and its folders.

3. You can run any of the tasks in the folder for the connection; any resulting reports appear in the reports folder for the connection.

## Selecting an Active Interface

If you have more than one external memory interface in an Arria 10 I/O column, you can select one instance as the active interface for debugging.

1. To select one of multiple EMIF instances in an Arria 10 I/O column, use the **Set Active Interface** dialog box.



2. If you want to generate reports for the new active interface, you must first recalibrate the interface.

## Reports

The toolkit can generate a variety of reports, including summary, calibration, and margining reports for external memory interface connections. To generate a supported type of report for a connection, you run the associated task in the tasks folder for that connection.

## Summary Report

The Summary Report provides an overview of the memory interface; it consists of the following tables:

- Summary table. Provides a high-level summary of calibration results. This table lists details about the connection, IP version, IP protocol, and basic calibration results, including calibration failures. This table also lists the estimated average read and write data valid windows, and the calibrated read and write latencies.
- Interface Details table. Provides details about the parameterization of the memory IP. This table allows you to verify that the parameters in use match the actual memory device in use.
- Groups Masked from Calibration table. Lists any groups that were masked from calibration when calibration occurred. Masked groups are ignored during calibration.

**Note:** This table applies only to UniPHY-based interfaces; it is not applicable to Arria 10 EMIF.

- Ranks Masked from Calibration tables (DDR2 and DDR3 only). Lists any ranks that were masked from calibration when calibration occurred. Masked ranks are ignored during calibration.

## Calibration Report (UniPHY)

The Calibration Report provides detailed information about the margins observed before and after calibration, and the settings applied to the memory interface during calibration; it consists of the following tables:

- Per DQS Group Calibration table: Lists calibration results for each group. If a group fails calibration, this table also lists the reason for the failure.

**Note:** If a group fails calibration, the calibration routine skips all remaining groups. You can deactivate this behaviour by running the **Enable Calibration for All Groups On Failure** command in the toolkit.

- DQ Pin Margins Observed Before Calibration table: Lists the DQ pin margins observed before calibration occurs. You can refer to this table to see the per-bit skews resulting from the specific silicon and board that you are using.
- DQS Group Margins Observed During Calibration table: Lists the DQS group margins observed during calibration.
- DQ Pin Settings After Calibration and DQS Group Settings After Calibration table: Lists the settings made to all dynamically controllable parts of the memory interface as a result of calibration. You can refer to this table to see the modifications made by the calibration algorithm.

## Calibration Report (Arria 10 EMIF)

The Calibration Report provides detailed information about the margins observed during calibration, and the settings applied to the memory interface during calibration; it consists of the following tables:

- Calibration Status Per Group table: Lists the pass/fail status per group.
- DQ Pin Margins Observed During Calibration table: Lists the DQ read/write margins and calibrated delay settings. These are the expected margins after calibration, based on calibration data patterns. This table also contains DM/DBI margins, if applicable.
- DQS Pin Margins Observed During Calibration table: Lists the DQS margins observed during calibration.
- FIFO Settings table: Lists the VFIFO and LFIFO settings made during calibration.
- Latency Observed During Calibration table: Lists the calibrated read/write latency.
- Address/Command Margins Observed During Calibration table: Lists the margins on calibrated A/C pins, for protocols that support Address/Command calibration.

## Margin Report

The Margin Report lists the post-calibration margins for each DQ and data mask pin, keeping all other pin settings constant; it consists of the following tables:

- DQ Pin Post Calibration Margins table. Lists the margin data in tabular format.
- Read Data Valid Windows report. Shows read data valid windows in graphical format.
- Write Data Valid Windows report. Shows write data valid windows in graphical format.

**Note:** The Margin Report applies only to UniPHY-based interfaces; it is not applicable to Arria 10 EMIF. For Arria 10 EMIF, the Calibration Report provides equivalent information.

# Operational Considerations

Some features and considerations are of interest in particular situations.

## Specifying a Particular JDI File

Correct operation of the EMIF Toolkit depends on the correct JDI being used when linking the project to the device. The JDI file is produced by the Quartus Prime Assembler, and contains a list of all system level debug nodes and their heirarchy path names. If the default **.jdi** file name is incorrect for your project, you must specify the correct **.jdi** file. The **.jdi** file is supplied during the link-project-to-device step, where the **revision\_name.jdi** file in the project directory is used by default. To supply an alternative **.jdi** file, click on the ellipse then select the correct **.jdi** file.

## PLL Status

When connecting to DDR-based external memory interface connections, the PLL status appears in the Establish Connection dialog box when the IP is generated to use the CSR controller port, allowing you to immediately see whether the PLL status is locked. If the PLL is not locked, no communication can occur until the PLL becomes locked and the memory interface reset is deasserted.

When you are linking your project to a device, an error message will occur if the toolkit detects that a JTAG Avalon-MM master has no clock running. You can run the **Reindex Connections** task to have the toolkit rescan for connections and update the status and type of found connections in the Linked Connections report.

## Margining Reports

The EMIF Toolkit can display margining information showing the post-calibration data-valid windows for reads and writes. Margining information is determined by individually modifying the input and output delay chains for each data and strobe/clock pin to determine the working region. The toolkit can display margining data in both tabular and hierachial formats.

## Group Masks

To aid in debugging your external memory interface, the EMIF Toolkit allows you to mask individual groups and ranks from calibration. Masked groups and ranks are skipped during the calibration process, meaning that only unmasked groups and ranks are included in calibration. Subsequent mask operations overwrite any previous masks.

## Using with Arria 10 Devices in a PHY-Only Configuration

If you want to use the Debug Toolkit with an Arria 10 EMIF IP in a PHY-only configuration, you must connect a debug clock and a reset signal to the Arria 10 external memory interface debug component. Altera recommends using the AFI clock and AFI reset for this purpose.

**Note:** For information about calibration stages in UniPHY-based interfaces, refer to *UniPHY Calibration Stages* in the *Functional Description - UniPHY* chapter. For information about calibration stages in Arria 10 EMIF IP, refer to *Arria 10 EMIF Calibration*, in the *Functional Description - Arria 10 EMIF* chapter.

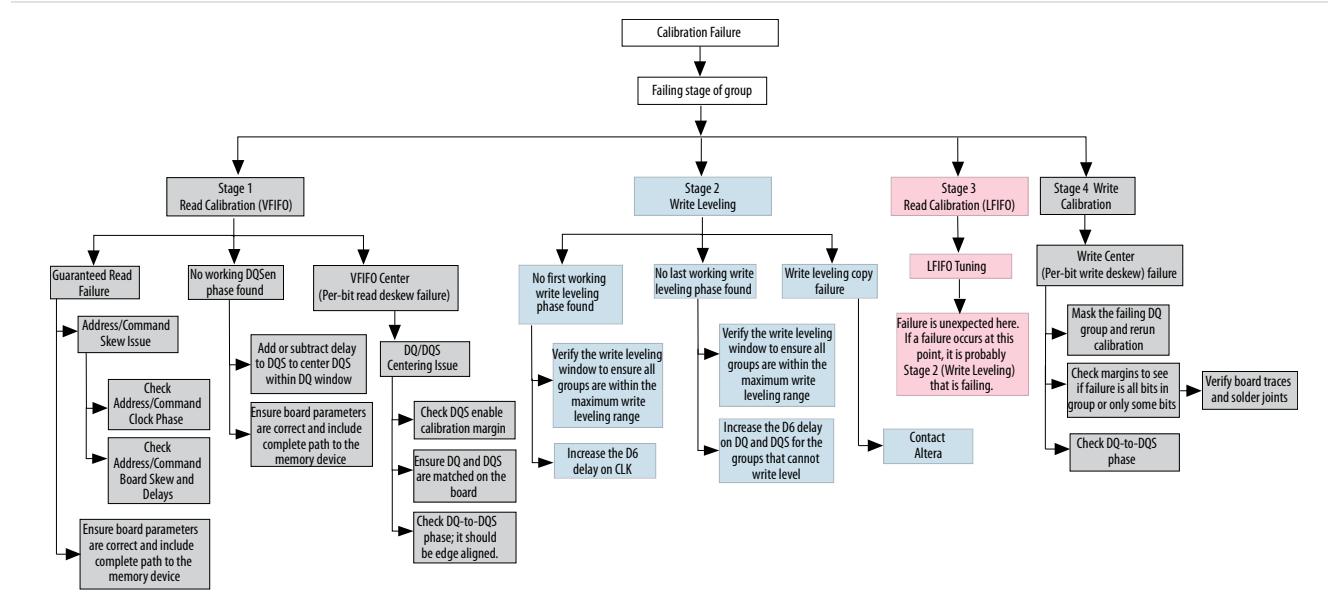
### Related Information

- [Calibration Stages](#) on page 1-52
- [Functional Description—UniPHY](#) on page 1-1
- [Functional Description—Arria 10 EMIF](#) on page 2-1

## Troubleshooting

In the event of calibration failure, refer to the following figure to assist in troubleshooting your design. Calibration results and failing stages are available through the external memory interface toolkit.

**Figure 15-3: Debugging Tips**



## Debug Report for Arria V and Cyclone V SoC Devices

The External Memory Interface Debug Toolkit and EMIF On-Chip Debug Port do not work with Arria V and Cyclone V SoC devices. Debugging information for Arria V and Cyclone V SoC devices is available by enabling a debug output report, which contains similar information.

## Enabling the Debug Report for Arria V and Cyclone V SoC Devices

To enable a debug report for Arria V or Cyclone V SoC devices, perform the following steps:

1. Open the `<design_name>/hps_isw_handoff/sequencerDefines.h` file in a text editor.
2. In the `sequencerDefines.h` file, locate the following line: `#define RUNTIME_CAL_REPORT 0`
3. Change `#define RUNTIME_CAL_REPORT 0` to `#define RUNTIME_CAL_REPORT 1`, and save the file.
4. Generate the board support package (BSP) with semihosting enabled, or with UART output.

The system will now generate the debugging report as part of the calibration process.

## Determining the Failing Calibration Stage for a Cyclone V or Arria V HPS SDRAM Controller

To determine the failing calibration stage, you must turn on the debug output report by setting the `RUNTIME_CAL_REPORT` option to 1 in the `sequencerDefines.h` file, located in the `hps_isw_handoff` directory.

If calibration fails, the following statements are printed in the debug output report:

```
SEQ.C: Calibration Failed
SEQ.C: Error Stage : <Num>
SEQ.C: Error Substage: <Num>
SEQ.C: Error Group : <Num>
```

To determine the stage and sub-stage, open the `sequencer.h` file in the `hps_isw_handoff` directory and look for the calibration defines:

```
/* calibration stages */
#define CAL_STAGE_NIL 0
#define CAL_STAGE_VFIFO 1
#define CAL_STAGE_WLEVEL 2
#define CAL_STAGE_LFIFO 3
#define CAL_STAGE_WRITES 4
#define CAL_STAGE_FULLTEST 5
#define CAL_STAGE_REFRESH 6
#define CAL_STAGE_CAL_SKIPPED 7
#define CAL_STAGE_CAL_ABORTED 8
#define CAL_STAGE_VFIFO_AFTER_WRITES 9
/* calibration substages */
#define CAL_SUBSTAGE_NIL 0
#define CAL_SUBSTAGE_GUARANTEED_READ 1
#define CAL_SUBSTAGE_DQS_EN_PHASE 2
#define CAL_SUBSTAGE_VFIFO_CENTER 3
#define CAL_SUBSTAGE_WORKING_DELAY 1
#define CAL_SUBSTAGE_LAST_WORKING_DELAY 2
#define CAL_SUBSTAGE_WLEVEL_COPY 3
#define CAL_SUBSTAGE_WRITES_CENTER 1
#define CAL_SUBSTAGE_READ_LATENCY 1
#define CAL_SUBSTAGE_REFRESH 1
```

For details about the stages of calibration, refer to *Calibration Stages* in *Functional Description - UniPHY*.

### Related Information

[Calibration Stages](#) on page 1-52

## On-Chip Debug Port for UniPHY-based EMIF IP

The EMIF On-Chip Debug Port allows user logic to access the same calibration data used by the EMIF Toolkit, and allows user logic to send commands to the sequencer. You can use the EMIF On-Chip Debug

Port to access calibration data for your design and to send commands to the sequencer just as the EMIF Toolkit would. The following information is available:

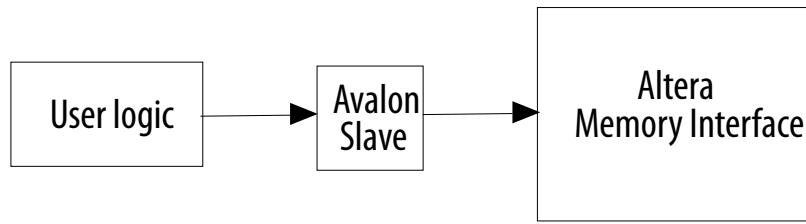
- Pass/fail status for each DQS group
- Read and write data valid windows for each group

In addition, user logic can request the following commands from the sequencer:

- Destructive recalibration of all groups
- Masking of groups and ranks
- Generation of per-DQ pin margining data as part of calibration

The user logic communicates through an Avalon-MM slave interface as shown below.

**Figure 15-4: User Logic Access**



## Access Protocol

The On-Chip Debug Port provides access to calibration data through an Avalon-MM slave interface. To send a command to the sequencer, user logic sends a command code to the command space in sequencer memory. The sequencer polls the command space for new commands after each group completes calibration, and continuously after overall calibration has completed.

The communication protocol to send commands from user logic to the sequencer uses a multistep handshake with a data structure as shown below, and an algorithm as shown in the figure which follows.

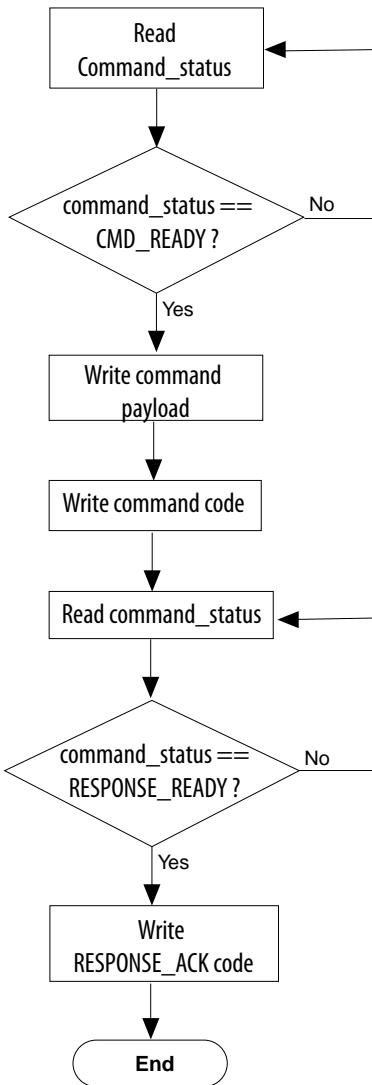
```
typedef struct_debug_data_struct {  
    ...  
    // Command interaction  
    alt_u32 requested_command;  
    alt_u32 command_status;  
    alt_u32 command_parameters[COMMAND_PARAM_WORDS];  
}
```

To send a command to the sequencer, user logic must first poll the `command_status` word for a value of `TCLDBG_TX_STATUS_CMD_READY`, which indicates that the sequencer is ready to accept commands. When the sequencer is ready to accept commands, user logic must write the command parameters into `command_parameters`, and then write the command code into `requested_command`.

The sequencer detects the command code and replaces `command_status` with `TCLDBG_TX_STATUS_CMD_EXE`, to indicate that it is processing the command. When the sequencer has finished running the command, it sets `command_status` to `TCLDBG_TX_STATUS_RESPONSE_READY` to indicate that the result of the command is available to be read. (If the sequencer rejects the requested command as illegal, it sets `command_status` to `TCLDBG_TX_STATUS_ILLEGAL_CMD`.)

User logic acknowledges completion of the command by writing `TCLDBG_CMD_RESPONSE_ACK` to `requested_command`. The sequencer responds by setting `command_status` back to `STATUS_CMD_READY`. (If an illegal command is received, it must be cleared using `CMD_RESPONSE_ACK`.)

Figure 15-5: Debugging Algorithm Flowchart



## Command Codes Reference

The following table lists the supported command codes for the On-Chip Debug Port.

Table 15-1: Supported Command Codes

Command	Parameters	Description
TCLDBG_RUN_MEM_CALIBRATE	None	Runs the calibration routine.
TCLDBG_MARK_ALL_DQS_GROUPS_AS_VALID	None	Marks all groups as valid for calibration.
TCLDBG_MARK_GROUP_AS_SKIP	Group to skip	Mark the specified group to be skipped by calibration.

Command	Parameters	Description
TCLDBG_MARK_ALL_RANKS_AS_VALID	None	Mark all ranks as valid for calibration
TCLDBG_MARK_RANK_AS_SKIP	Rank to skip	Mark the specified rank to be skipped by calibration.
TCLDBG_ENABLE_MARGIN_REPORT	None	Enables generation of the margin report.

## Header Files

The external memory interface IP generates header files which identify the debug data structures and memory locations used with the EMIF On-Chip Debug Port. You should refer to these header files for information required for use with your core user logic. It is highly recommended to use a software component (such as a Nios II processor) to access the calibration debug data.

The header files are unique to your IP parameterization and version, therefore you must ensure that you are referring to the correct version of header for your design. The names of the header files are: **core\_debug.h** and **core\_debugDefines.h**. The header files reside in **<design\_name>/<design\_name>\_s0\_software**.

## Generating IP With the Debug Port

The following steps summarize the procedure for implementing your IP with the EMIF On-Chip Debug Port enabled.

1. Start the Quartus Prime software and generate a new external memory interface. For QDR II and RLDRAM II protocols, ensure that sequencer optimization is set to **Performance** (for Nios II-based sequencer).
2. On the **Diagnostics** tab of the parameter editor, turn on **Enable EMIF On-Chip Debug Port**.
3. Ensure that the **EMIF On-Chip Debug Port interface type** is set to **Avalon-MM Slave**.
4. Click **Finish** to generate your IP.
5. Find the Avalon interface in the top-level generated file. Connect this interface to your debug component.

```
input wire [19:0] seq_debug_addr,      // seq_debug.address
input wire          seq_debug_read_req, // .read
output wire [31:0] seq_debug_rdata,    // .readdata
input  wire          seq_debug_write_req, // .write
input  wire [31:0] seq_debug_wdata,    // .writedata
output wire          seq_debug_waitrequest, // .waitrequest
input  wire [3:0]   seq_debug_be,       // .byteenable
output wire          seq_debug_rdata_valid // .readdatavalid
```

If you are using UniPHY-based IP with the hard memory controller, also connect the `seq_debug_clk` and `seq_debug_reset_in` signals to clock and asynchronous reset signals that control your debug logic.

6. Find the `core_debug.h` and `core_debugDefines.h` header files in `<design_name>/<design_name>-s0_software` and include these files in your debug component code.
7. Write your debug component using the supported command codes, to read and write to the Avalon-MM interface.

The debug data structure resides at the memory address `SEQ_CORE_DEBUG_BASE`, which is defined in the `core_debugDefines.h` header file.

## Example C Code for Accessing Debug Data

A typical use of the EMIF On-Chip Debug Port might be to recalibrate the external memory interface, and then access the reports directly using the `summary_report_ptr`, `cal_report_ptr`, and `margin_report_ptr` pointers, which are part of the debug data structure.

The following code sample illustrates:

```
/*
 * DDR3 UniPHY sequencer core access example
 */

#include <stdio.h>
#include <unistd.h>
#include <io.h>
#include "core_debugDefines.h"

int send_command(volatile debug_data_t* debug_data_ptr, int command, int args[], int num_args)
{
    volatile int i, response;

    // Wait until command_status is ready
    do {
        response = IORD_32DIRECT(&(debug_data_ptr->command_status), 0);
    } while(response != TCLDBG_TX_STATUS_CMD_READY);

    // Load arguments
    if(num_args > COMMAND_PARAM_WORDS)
    {
        // Too many arguments
        return 0;
    }
    for(i = 0; i < num_args; i++)
    {
        IOWR_32DIRECT(&(debug_data_ptr->command_parameters[i]), 0, args[i]);
    }
    // Send command code
    IOWR_32DIRECT(&(debug_data_ptr->requested_command), 0, command);
    // Wait for acknowledgment
    do {
        response = IORD_32DIRECT(&(debug_data_ptr->command_status), 0);
    } while(response != TCLDBG_TX_STATUS_RESPONSE_READY && response != TCLDBG_TX_STATUS_ILLEGAL_CMD);
    // Acknowledge response
    IOWR_32DIRECT(&(debug_data_ptr->requested_command), 0, TCLDBG_CMD_RESPONSE_ACK);
    // Return 1 on success, 0 on illegal command
    return (response != TCLDBG_TX_STATUS_ILLEGAL_CMD);
}
int main()
{
    volatile debug_data_t* my_debug_data_ptr;
```

```
volatile debug_summary_report_t* my_summary_report_ptr;
volatile debug_cal_report_t* my_cal_report_ptr;
volatile debug_margin_report_t* my_margin_report_ptr;
volatile debug_cal_observed_dq_margins_t* cal_observed_dq_margins_ptr;
int i, j, size;
int args[COMMAND_PARAM_WORDS];
// Initialize pointers to the debug reports
my_debug_data_ptr = (debug_data_t*)SEQ_CORE_DEBUG_BASE;
my_summary_report_ptr = (debug_summary_report_t*)
(IORD_32DIRECT(&(my_debug_data_ptr->summary_report_ptr), 0));
my_cal_report_ptr = (debug_cal_report_t*)(IORD_32DIRECT(&(my_debug_data_ptr-
>cal_report_ptr), 0));
my_margin_report_ptr = (debug_margin_report_t*)(IORD_32DIRECT(&(my_debug_data_ptr-
>margin_report_ptr), 0));

// Activate all groups and ranks
send_command(my_debug_data_ptr, TCLDBG_MARK_ALL_DQS_GROUPS_AS_VALID, 0, 0);
send_command(my_debug_data_ptr, TCLDBG_MARK_ALL_RANKS_AS_VALID, 0, 0);
send_command(my_debug_data_ptr, TCLDBG_ENABLE_MARGIN_REPORT, 0, 0);

// Mask group 4
args[0] = 4;
send_command(my_debug_data_ptr, TCLDBG_MARK_GROUP_AS_SKIP, args, 1);
send_command(my_debug_data_ptr, TCLDBG_RUN_MEM_CALIBRATE, 0, 0);

// SUMMARY
printf("SUMMARY REPORT\n");
printf("mem_address_width: %u\n", IORD_32DIRECT(&(my_summary_report_ptr-
>mem_address_width), 0));
printf("mem_bank_width: %u\n", IORD_32DIRECT(&(my_summary_report_ptr-
>mem_bank_width), 0));
// etc...

// CAL REPORT
printf("CALIBRATION REPORT\n");
// DQ read margins
for(i = 0; i < RW_MGR_MEM_DATA_WIDTH; i++)
{
    cal_observed_dq_margins_ptr = &(my_cal_report_ptr->cal_dq_in_margins[i]);
    printf("0x%lx DQ %d Read Margin (taps): -%d : %d\n", (unsigned
    int)cal_observed_dq_margins_ptr, i,
    IORD_32DIRECT(&(cal_observed_dq_margins_ptr->left_edge), 0),
    IORD_32DIRECT(&(cal_observed_dq_margins_ptr->right_edge), 0));
}
// etc...
return 0;
}
```

## On-Chip Debug Port for Arria 10 EMIF IP

The EMIF On-Chip Debug Port allows user logic to access the same calibration data used by the EMIF Toolkit, and allows user logic to send commands to the sequencer. You can use the EMIF On-Chip Debug Port to access calibration data for your design and to send commands to the sequencer just as the EMIF Toolkit would. The following information is available:

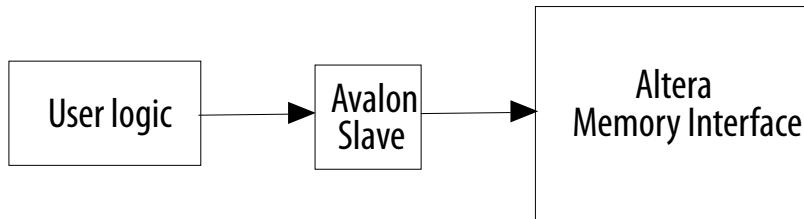
- Pass/fail status for each DQS group
- Read and write data valid windows for each group

In addition, user logic can request the following commands from the sequencer:

- Destructive recalibration of all groups
- Masking of groups and ranks
- Generation of per-DQ pin margining data as part of calibration

The user logic communicates through an Avalon-MM slave interface as shown below.

**Figure 15-6: User Logic Access**



## Access Protocol

The On-Chip Debug Port provides access to calibration data through an Avalon-MM slave interface. To send a command to the sequencer, user logic sends a command code to the command space in sequencer memory. The sequencer polls the command space for new commands after each group completes calibration, and continuously after overall calibration has completed.

The communication protocol to send commands from user logic to the sequencer uses a multistep handshake with a data structure as shown below, and an algorithm as shown in the figure which follows.

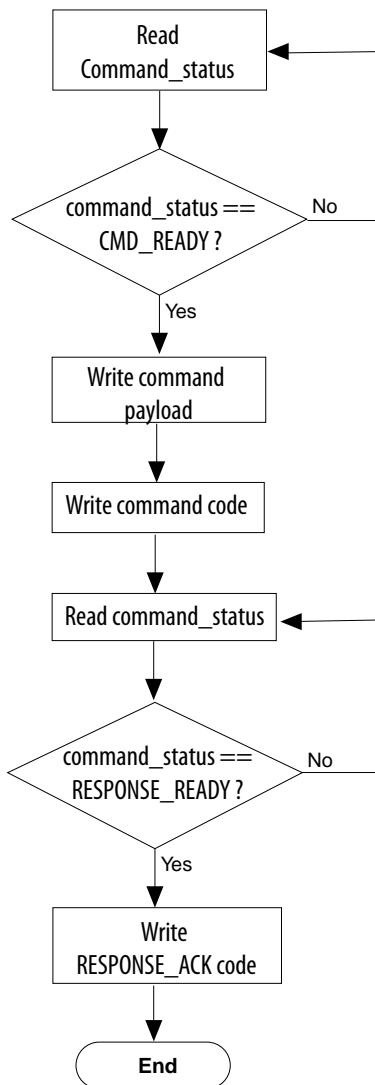
```

typedef struct_debug_data_struct {
    ...
    // Command interaction
    alt_u32 requested_command;
    alt_u32 command_status;
    alt_u32 command_parameters[COMMAND_PARAM_WORDS];
}
  
```

To send a command to the sequencer, user logic must first poll the `command_status` word for a value of `TCLDBG_TX_STATUS_CMD_READY`, which indicates that the sequencer is ready to accept commands. When the sequencer is ready to accept commands, user logic must write the command parameters into `command_parameters`, and then write the command code into `requested_command`.

The sequencer detects the command code and replaces `command_status` with `TCLDBG_TX_STATUS_CMD_EXE`, to indicate that it is processing the command. When the sequencer has finished running the command, it sets `command_status` to `TCLDBG_TX_STATUS_RESPONSE_READY` to indicate that the result of the command is available to be read. (If the sequencer rejects the requested command as illegal, it sets `command_status` to `TCLDBG_TX_STATUS_ILLEGAL_CMD`.)

User logic acknowledges completion of the command by writing `TCLDBG_CMD_RESPONSE_ACK` to `requested_command`. The sequencer responds by setting `command_status` back to `STATUS_CMD_READY`. (If an illegal command is received, it must be cleared using `CMD_RESPONSE_ACK`.)

**Figure 15-7: Debugging Algorithm Flowchart**

## EMIF On-Chip Debug Port

In Arria 10 and later families, access to on-chip debug is provided through software running on a Nios processor connected to the external memory interface.

If you enable the Use Soft Nios Processor for On-Chip Debug option, the system instantiates a soft Nios processor, and software files are provided as part of the EMIF IP.

Instructions on how to use the software are available in the following file: : <*variation\_name*>/altera\_emif\_arch\_nf\_<version number>/<synth|sim>/<*variation\_name*>\_altera\_emif\_arch\_nf\_<version number>\_<unique ID>\_readme.txt.

## On-Die Termination Calibration

The Calibrate Termination feature lets you determine the optimal **On-Die Termination** and **Output Drive Strength** settings for your memory interface, for Arria 10 and later families.

The **Calibrate Termination** function runs calibration with all available termination settings and selects the optimal settings based on the calibration margins.

The **Calibrate Termination** feature is available for DDR3, DDR4, and RLDRAM 3 protocols, on Arria 10 devices.

## Eye Diagram

The **Generate Eye Diagram** feature allows you to create read and write eye diagrams for each pin in your memory interface, for Arria 10 and later families.

The **Generate Eye Diagram** feature uses calibration data patterns to determine margins at each  $V_{ref}$  setting on both the FPGA pins and the memory device pins. A full calibration is done for each  $V_{ref}$  setting. Other settings, such as DQ delay chains, will change for each calibration. At the end of a **Generate Eye Diagram** command, a default calibration is run to restore original behavior.

The **Generate Eye Diagram** feature is available for DDR4 and QDR-IV protocols, on Arria 10 devices.

## Driver Margining for Arria 10 EMIF IP

The Driver Margining feature lets you measure margins on your memory interface using a driver with arbitrary traffic patterns.

Margins measured with this feature may differ from margins measured during calibration, because of different traffic patterns. Driver margining is not available if ECC is enabled.

To use driver margining, ensure that the following signals on the driver are connected to In-System Sources/Probes:

- Reset\_n: An active low reset signal
- Pass: A signal which indicates that the driver test has completed successfully. No further memory transactions must be sent after this signal is asserted.
- Fail: A signal which indicates that the driver test has failed. No further memory transactions must be sent after this signal is asserted.
- PNF (Pass Not Fail): An array of signals that indicate the pass/fail status of individual bits of a data burst. The PNF should be arranged such that each bit index corresponds to  $(\text{Bit of burst} * \text{DQ width}) + (\text{DQ pin})$ . A 1 indicates pass, 0 indicates fail. If the PNF width exceeds the capacity of one In-System Probe, specify them in `PNF[1]` and `PNF[2]`; otherwise, leave them blank.

If you are using the example design for EMIF, the In-System Sources/Probes can be enabled by adding the following line to your .qsf file:

```
set_global_assignment -name VERILOG_MACRO "ALTERA_EMIF_ENABLE_ISSP=1"
```

## Determining Margin

The Driver Margining feature lets you measure margins on your Arria 10 EMIF IP interface using a driver with arbitrary traffic patterns.

The Driver Margining feature is available only for DDR3 and DDR4 interfaces on Arria 10 devices, when ECC is not enabled.

1. Establish a connection to the desired interface and ensure that it has calibrated successfully.
2. Select **Driver Margining** from the **Commands** folder under the target interface connection.
3. Select the appropriate **In-System Sources/Probes** using the drop-down menus.
4. If required, set additional options in the **Advanced Options** section:

- Specify **Traffic Generator 2.0** to allow margining on a per-rank basis. Otherwise, margining is performed on all ranks together.
  - **Step size** specifies the granularity of the driver margining process. Larger step sizes allow faster margining but reduced accuracy. It is recommended to omit this setting.
  - **Adjust delays after margining** causes delay settings to be adjusted to the center of the window based on driver margining results.
  - The **Margin Read**, **Write**, **Write DM**, and **DBI** checkboxes allow you to control which settings are tested during driver margining. You can uncheck boxes to allow driver margining to complete more quickly.
5. Click **OK** to run the tests.  
The toolkit measures margins for **DQ** read/write and **DM**. The process may take several minutes, depending on the margin size and the duration of the driver tests. The test results are available in the *Margin Report*.

## Read Setting and Apply Setting Commands for Arria 10 EMIF IP

The **Read Setting** command allows you to read calibration settings directly from the EMIF PHY. The **Apply Setting** command allows you to write calibration settings, to override existing settings for testing purposes.

### Reading or Applying Calibration Settings

The **Read Setting** and **Apply Setting** commands let you read and write calibration settings directly.

The **Read Setting** and **Apply Setting** commands are available only for DDR3 and DDR4 interfaces on Arria 10 devices.

1. Establish a connection to the desired interface.
2. Select **Read Setting** or **Apply Setting** from the **Settings** folder under the target interface connection.
3. Select the desired setting type.
4. Select the desired rank shadow register to modify. (Leave this field blank if it is not applicable).
5. Select the index of the pin or group to modify.
6. (For the **Apply Setting** command) Enter the new value to apply to the desired location.
7. Click **OK**.

The setting is read (or applied) using a read/write at the address indicated in the Tcl command window. You can perform similar transactions using the On-Chip Debug Port.

## Configuring the Traffic Generator 2.0

If you have an example design with the Traffic Generator 2.0 enabled, you can configure the traffic pattern using the EMIF Debug Toolkit.

1. Establish a connection to the Traffic Generator by clicking **Create Traffic Generator Connection**.
2. Click **Run Custom Traffic Pattern**.
3. On the **Data** tab, you can set the data to be written on each pin.

- You can configure each pin individually, or you can use the **All Pins** option to configure all pins simultaneously.
  - All pins can transmit PRBS data, or each pin can transmit its own fixed pattern.
  - The **Seed/Fixed Pattern** has a size in bits equal to the pattern length specified during IP generation (8, 16, or 32). The **Seed/Fixed Pattern** can be specified in decimal or in hexadecimal, using a 0x prefix.
  - If you enable **Test data mask**, the data mask pins are tested by first writing a fixed data pattern to memory, and then running the traffic generator using the specified data mask pattern.
4. On the **Address** tab, you can customize the addresses used by the traffic generator.
- The **Address Mode** specifies the pattern for generating addresses:
    - In **Sequential** address mode each address is incremented from the previous by the **Sequential address increment**.
    - In **Random** address mode, each address is generated randomly.
    - In **Random Sequential** address mode addresses are randomly generated, then incremented sequentially a number of times equal to the **Number of sequential addresses** setting.
  - **Start address** specifies the starting address used in **Sequential** mode.
  - **Number of sequential addresses** specifies the number of sequential addresses to generate before generating a new random address in **Random sequential** mode.
  - **Sequential address increment** specifies the step size between consecutive addresses in **Sequential** and **Random sequential** modes.
  - **Return to start address** causes the address generator to return to the start address after each loop in **Sequential** address mode.
  - **Mask mode** specifies restrictions on various address components:
    - **Disabled** deactivates masking.
    - **Fixed** restricts the specified address component to the specified value.
    - **Full Cycling** causes the specified address component to be incremented on each new address.
    - **Partial Cycling** causes bank addresses to be cycled for maximum efficiency.

5. On the **Loops** tab, you can set parameters related to the test duration.

- The **Loops** parameter specifies the number of test iterations. Any random address or data generators are not reset between loops.
- The **Writes/Reads per block** parameter specifies the number of consecutive write/read operations to issue. Typically, the number of writes and reads should be identical in order to verify each write with a corresponding read.
- The **Write/Read repeats** parameter specifies the number of times to repeat each write or read operation.
- The **Avalon burst length** parameter specifies the burst length of each Avalon transaction.

6. Click **Ok** to configure the traffic generator and start the traffic pattern.

7. Click **Tasks > Generate All Reports** to view the results of the test.

#### Related Information

- [EMIF Configurable Traffic Generator 2.0 Reference](#)
- [Configurable Traffic Generator 2.0 Parameters](#)
- [Configurable Traffic Generator 2.0 Configuration Options](#)

- [Performing Your Own Tests Using Traffic Generator 2.0](#)

## The Traffic Generator 2.0 Report

The traffic generator report provides information about the configuration of the Traffic Generator 2.0 and the result of the most recent run of traffic.

### Understanding the Traffic Generator 2.0 Report

The traffic generator report contains the following information:

- A **Pass flag** value of 1 indicates that the run completed with no errors.
- A **Fail flag** value of 1 indicates that the run encountered one or more errors.
- The **Failure Count** indicates the number of read transactions where data did not match the expected value.
- The **First failure** address indicates the address corresponding to the first data mismatch.
- The **Version** indicates the version number of the traffic generator.
- The **Number of data generators** indicates the number of data pins at the memory interface.
- The **Number of byte enable generators** indicates the number of byte enable and data mask pins at the memory interface.
- The **Rank Address**, **Bank address**, and **Bank group width** values indicate the number of bits in the Avalon address corresponding to each of those components.
- The **Data/Byte enable pattern length** indicates the number of bits in the fixed pattern used on each data/byte enable pin.
- The **PNF** (pass not fail) value indicates the persistent pass/fail status for each bit in the Avalon data. It is also presented on a per-memory-pin basis for each beat within a memory burst.
- **Fail Expected Data** is the data that was expected on the first failing transaction (if applicable).
- **Fail Read Data** is the data that was received on the first failing transaction (if applicable).

## Example Tcl Script for Running the EMIF Debug Toolkit

If you want, you can run the EMIF Debug Toolkit using a Tcl script. The following example Tcl script is applicable to all device families.

The following example Tcl script opens a file, runs the debug toolkit, and writes the resulting calibration reports to a file.

You should adjust the variables in the script to match your design. You can then run the script using the command `quartus_sh -t example.tcl`.

```
# Modify the following variables for your project
set project "ed_synth.qpf"
# Index of the programming cable. Can be listed using "get_hardware_names"
set hardware_index 1
# Index of the device on the specified cable. Can be listed using "get_device_names"
set device_index 1
# SOF file containing the EMIF to debug
set sof "ed_synth.sof"
# Connection ID of the EMIF debug interface. Can be listed using "get_connections"
set connection_id 2
# Output file
set report "toolkit.rpt"

# The following code opens a project and writes its calibration reports to a file.
project_open $project
load_package ::quartus::external_memif_toolkit
```

```

initialize_connections
set hardware_name [lindex [get_hardware_names] $hardware_index]
set device_name [lindex [get_device_names -hardware_name $hardware_name]
$device_index]
link_project_to_device -device_name $device_name -hardware_name $hardware_name -
sof_file $sof
establish_connection -id $connection_id
create_connection_report -id $connection_id -report_type summary
create_connection_report -id $connection_id -report_type calib
write_connection_target_report -id $connection_id -file $report

```

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	<ul style="list-style-type: none"> <li>Added additional option to step 1 of <i>Establishing Communication to Connections</i>.</li> <li>Added sentence to second bullet in <i>Eye Diagram</i>.</li> <li>Expanded step 4 and added step 5, in <i>Determining Margin</i>.</li> <li>Added <i>Configuring the Traffic Generator 2.0</i> and <i>The Traffic Generator 2.0 Report</i>.</li> </ul>
November 2015	2015.11.02	<ul style="list-style-type: none"> <li>Changed title of <i>Architecture</i> section to <i>User Interface</i>.</li> <li>Added sentence to <i>Driver Margining</i> section stating that driver margining is not available if ECC is enabled.</li> <li>Removed note that the memory map for Arria 10 On-Chip Debug would be available in a future release.</li> <li>Created separate On-Chip Debug sections for UniPHY-based EMIF IP and Arria 10 EMIF IP.</li> <li>Changed title of <i>Driver Margining (Arria 10 only)</i> section to <i>Driver Margining for Arria 10 EMIF IP</i>.</li> <li>Changed title of <i>Read Setting and Apply Setting Commands (Arria 10 only)</i> to <i>Read Setting and Apply Setting Commands for Arria 10 EMIF IP</i>.</li> <li>Added section <i>Example Tcl Script for Running the EMIF Debug Toolkit</i>.</li> <li>Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.</li> </ul>
May 2015	2015.05.04	<ul style="list-style-type: none"> <li>Added <i>Determining the Failing Calibration Stage for a Cyclone V or Arria V HPS SDRAM Controller</i>.</li> <li>Changed occurrences of <i>On-Chip Debug Toolkit</i> to <i>On-Chip Debug Port</i>.</li> <li>Added <i>Driver Margining (Arria 10 only)</i> and <i>Determining Margin</i>.</li> <li>Added <i>Read Setting and Apply Setting Commands (Arria 10 only)</i> and <i>Reading or Applying Calibration Settings</i>.</li> </ul>

Date	Version	Changes
December 2014	2014.12.15	<ul style="list-style-type: none"><li>Added paragraph to step 5 of <i>Generating IP With the Debug Port</i>.</li><li>Added mention of <code>seq_debug_clk</code> and <code>seq_debug_reset_in</code> to step 5 of <i>Generating IP With the Debug Port</i>.</li></ul>
August 2014	2014.08.15	Maintenance release.
December 2013	2013.12.16	Maintenance release.
November 2012	2.2	<ul style="list-style-type: none"><li>Changes to <i>Setup and Use</i> and <i>General Workflow</i> sections.</li><li>Added <i>EMIF On-Chip Debug Toolkit</i> section</li><li>Changed chapter number from 11 to 13.</li></ul>
August 2012	2.1	Added table of debugging tips.
June 2012	2.0	<ul style="list-style-type: none"><li>Revised content for new UniPHY EMIF Toolkit.</li><li>Added Feedback icon.</li></ul>
November 2011	1.0	Harvested 11.0 DDR2 and DDR3 SDRAM Controller with UniPHY EMIF Toolkit content.

# Upgrading to UniPHY-based Controllers from ALTMEMPHY-based Controllers **16**

2016.05.02

EMI\_RM



Subscribe



Send Feedback

The following topics describe the process of upgrading to UniPHY from DDR2 or DDR3 SDRAM High-Performance Controller II with ALTMEMPHY Designs.

**NOTE:** Designs that do not use the AFI cannot be upgraded to UniPHY. If your design uses non-AFI IP cores, Altera recommends that you start a new design with the UniPHY IP core. In addition, Altera recommends that any new designs targeting Stratix III, Stratix IV, or Stratix V use the UniPHY datapath.

To upgrade your ALTMEMPHY-based DDR2 or DDR3 SDRAM High-Performance Controller II design to a DDR2 or DDR3 SDRAM controller with UniPHY IP core, you must complete the tasks listed below:

1. Generating Equivalent Design
2. Replacing the ALTMEMPHY Datapath with UniPHY Datapath
3. Resolving Port Name Differences
4. Creating OCT Signals
5. Running Pin Assignments Script
6. Removing Obsolete Files
7. Simulating your Design

The following topics describe these tasks in detail.

## Related Information

- [Generating Equivalent Design](#) on page 16-1
- [Replacing the ALTMEMPHY Datapath with UniPHY Datapath](#) on page 16-2
- [Resolving Port Name Differences](#) on page 16-2
- [Creating OCT Signals](#) on page 16-4
- [Running Pin Assignments Script](#) on page 16-4
- [Removing Obsolete Files](#) on page 16-4
- [Simulating your Design](#) on page 16-4
- [Creating OCT Signals](#) on page 16-4

## Generating Equivalent Design

Create a new DDR2 or DDR3 SDRAM controller with UniPHY IP core, by following the steps in *Implementing and Parameterizing Memory IP* and apply the following guidelines:

© 2016 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO  
9001:2008  
Registered

- Specify the same variation name as the ALTMEMPHY variation.
- Specify a directory different than the ALTMEMPHY design directory to prevent files from overwriting each other during generation.

To ease the migration process, ensure the UniPHY-based design you create is as similar as possible to the existing ALTMEMPHY-based design. In particular, you should ensure the following settings are the same in your UniPHY-based design:

- **PHY settings** tab
- FPGA speed grade
- PLL reference clock
- Memory clock frequency
- There is no need to change the default **Address and command clock phase settings**; however, if you have board skew effects in your ALTMEMPHY design, enter the difference between that clock phase and the default clock phase into the **Address and command clock phase settings**.
- **Memory Parameters** tab—all parameters must match.
- **Memory Timing** tab—all parameters must match.
- **Board settings** tab—all parameters must match.
- **Controller settings** tab—all parameters must match

**Note:** In ALTMEMPHY-based designs you can turn off dynamic OCT. However, all UniPHY-based designs use dynamic parallel OCT and you cannot turn it off.

#### Related Information

[Implementing and Parameterizing Memory IP](#)

## Replacing the ALTMEMPHY Datapath with UniPHY Datapath

To replace the ALTMEMPHY datapath with the UniPHY datapath, follow these steps:

1. In the Quartus Prime software, open the Assignment Editor, on the Assignments menu click **Assignment Editor**.
2. Manually delete all of the assignments related to the external memory interface pins, except for the location assignments if you are preserving the pinout. By default, these pin names start with the `mem` prefix, though in your design they may have a different name.
3. Remove the old ALTMEMPHY **.qip** file from the project, as follows:
  - a. On the Assignments menu click **Settings**.
  - b. Specify the old **.qip**, and click **Remove**.

Your design now uses the UniPHY datapath.

## Resolving Port Name Differences

Several port names in the ALTMEMPHY datapath are different than in the UniPHY datapath. The different names may cause compilation errors.

This topic describes the changes you must make in the RTL for the entity that instantiates the memory IP core. Each change applies to a specific port in the ALTMEMPHY datapath. Unconnected ports require no changes.

In some instances, multiple ports in ALTMEMPHY-based designs are mapped to a single port in UniPHY-based designs. If you use both ports in ALTMEMPHY-based designs, assign a temporary signal

to the common port and connect it to the original wires. The following table shows the changes you must make.

**Table 16-1: Changes to ALTMEMPHY Port Names**

ALTMEMPHY Port	Changes
aux_full_rate_clk	The UniPHY-based design does not generate this signal. You can generate it if you require it.
aux_scan_clk	The UniPHY-based design does not generate this signal. You can generate it if you require it.
aux_scan_clk_reset_n	The UniPHY-based design does not generate this signal. You can generate it if you require it.
dll_reference_clk	The UniPHY-based design does not generate this signal. You can generate it if you require it.
dqs_delay_ctrl_export	This signal is for DLL sharing between ALTMEMPHY instances and is not applicable for UniPHY-based designs.
local_address	Rename to avl_addr.
local_be	Rename to avl_be.
local_burstbegin	Rename to avl_burstbegin.
local_rdata	Rename to avl_rdata.
local_rdata_valid	Rename to avl_rdata_valid.
local_read_req	Rename to avl_read_req.
local_ready	Rename to avl_ready.
local_size	Rename to avl_size.
local_wdata	Rename to avl_wdata.
local_write_req	Rename to avl_write_req.
mem_addr	Rename to mem_a.
mem_clk	Rename to mem_ck.
mem_clk_n	Rename to mem_ck_n.
mem_dqsn	Rename to mem_dqs_n.

ALTMEMPHY Port	Changes
oct_ctl_rs_value	Remove from design (see “Creating OCT Signals”).
oct_ctl_rt_value	Remove from design (see “Creating OCT Signals”).
phy_clk	Rename to afi_clk.
reset_phy_clk_n	Rename to afi_reset_n.
local_refresh_ack reset_request_n	The controller no longer exposes these signals to the top-level design, so comment out these outputs. If you need it, bring the wire out from the high-performance II controller entity in <b>&lt;project_directory&gt;/&lt;variation name&gt;.v</b> .

**Related Information**

[Creating OCT Signals](#) on page 16-4

## Creating OCT Signals

In ALTMEMPHY-based designs, the Quartus Prime Fitter creates the `alt_oct` block outside the IP core and connects it to the `oct_ctl_rs_value` and `oct_ctl_rt_value` signals.

In UniPHY-based designs, the OCT block is part of the IP core, so the design no longer requires these two ports. Instead, the UniPHY-based design requires two additional ports, `oct_rup` and `oct_rdn` (for Stratix III and Stratix IV devices), or `oct_rzqin` (for Stratix V devices). You must create these ports in the instantiating entity as input pins and connect to the UniPHY instance. Then route these pins to the top-level design and connect to the OCT R<sub>UP</sub> and R<sub>DOWN</sub> resistors on the board.

For information on OCT control block sharing, refer to “The OCT Sharing Interface” in this volume.

## Running Pin Assignments Script

Remap your design by running analysis and synthesis.

When analysis and synthesis completes, run the pin assignments Tcl script and then verify the new pin assignments in the Assignment Editor.

## Removing Obsolete Files

After you upgrade the design, you may remove the unnecessary ALTMEMPHY design files from your project.

To identify these files, examine the original ALTMEMPHY-generated `.qip` file in any text editor.

## Simulating your Design

You must use the UniPHY memory model to simulate your new design.

To use the UniPHY memory model, follow these steps:

1. Edit your instantiation of the UniPHY datapath to ensure the local\_init\_done, local\_cal\_success, local\_cal\_fail, soft\_reset\_n, oct\_rdn, oct\_rup, reset\_phy\_clk\_n, and phy\_clk signals are at the top-level entity so that an instantiating testbench can refer to those signals.
2. To use the UniPHY testbench and memory model, generate the example design when generating your IP instantiation.
3. Specify that your third-party simulator should use the UniPHY testbench and memory model instead of the ALTMEMPHY memory model, as follows:
  - a. On the Assignments menu, point to **Settings** and click the **Project Settings** window.
  - b. Select the **Simulation** tab, click **Test Benches**, click **Edit**, and replace the ALTMEMPHY testbench files with the following files:
    - \<project directory>\<variation name>\_example\_design\simulation\verilog\submodules\altera\_avalon\_clock\_source.sv or \<project directory>\<variation name>\_example\_design\simulation\vhdl\submodules\altera\_avalon\_clock\_source.vhd
    - \<project directory>\<variation name>\_example\_design\simulation\verilog\submodules\altera\_avalon\_reset\_source.sv or \<project directory>\<variation name>\_example\_design\simulation\vhdl\submodules\altera\_avalon\_reset\_source.vhd
    - \<project directory>\<variation name>\_example\_design\simulation\verilog\<variation name>\_example\_sim.v or \uniphy\<variation name>\_example\_design\simulation\vhdl\<variation name>\_example\_sim.vhd
    - \<project directory>\<variation name>\_example\_design\simulation\verilog\submodules\verbosity\_pkg.sv
    - \<project directory>\<variation name>\_example\_design\simulation\verilog\submodules\status\_checker\_no\_ifdef\_params.sv or \<project directory>\<variation name>\_example\_design\simulation\vhdl\submodules\status\_checker\_no\_ifdef\_params.vhd
    - \<project directory>\<variation name>\_example\_design\simulation\verilog\submodules\alt\_mem\_if\_common\_ddr\_mem\_model\_ddr3\_mem\_if\_dm\_pins\_en\_mem\_if\_dqsn\_en.sv or \<project directory>\<variation name>\_example\_design\simulation\vhdl\submodules\alt\_mem\_if\_common\_ddr\_mem\_model\_ddr3\_mem\_if\_dm\_pins\_en\_mem\_if\_dqsn\_en.vhd
    - \<project directory>\<variation name>\_example\_design\simulation\verilog\submodules\alt\_mem\_if\_ddr3\_mem\_model\_top\_ddr3\_mem\_if\_dm\_pins\_en\_mem\_if\_dqsn\_en.sv or \<project directory>\<variation name>\_example\_design\simulation\vhdl\submodules\alt\_mem\_if\_ddr3\_mem\_model\_top\_ddr3\_mem\_if\_dm\_pins\_en\_mem\_if\_dqsn\_en.vhd
4. Open the <variation name>\_example\_sim.v file and find the UniPHY-generated simulation example design module name below: <variation name>\_example\_sim\_e0.
5. Change the module name above to the name of your top-level design module.
6. Refer to the following table and update the listed port names of the example design in the UniPHY-generated <variation name>\_example\_sim.v file.

Table 16-2: Example Design Port Names

Example Design Name	New Name
pll_ref_clk	Rename to clock_source.
mem_a	Rename to mem_addr.
mem_ck	Rename to mem_clk.

Example Design Name	New Name
mem_ck_n	Rename to mem_clk_n.
mem_dqs_n	Rename to mem_dqsn.
drv_status_pass	Rename to pnf.
afi_clk	Rename to phy_clk.
afi_reset_n	Rename to reset_phy_clk_n.
drv_status_fail	This signal is not available, so comment out this output.
afi_half_clk	This signal is not exposed to the top-level design, so comment out this output.

For more information about generating example simulation files, refer to Simulating Memory IP, in volume 2 of the *External Memory Interface Handbook*.

## Document Revision History

Date	Version	Changes
May 2016	2016.05.02	Maintenance release.
November 2015	2015.11.02	Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i> .
May 2015	2015.05.04	Maintenance release.
December 2014	2014.12.15	Maintenance release.
August 2014	2014.08.15	Maintenance release.
December 2013	2013.12.16	Removed local_wdata_req from port names table.
November 2012	2.3	Changed chapter number from 12 to 14.
June 2012	2.2	Added Feedback icon.
November 2011	2.1	Revised <i>Simulating your Design</i> section.