



# **External Memory PHY Interface Megafunction User Guide (ALTMEMPHY)**

---



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

UG-01014-7.1

Software Version: 9.0 SP1  
Document Version: 7.1  
Document Date: April 2009

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. 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 Corporation. 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.



## Chapter 1. About this Megafunction

Device Family Support .....	1-1
Glossary .....	1-3
Introduction .....	1-4
Features .....	1-6
What's New in Version 9.0 and Version 9.0SP1 .....	1-6
Support and Performance .....	1-7
Resource Utilization .....	1-8
Latency .....	1-11
Legacy Integrated Static Datapath and Controller .....	1-11
High-Performance Controllers .....	1-12

## Chapter 2. Getting Started

System Requirements .....	2-1
Using the MegaWizard Plug-In Manager .....	2-1
MegaWizard Plug-In Manager Page Descriptions .....	2-2
ALTMEMPHY Parameter Settings Page .....	2-3
Memory Settings .....	2-4
PHY Settings .....	2-20
Controller Interface Settings .....	2-24
ALTMEMPHY EDA Page .....	2-25
ALTMEMPHY Summary Page .....	2-26
Generated Files .....	2-28
Instantiating Multiple ALTMEMPHY Instances .....	2-31
Clock Sharing .....	2-31
Stratix III and Stratix IV Devices .....	2-36
Creating an Emulated x36 QDRII+/QDRII SRAM ALTMEMPHY Variation .....	2-37
Inferring Megafunctions from HDL Code .....	2-40
Compiling in the Quartus II Software .....	2-40
Identifying a Megafunction After Compilation .....	2-42
Analyzing Timing .....	2-42
Timing Constraints .....	2-43
Timing Analysis Using the TimeQuest Timing Analyzer .....	2-44
Optimizing Timing .....	2-44
Analyzed Timing Paths .....	2-46
Timing Closure .....	2-50
Simulating your Design .....	2-51
DDR2/DDR SDRAM, DDR3 SDRAM (Component), and QDRII+/QDRII SRAM Simulation ...	2-52
DDR3 SDRAM (DIMM) Simulation .....	2-52
Simulation Files .....	2-54

## Chapter 3. Specifications

Introduction .....	3-1
Calibration .....	3-1

DDR2/DDR SDRAM and DDR3 SDRAM (Component)	3-1
Step 1: Memory Device Initialization	3-2
Step 2: Write Training Patterns	3-2
Step 3: Read Resynchronization (Capture) Clock Phase	3-3
Step 4: Read and Write Datapath Timing	3-3
Step 5: Address and Command Clock Cycle	3-3
Step 6: Postamble	3-3
Step 7: Prepare for User Mode	3-4
VT Tracking	3-4
Mimic Path	3-4
DDR3 SDRAM (DIMM)	3-5
Step 1: Memory Device Initialization	3-8
Step 2: Write Leveling	3-8
Step 3: Write Training Patterns	3-8
Step 4: Read Resynchronization	3-8
Step 5: Address and Command Path Clock Cycle	3-8
Step 6: Postamble	3-8
Step 7: Write Clock Path Setup	3-9
Step 8: Prepare for User Mode	3-9
VT Tracking	3-9
Mimic Path	3-9
QDRII+/QDRII SRAM	3-9
Calibration Process	3-11
PHY-to-Controller Interfaces	3-12
Half-Rate Read and Write Data Mapping	3-21
Read Data Mapping	3-22
Write Data Mapping	3-23
Half-Rate Address and Command Clocking	3-24
Ports	3-26

#### **Chapter 4. Support for Stratix IV, Stratix III, HardCopy IV, and HardCopy III Devices**

Introduction	4-1
DDR2/DDR SDRAM	4-1
Half-Rate Support	4-1
Read Datapath	4-1
Clock and Reset Management	4-3
Write Datapath	4-6
Address and Command Datapath	4-7
Full-Rate Support	4-7
Read Datapath	4-7
Clock and Reset Management	4-7
Write Datapath	4-7
Address and Command Datapath	4-8
DDR3 SDRAM	4-8
Read Datapath	4-8
Data Capture, Resynchronization, and Demultiplexing	4-9
Read Data Storage Logic	4-10
Postamble Protection	4-10
Clock and Reset Management	4-10
Clock Management	4-10
Reset Management	4-12
Write Datapath	4-14
Address and Command Datapath	4-15
QDRII+/QDRII SRAM	4-15

Read Datapath . . . . .	4-16
Data Capture, Resynchronization, and Demultiplexing . . . . .	4-16
Data Alignment . . . . .	4-17
Clock and Reset Management . . . . .	4-17
Clock Management . . . . .	4-17
Reset Management . . . . .	4-18
Write Datapath . . . . .	4-18
Address and Command Datapath . . . . .	4-19
x36 Emulation for QDRII+/QDRII SRAM Interfaces . . . . .	4-19
Timing Impact on x36 Emulation . . . . .	4-21
Rules to Combine Groups . . . . .	4-22
Determining the CQ/CQn Arrival Time Skew . . . . .	4-25
Dynamic OCT Support . . . . .	4-26
Write-to-Read Timing Requirement . . . . .	4-27
Read-to-Write Timing Requirement . . . . .	4-28

## Chapter 5. Support for Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Devices

Introduction . . . . .	5-1
DDR2/DDR SDRAM . . . . .	5-1
Half-Rate Support . . . . .	5-1
Read Datapath . . . . .	5-1
Clock and Reset Management . . . . .	5-4
Write Datapath . . . . .	5-10
Address and Command Datapath . . . . .	5-11
Full-Rate Support . . . . .	5-12
Read Datapath . . . . .	5-13
Clock and Reset Management . . . . .	5-13
Write Datapath . . . . .	5-13
Address and Command Datapath . . . . .	5-13
DDR3 SDRAM . . . . .	5-13
Read Datapath . . . . .	5-14
Data Capture and Resynchronization . . . . .	5-14
Data Demultiplexing . . . . .	5-15
Read Data Alignment . . . . .	5-15
Postamble Protection . . . . .	5-15
Clock and Reset Management . . . . .	5-15
Clock Management . . . . .	5-15
Reset Management . . . . .	5-18
Write Datapath . . . . .	5-18
Address and Command Datapath . . . . .	5-19

## Chapter 6. ALTMEMPHY Support for Cyclone III Devices

Introduction . . . . .	6-1
DDR2/DDR SDRAM . . . . .	6-1
Half-Rate Support . . . . .	6-1
Read Datapath . . . . .	6-1
Clock and Reset Management . . . . .	6-2
Reset Management . . . . .	6-3
Write Datapath . . . . .	6-3
Address and Command Datapath . . . . .	6-3

Full-Rate Support	6-4
Read Datapath	6-4
Postamble Protection	6-4
Clock and Reset Management	6-4
Write Datapath	6-4
Address and Command Datapath	6-4
<b>Chapter 7. Integrating the ALTMEMPHY Variation</b>	
Introduction	7-1
Preliminary Steps	7-1
Design Considerations	7-1
Clocks and Resets	7-2
Calibration Process Requirements	7-2
Other Local Interface Requirements	7-2
High-Performance Controller with AFI	7-3
Address and Command Interfacing	7-3
Handshake Mechanism Between Read Commands and Read Data	7-3
Handshake Mechanism Between Write Commands and Write Data	7-4
QDRII+/QDRII SDRAM	7-5
<b>Appendix A. Non-AFI</b>	
Introduction	A-1
PHY-to-Controller Interfaces	A-1
Initialization Timing	A-2
DDR SDRAM Initialization Timing	A-2
DDR2 SDRAM Initialization Timing	A-4
Ports	A-6
Understanding the Testbench	A-18
PLL Initialization and Lock	A-18
Memory Device Initialization	A-19
Interface Training and Calibration	A-19
Write Training Data	A-19
Calibration	A-19
Functional Memory Use	A-20
Functional Simulation—the ModelSim Wave and Transcript Window	A-20
Full Window Stage Identification	A-20
Initialization Stage	A-22
Write Training Data Stage	A-24
Read Calibration Phase	A-26
Functional Memory Use Stage	A-30
Additional Debug Signals	A-32
PLL and PLL Reconfiguration Signals	A-32
Calibration Status Interface	A-33
Additional Calibration Status Interface Signals	A-33
Design Considerations	A-34
Clocks and Resets	A-34
Calibration Process Requirements	A-34
Local Interface Requirements	A-35
DDR2/DDR SDRAM Half-Rate Controller	A-35
Handshake Mechanism Between Read Commands and Read Data	A-36
Handshake Mechanism Between Write Commands and Write Data	A-37

---

DDR2/DDR SDRAM Full-Rate Controller .....	A-39
Handshake Mechanism Between Read Commands and Read Data .....	A-40
Handshake Mechanism Between Write Commands and Write Data .....	A-42

**Chapter Info. Additional Information**

Revision History .....	Info-1
References .....	Info-3





## List of Figures

Figure 1–1: Full-Rate and Half-Rate Controller Description .....	1-2
Figure 1–2: Major Blocks of the ALTMEMPHY Megafunction Interfacing with the Controller and the External Memory (Note 1) .....	1-5
Figure 1–3: Typical Latency Path .....	1-12
Figure 2–1: MegaWizard Plug-In Manager [page 1] .....	2-2
Figure 2–2: MegaWizard Plug-In Manager [page 2a] .....	2-3
Figure 2–3: ALTMEMPHY Parameter Settings Page .....	2-4
Figure 2–4: Preset Editor for the DDR3 SDRAM Variation .....	2-7
Figure 2–5: Preset Editor for DDR2/DDR SDRAM Interfaces .....	2-13
Figure 2–6: Preset Editor for QDRII+/QDRII SRAM Interfaces .....	2-18
Figure 2–7: ALTMEMPHY PHY Parameter Settings Page .....	2-21
Figure 2–8: Controller Interface Settings .....	2-24
Figure 2–9: ALTMEMPHY Simulation Model Generation Page .....	2-25
Figure 2–10: ALTMEMPHY Summary Page .....	2-27
Figure 2–11: ALTMEMPHY Megafunction Generation Window .....	2-28
Figure 2–12: Compilation Report PLL Usage .....	2-36
Figure 2–13: Select a ×18 Device .....	2-38
Figure 2–14: Preset Editor .....	2-39
Figure 2–15: DDR SDRAM High-Performance Controller System-Level Diagram .....	2-41
Figure 2–16: Optimization Technique .....	2-45
Figure 2–17: Physical Synthesis Optimizations .....	2-46
Figure 3–1: Calibration Flow—DDR2/DDR SDRAM and DDR3 SDRAM (Component) .....	3-2
Figure 3–2: Mimic Path in Arria GX, Arria II GX, Cyclone III, Stratix II GX, and Stratix II Devices .....	3-5
Figure 3–3: DDR3 SDRAM Unbuffered Module Clock Topology .....	3-6
Figure 3–4: Calibration Flow—DDR3 SDRAM (DIMM) .....	3-7
Figure 3–5: Resynchronization Clock in QDRII+/QDRII SRAM ALTMEMPHY Megafunction .....	3-10
Figure 3–6: QDRII+/QDRII SRAM Calibration Flowchart .....	3-11
Figure 3–7: AFI PHY Connections .....	3-13
Figure 3–8: Half-Rate Write with Word-Unaligned Data .....	3-14
Figure 3–9: Half-Rate Write with Word-Aligned Data .....	3-14
Figure 3–10: Full-Rate Write .....	3-14
Figure 3–11: Full-Rate Reads .....	3-15
Figure 3–12: Half-Rate Reads .....	3-15
Figure 3–13: Word-Aligned Writes .....	3-18
Figure 3–14: Word-Aligned Reads .....	3-19
Figure 3–15: Word-Unaligned Writes .....	3-20
Figure 3–16: Word-Unaligned Reads .....	3-21
Figure 3–17: Read Data Ordering .....	3-22
Figure 3–18: Data Mapping And Write Datapath .....	3-24
Figure 3–19: Address and Command Clock (0° Phase) (Note 1) .....	3-25
Figure 3–20: Address and Command Clock (90° Phase) (Note 1) .....	3-25
Figure 3–21: Address and Command Clock (180° Phase) (Note 1) .....	3-25
Figure 3–22: Address and Command Clock (270° Phase) (Note 1) .....	3-26
Figure 4–1: DDR2/DDR SDRAM Data Capture and Read Data Mapping in Stratix IV and Stratix III Devices	4-2
Figure 4–2: DDR3 SDRAM Data Capture and Read Data Mapping in Stratix IV and Stratix III Devices	4-9
Figure 4–3: ALTMEMPHY Reset Management Block for Stratix IV and Stratix III Devices .....	4-13
Figure 4–4: DDR3 SDRAM Write Datapath in Stratix IV and Stratix III Devices .....	4-14

Figure 4–5: Write Data Alignment from the DDR3 SDRAM Controller .....	4-15
Figure 4–6: QDRII+/QDRII SRAM Read Datapath .....	4-16
Figure 4–7: QDRII+/QDRII SRAM Write Datapath in Stratix IV and Stratix III Devices .....	4-19
Figure 4–8: Board Trace Connection for Emulated x36 QDRII+/QDRII SRAM Interface .....	4-21
Figure 4–9: Board Simulation Topology Example .....	4-25
Figure 4–10: Board Simulation Results .....	4-26
Figure 4–11: OCT Switching and Read Timing .....	4-27
Figure 5–1: DDR2/DDR SDRAM Read Datapath in Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Devices (Note 1) .....	5-2
Figure 5–2: Relationship Between Postamble Clock and Resynchronization Clock (Note 1) .....	5-3
Figure 5–3: ALTMEMPHY Reset Management Block for Arria GX, Arria II GX, Cyclone III, HardCopy II, Stratix II, and Stratix II GX Devices (Note 1) .....	5-10
Figure 5–4: DDR2/DDR SDRAM Write Datapath in Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Devices .....	5-11
Figure 5–5: Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Address and Command Datapath .....	5-12
Figure 5–6: DDR3 SDRAM Read Datapath in Arria II GX Devices .....	5-14
Figure 5–7: ALTMEMPHY Reset Management Block for Arria II GX Devices .....	5-18
Figure 5–8: DDR3 SDRAM Write Datapath in Arria II GX Devices .....	5-19
Figure 6–1: Cyclone III Read Datapath .....	6-1
Figure 7–1: Address and Command and Read-Path Timing—Full-Rate Design .....	7-3
Figure 7–2: Second Read Alignment—Half-Rate Design .....	7-4
Figure 7–3: Timing for ctl_dqs_burst, ctl_wdata_valid, Address, and Command—Half-Rate Design ..	7-4
Figure 7–4: QDRII+ SRAM Interface Simulation Example .....	7-5
Figure 7–5: QDRII+ SRAM Write Example .....	7-7
Figure 7–6: QDRII+ SRAM Read Example .....	7-9
Figure A–1: The Four ALTMEMPHY Megafunction Interfaces .....	A-1
Figure A–2: DDR SDRAM Device Initialization Timing .....	A-3
Figure A–3: DDR2 SDRAM Device Initialization Timing .....	A-5
Figure A–4: Example Testbench .....	A-21
Figure A–5: DDR2 SDRAM Simulation Initialization Phase .....	A-23
Figure A–6: Writing Data Training .....	A-25
Figure A–7: Read Calibration Phase .....	A-27
Figure A–8: Calibration Phase - Setting the Optimum Phase .....	A-29
Figure A–9: Functional Memory Use Stage .....	A-31
Figure A–10: Read Commands and Read Data (Half-Rate Controller) .....	A-36
Figure A–11: Write Commands and Write Data (Half-Rate Controller) .....	A-38
Figure A–12: Read Commands and Read Data (Full-Rate Controller) .....	A-41
Figure A–13: Write Commands and Write Data (Full-Rate Controller) .....	A-43

## List of Tables

Table 1–1: ALTMEMPHY Memory Support Summary .....	1-1
Table 1–2: ALTMEMPHY Full-Rate and Half-Rate Support (Note 1) .....	1-2
Table 1–3: Glossary of Terms .....	1-3
Table 1–4: Full-rate Design Maximum Supported Frequencies (Note 1), (2) .....	1-7
Table 1–5: Half-rate Design Maximum Supported Frequencies (Note 1), (2) .....	1-8
Table 1–6: Half-rate Design Maximum Supported Frequencies—QDRII+/QDRII SRAM .....	1-8
Table 1–7: Resource Utilization in Arria II GX Devices (Note 1) .....	1-9
Table 1–8: Resource Utilization in Cyclone III Devices (Note 1) .....	1-9
Table 1–9: Resource Utilization in Stratix II Devices (Note 1) and (2) .....	1-10
Table 1–10: Resource Utilization in Stratix III and Stratix IV Devices (Note 1) .....	1-10
Table 1–11: Latency of DDR2 SDRAM Legacy Integrated Static PHY and Controllers .....	1-11
Table 1–12: High Performance Controller Latency Stages and Descriptions .....	1-12
Table 1–13: Typical Read Latency in Arria GX High-Performance Controllers (Note 1), (2) .....	1-13
Table 1–14: Typical Write Latency in Arria GX High-Performance Controllers (Note 1), (2) .....	1-14
Table 1–15: Typical Read Latency in Arria II GX High-Performance Controllers (Note 1), (2) .....	1-14
Table 1–16: Typical Write Latency in Arria II GX High-Performance Controllers (Note 1), (2) .....	1-14
Table 1–17: Typical Read Latency in Cyclone III High-Performance Controllers (Note 1), (2) .....	1-15
Table 1–18: Typical Write Latency in Cyclone III High-Performance Controllers (Note 1), (2) .....	1-16
Table 1–19: Typical Read Latency in Stratix IV and Stratix III High-Performance Controllers (Note 1), (2) 1-16	
Table 1–20: Typical Write Latency in Stratix IV and Stratix III High-Performance Controllers (Note 1), (2) 1-17	
Table 1–21: Typical Read Latency in Stratix II and Stratix II GX High-Performance Controllers (Note 1), (2) .....	1-18
Table 1–22: Typical Write Latency in Stratix II and Stratix II GX High-Performance Controllers (Note 1), (2) .....	1-19
Table 2–1: General Settings .....	2-5
Table 2–2: Memory Presets List .....	2-5
Table 2–3: DDR3 SDRAM Attributes Settings .....	2-8
Table 2–4: DDR3 SDRAM Initialization Options .....	2-9
Table 2–5: DDR3 SDRAM Timing Parameter Settings (Note 1) .....	2-11
Table 2–6: DDR2 SDRAM Attributes Settings .....	2-14
Table 2–7: DDR2/DDR SDRAM Initialization Options .....	2-15
Table 2–8: DDR2/DDR SDRAM Timing Parameter Settings (Note 1) .....	2-16
Table 2–9: QDRII+/QDRII SRAM Attribute Settings .....	2-19
Table 2–10: QDRII+/QDRII SRAM Initialization Options .....	2-20
Table 2–11: QDRII+/QDRII SRAM Timing Parameter Settings .....	2-20
Table 2–12: ALTMEMPHY PHY Settings .....	2-21
Table 2–13: ALTMEMPHY Summary Page .....	2-27
Table 2–14: Generated Files .....	2-28
Table 2–15: Modules in <variation_name>_alt_mem_phy.v File .....	2-30
Table 2–16: PLL Merging Assignments .....	2-33
Table 2–17: ALTMEMPHY Timing Paths (Note 1) .....	2-46
Table 3–1: Interface to the SDRAM Devices (Note 1) .....	3-27
Table 3–2: AFI Signals .....	3-27
Table 3–3: Other Interface Signals .....	3-30
Table 3–4: I/O Interface to QDRII+/QDRII SRAM (Note 1) .....	3-32
Table 3–5: Clock and Reset Signals for QDRII+/QDRII SRAMI .....	3-33


Table 3–6: User-Mode Calibrated OCT Control Signals for QDRII+/QDRII SRAM (Note 1), (2) . . . .	3-34
Table 3–7: Datapath Interface for QDRII+/QDRII SRAM (Note 1) . . . . .	3-34
Table 3–8: Calibration Status Signals for QDRII+/QDRII SRAM (Note 1) . . . . .	3-35
Table 3–9: Parameters . . . . .	3-36
Table 4–1: DDR2 SDRAM Clocking in Stratix IV and Stratix III Devices . . . . .	4-4
Table 4–2: DDR3 SDRAM Clocking Stratix IV and Stratix III Devices . . . . .	4-11
Table 4–3: QDRII+/QDRII SRAM Clocking in Stratix IV and Stratix III Devices . . . . .	4-18
Table 4–4: Possible Group Combinations in Stratix III Devices . . . . .	4-23
Table 4–5: Possible Group Combinations in Stratix IV Devices . . . . .	4-23
Table 5–1: DDR2/DDR SDRAM Clocking in Arria GX, HardCopy II, Stratix II, and Stratix II GX Devices . . . . .	5-5
Table 5–2: DDR2/DDR SDRAM Clocking in Arria II GX Devices . . . . .	5-7
Table 5–3: DDR3 SDRAM Clocking in Arria II GX Devices . . . . .	5-16
Table 6–1: ALTPLL Clocks . . . . .	6-3
Table A–1: I/O Interface for DDR2 and DDR SDRAM—non-AFI (Note 1) . . . . .	A-6
Table A–2: Clock and Reset Signals for DDR2/DDR SDRAM—non-AFI (Note 1) . . . . .	A-7
Table A–3: PLL Reconfiguration Signals for DDR2 and DDR SDRAM—non-AFI . . . . .	A-8
Table A–4: External DLL Signals for DDR2 and DDR SDRAM—non-AFI . . . . .	A-9
Table A–5: User-Mode Calibrated OCT Control Signals for DDR2/DDR SDRAM—non-AFI (Note 1), (2) . . . . .	A-10
Table A–6: Interface to the Memory Controller for DDR2 and DDR SDRAM—non-AFI (Note 1) . . . . .	A-11
Table A–7: Local Interface Signals for DDR2 and DDR SDRAM—non-AFI (Note 1) . . . . .	A-14
Table A–8: Datapath Interface for DDR2 and DDR SDRAM—non-AFI (Note 1) . . . . .	A-15
Table A–9: Calibration Status Interface for DDR2 and DDR SDRAM—non-AFI . . . . .	A-17
Table A–10: Additional Calibration Status Signals—non-AFI . . . . .	A-18
Table A–11: Signals for Calibration Status . . . . .	A-33
Table A–12: ADDR_CMS_ADD_1T Settings . . . . .	A-44

## Device Family Support

Megafunctions provide either full or preliminary support for target Altera device families:

- Full support means the megafunction meets all functional and timing requirements for the device family and can be used in production designs.
- Preliminary support means the megafunction meets all functional requirements, but can still be undergoing timing analysis for the device family.

Table 1–1 shows the level of support offered by the ALTMEMPHY megafunction for each Altera device family.

 The Quartus® II software version 9.0 only supports HardCopy III and HardCopy IV ASICs as a companion devices to the Stratix III and Stratix IV target devices respectively.

**Table 1–1.** ALTMEMPHY Memory Support Summary

Device Family	Memory Type			
	DDR3 SDRAM	DDR2 SDRAM	DDR SDRAM	QDRII+/QDRII SRAM
Arria® GX	—	Full	Full	—
Arria II GX	Preliminary	Preliminary	Preliminary	—
Cyclone® III	—	Full	Full	—
HardCopy® II	—	Full	Full	—
HardCopy III	Preliminary	Preliminary	Preliminary	Preliminary
HardCopy IV	Preliminary	Preliminary	Preliminary	Preliminary
Stratix® II	—	Full	Full	—
Stratix II GX	—	Full	Full	—
Stratix III	Full	Full	Full	Full
Stratix IV	Preliminary	Preliminary	Preliminary	Preliminary

Use the ALTMEMPHY megafunction to create the datapath for DDR3, DDR2, or DDR SDRAM, and QDRII+/QDRII SRAM interfaces. The ALTMEMPHY megafunction offers full-rate or half-rate DDR2 and DDR SDRAM interfaces and half-rate DDR3 SDRAM and QDRII+/QDRII SRAM interfaces.


 There is no support for creating an ALTMEMPHY megafunction variation for RLDRAM II interfaces through the ALTMEMPHY MegaWizard™ Plug-In.

Figure 1–1 shows the differences in the datapath width and frequency at which data is handled between full-rate and half-rate controllers.

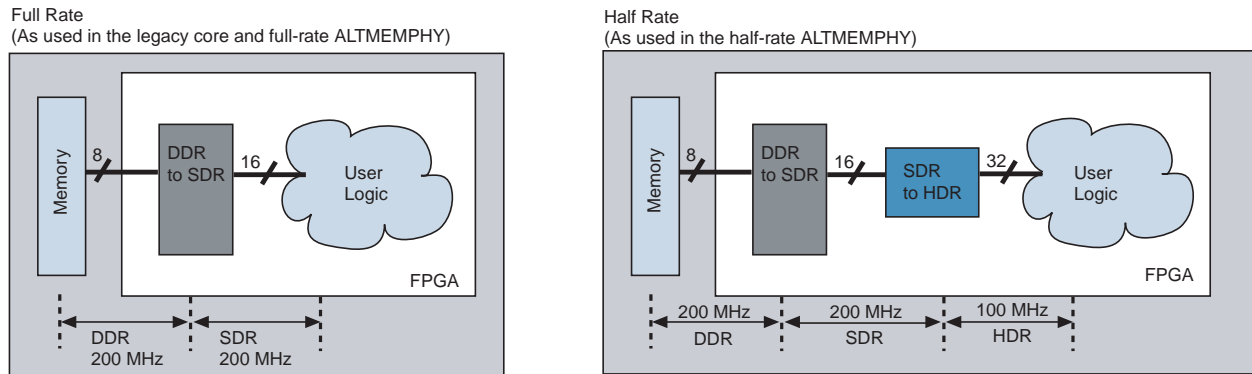
**Figure 1-1.** Full-Rate and Half-Rate Controller Description

Table 1-2 shows the level of support currently offered by the ALTMEMPHY megafunction for each Altera device family.

**Table 1-2.** ALTMEMPHY Full-Rate and Half-Rate Support (Note 1)

Device Family	Memory Type			
	DDR3 SDRAM	DDR2 SDRAM	DDR SDRAM	QDRII+/QDRII SRAM
Arria GX	—	Full-rate and half-rate	Full-rate and half-rate	—
Arria II GX	Half-rate	Full-rate and half-rate	Full-rate and half-rate	—
Cyclone III	—	Full-rate and half-rate	Full-rate and half-rate	—
HardCopy II	—	Full-rate and half-rate	Full-rate and half-rate	—
HardCopy III	Half-rate	Full-rate and half-rate	Full-rate and half-rate	Half-rate
HardCopy IV	Half-rate	Full-rate and half-rate	Full-rate and half-rate	Half-rate
Stratix II	—	Full-rate and half-rate	Full-rate and half-rate	—
Stratix III	Half-rate	Full-rate and half-rate	Full-rate and half-rate	Half-rate
Stratix IV	Half-rate	Full-rate and half-rate	Full-rate and half-rate	Half-rate
Stratix II GX	—	Full-rate and half-rate	Full-rate and half-rate	—

**Note to Table 1-2:**

(1) The ALTMEMPHY megafunction does not support RLDRAM II interfaces.

## Glossary

To understand the functionality of the ALTMEMPHY megafunction, refer to the glossary of terms in [Table 1-3](#).

**Table 1-3.** Glossary of Terms

Term	Description
Altera PHY interface (AFI)	The Altera PHY Interface (AFI) is independent of the controller. ALTMEMPHY instances with the AFI use the sequencer to perform memory initialization and calibration and do not need an external memory controller.
Calibration	The process of setting up the initial relationship between clocks; for example, the resynchronization window to provide the greatest timing margin in DDR3, DDR2, and DDR SDRAM interfaces in Stratix series devices. The initial calibration is done only once at system reset after device initialization is complete.
Double data rate (DDR)	Data that changes on both edges of a clock or strobe operating at the full-rate clock frequency.
Full-rate clock	Clock with a frequency that is equal to the frequency of the memory interface clock.
Full-rate controller	A memory controller implemented as a full-rate design.
Full-rate design	A variation of the ALTMEMPHY megafunction for DDR2 and DDR SDRAM interfaces where the clock frequency of the controller and user-interface logic is the same as the memory interface clock (see <a href="#">Figure 1-1</a> ).
Half data rate (HDR)	Data that changes on one edge of the half-rate clock (twice the width of SDR data and four times the width of DDR data).
Half-rate clock	Clock with a frequency that is half the frequency of the memory interface clock.
Half-rate controller	A memory controller implemented as a half-rate design.
Half-rate design	A variation of the ALTMEMPHY megafunction for DDR3, DDR2, DDR SDRAM, QDR11, and QDR11+ SRAM interfaces where the clock frequency of the controller and user-interface logic is half of the memory interface clock frequency (see <a href="#">Figure 1-1</a> ).
High-performance controller	A memory controller that uses the ALTMEMPHY megafunction for the datapath. Altera offers the high-performance controller for DDR3, DDR2, and DDR SDRAM interfaces.
Legacy controller	The legacy integrated static datapath and controller MegaCore functions with no support for calibration and tracking. For more information about legacy integrated static datapath and controller MegaCore functions, refer to the <a href="#">DDR and DDR2 SDRAM Controller Compiler User Guide</a> , <a href="#">QDR11 SRAM Controller MegaCore Function User Guide</a> , and <a href="#">RLDRAM II Controller MegaCore Function User Guide</a> .
On-chip termination (OCT)	A device feature that eliminates the need of external resistors for termination. The ALTMEMPHY megafunction supports dynamic OCT for DDR3 and DDR2 SDRAM variations for Stratix IV and Stratix III devices and static OCT for QDR11+/QDR11 variations.
Sequencer	The logic block that performs calibration and tracking operations. This module is the only encrypted code in the ALTMEMPHY megafunction.
Single data rate (SDR)	Data that changes on one edge of the full-rate clock (twice the width of DDR data).
Tracking	Tracking is performed as a background process during device operation to track voltage and temperature (VT) variations to maintain the data valid window that was achieved at calibration. Only applies to DDR3, DDR2, and DDR SDRAM interfaces.

## Introduction

The ALTMEMPHY megafunction is an interface between a memory controller and memory devices and performs read and write operations to the memory. The megafunction is available as a stand-alone product or as an integrated product with Altera high-performance memory controllers. As a stand-alone product, use the ALTMEMPHY megafunction with either custom or third-party controllers.

The ALTMEMPHY megafunction creates the datapath between the memory device and the memory controller and user logic in various Altera devices (see [Table 1-1](#)). The easy-to-use ALTMEMPHY megafunction GUI helps you configure multiple variations of a memory interface. Currently, you can use the ALTMEMPHY megafunction to create a datapath for DDR3, DDR2, DDR SDRAM, and QDRII+/QDRII SRAM interfaces.

You can then connect the ALTMEMPHY megafunction variation with either a user-designed controller or with an Altera high-performance controller. Currently, Altera offers a high-performance controller for DDR3, DDR2, and DDR SDRAM interfaces. In addition, the ALTMEMPHY megafunction and the Altera high-performance controllers are available for full-rate and half-rate DDR2 and DDR SDRAM interfaces.

The ALTMEMPHY megafunction supports half-rate DDR3 SDRAM and QDRII+/QDRII SRAM interfaces. Altera currently does not provide a high-performance controller for QDRII+/QDRII SRAM.



For device families not supported by the ALTMEMPHY megafunction (such as Cyclone, Cyclone II, Stratix, and Stratix GX devices), use the Altera legacy integrated static datapath and controller MegaCore functions.

The major advantage of the ALTMEMPHY megafunction is that it supports an initial calibration sequence to remove process variations in both the Altera device and the memory device.

If the ALTMEMPHY megafunction does not suit your needs, you can also create your own memory interface datapath using the ALTDLL and ALTDQ\_DQS megafunctions, available in the Quartus II software.

The ALTMEMPHY megafunction for DDR3, DDR2, and DDR SDRAM offers two different PHY-to-controller interfaces: AFI and non-AFI. The AFI is supported for all variations of ALTMEMPHY for DDR3, DDR2, and DDR SDRAM. ALTMEMPHY for DDR3 SDRAM only support the AFI. The AFI results in a simpler connection between the PHY and controller, so Altera recommends that you use the AFI for new designs; only use the non-AFI for existing designs.

The ALTMEMPHY megafunction for QDRII+/QDRII SRAM is only available with the non-AFI.



The chapters in this user guide only contain information related to the AFI. For non-AFI information, refer to [Appendix A, Non-AFI](#).



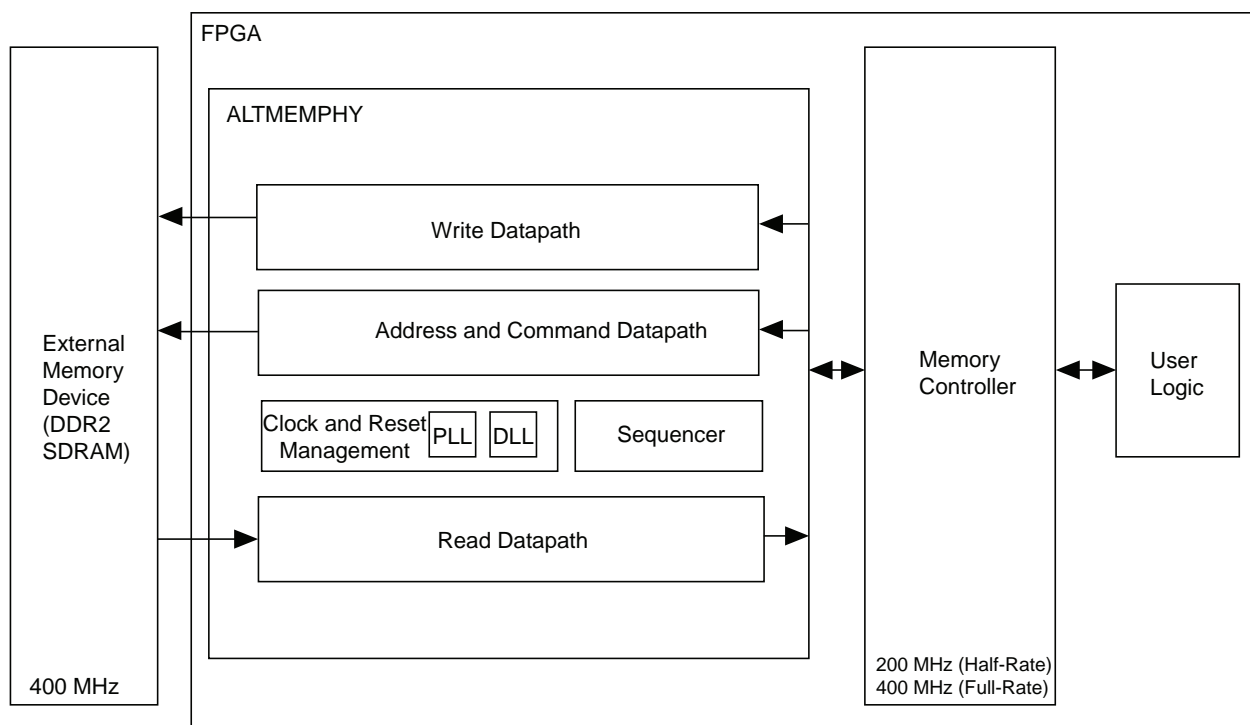
In DDR3, DDR2, and DDR SDRAM interfaces in Arria series and Stratix series devices, the calibration process centers the resynchronization clock phase into the middle of the captured data valid window to maximize the resynchronization setup and hold margin. During the DDR3, DDR2, and DDR SDRAM operation, the VT tracking mechanism eliminates the effects of VT variations on resynchronization timing margin.

In DDR2 and DDR SDRAM interfaces in Cyclone III devices, the ALTMEMPHY calibration process centers the read data capture valid window instead of the resynchronization window. The ALTMEMPHY variation for Cyclone III devices also has a tracking mechanism that monitors and eliminates VT variations on the external memory interface.

In QDRII+ and QDRII SRAM interfaces (supported in Stratix IV and Stratix III devices), the calibration process picks the correct clock edge to align and resynchronize the captured data in the HDR clock domain. Because the memory read clock resynchronizes data to the core logic, there is no tracking mechanism necessary for the ALTMEMPHY variation for QDRII+ and QDRII SRAM interfaces.

Figure 1-2 shows the major blocks of the ALTMEMPHY megafunction and how it interfaces with the external memory device and the controller. The ALTPLL megafunction is instantiated inside the ALTMEMPHY megafunction, so that you do not need to generate the clock to any of the ALTMEMPHY blocks.

**Figure 1-2.** Major Blocks of the ALTMEMPHY Megafunction Interfacing with the Controller and the External Memory  
(Note 1)




**Note to Figure 1-2:**

(1) DDR2 SDRAM at 400 MHz is shown as an example. Ensure that your FPGA and external memory combination can achieve your target speed.

The ALTMEMPHY megafunction is composed of the following blocks:

- Write datapath
- Address and command datapath
- Clock and reset management, including DLL and PLL
- Sequencer
- Read datapath

 For more information on the ALTMEMPHY blocks, refer to: “Support for Stratix IV, Stratix III, HardCopy IV, and HardCopy III Devices” on page 4–1, “Support for Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Devices” on page 5–1, and “ALTMEMPHY Support for Cyclone III Devices” on page 6–1.

## Features

The ALTMEMPHY megafunction offers the following features:

- Simple setup
- Automated initial calibration eliminating complicated read data timing calculations
- VT tracking that guarantees maximum stable performance for DDR3, DDR2, and DDR SDRAM interface
- Self-contained datapath that makes connection to an Altera controller or a third-party controller independent of the critical timing paths
- Full-rate and half-rate DDR2 and DDR SDRAM interfaces; half-rate DDR3 SDRAM and QDRII+/QDRII SRAM interfaces

## What’s New in Version 9.0 and Version 9.0SP1

The ALTMEMPHY megafunction has the following new features:


- Support for DDR3 SDRAM components without leveling on Stratix III and Stratix IV devices
- Full support for Stratix III devices
- Preliminary support for DDR3 SDRAM components and DDR2/DDR SDRAM on Arria II GX devices
- Support for the AFI for DDR3, DDR2 and DDR SDRAM on all supported devices
- Support for DDR3 SDRAM DIMMs with leveling on Stratix III and Stratix IV devices:
  - 533-MHz support—including read deskew and write deskew above 400 MHz
  - 300 to 360-MHz support
  - Support for ×4 DDR3 SDRAM

 For information about issues on the ALTMEMPHY megafunction in a particular Quartus II version, refer to the *Quartus II Software Release Notes*.

## Support and Performance

The performance numbers listed in this section assume the following optimal conditions:

- The highest speed grade of the device family (for FPGA families)
- Commercial operating condition range (for HardCopy-series ASICs)
- Preferred device resource usage, including I/O placement locations (using predefined pins from the device pin tables)

 For complete information on the maximum memory interface performance for every speed grade, refer to the respective device family handbook.

The supported operating frequencies in the following tables are technology maximums. Your design's actual achievable performance is based on design and system-specific factors, and static timing analysis of the completed design.


 Devices with preliminary support also have preliminary maximum supported frequencies until characterization is final.

Table 1-4 shows ALTMEMPHY megafunction maximum supported frequencies for full-rate designs.

**Table 1-4.** Full-rate Design Maximum Supported Frequencies *(Note 1), (2)*

Device	DDR2 SDRAM		DDR SDRAM	
	Single (MHz)	Multiple (MHz)	Single (MHz)	Multiple (MHz)
Arria GX	167	Preliminary	167	Preliminary
Arria II GX	267	Preliminary	200	Preliminary
Cyclone III	200	—	167	—
HardCopy II	267	Preliminary	200	Preliminary
HardCopy III	267	Preliminary	200	Preliminary
HardCopy IV	267	Preliminary	200	Preliminary
Stratix II	267	Preliminary	200	Preliminary
Stratix II GX	267	Preliminary	200	Preliminary
Stratix III	300	Preliminary	200	Preliminary
Stratix IV	300	Preliminary	200	Preliminary

**Notes to Table 1-4:**

- (1) Single refers to single chip select in components and single rank in DIMMs; multiple refers to multiple chip selects in components and multiple ranks in DIMMs.
- (2) For designs with the AFI, calibration is performed on all chip selects, timing analysis only performed on first chip select.

Table 1-5 and Table 1-6 show ALTMEMPHY megafunction maximum supported frequencies for half-rate designs.

**Table 1-5.** Half-rate Design Maximum Supported Frequencies (Note 1), (2)

Device	DDR3 SDRAM		DDR2 SDRAM		DDR SDRAM	
	Single (MHz)	Multiple (MHz)	Single (MHz)	Multiple (MHz)	Single (MHz)	Multiple (MHz)
Arria GX	—	—	233	Preliminary	200	Preliminary
Arria II GX	300 (3)	—	300	Preliminary	200	Preliminary
Cyclone III	—	—	200	—	167	—
HardCopy II	—	—	267	Preliminary	200	Preliminary
HardCopy III	400	—	333	Preliminary	200	Preliminary
HardCopy IV	400	—	333	Preliminary	200	Preliminary
Stratix II	—	—	333	Preliminary	200	Preliminary
Stratix II GX	—	—	333	Preliminary	200	Preliminary
Stratix III	533 (4)	—	400	Preliminary	200	Preliminary
Stratix IV	533 (4)	—	400	Preliminary	200	Preliminary

**Notes to Table 1-5:**

- (1) Single refers to single chip select in components and single rank in DIMMs; multiple refers to multiple chip selects in components and multiple ranks in DIMMs.
- (2) For designs with the AFI, calibration is performed on all chip selects, timing analysis only performed on first chip select.
- (3) Single chip select components only. ALTMEMPHY only supports DDR3 SDRAM component interfacing for Arria II GX devices using T-topology.
- (4) 533 MHz for single rank DIMMs; 400 MHz for components without leveling.

**Table 1-6.** Half-rate Design Maximum Supported Frequencies—QDRII+/QDRII SRAM

Device	QDRII+ SRAM		QDRII SRAM	
	Single Chip Select	Multiple Chip Select	Single Chip Select	Multiple Chip Select
Arria GX	—	—	—	—
Arria II GX	—	—	—	—
Cyclone III	—	—	—	—
HardCopy II	—	—	—	—
HardCopy III	350	Preliminary	300	Preliminary
HardCopy IV	350	Preliminary	300	Preliminary
Stratix II	—	—	—	—
Stratix II GX	—	—	—	—
Stratix III	400	Preliminary	350	Preliminary
Stratix IV	400	Preliminary	350	Preliminary

## Resource Utilization

Table 1-7 through Table 1-10 show the typical size of the ALTMEMPHY megafunction with the AFI in the Quartus II software version 9.0 for the following devices:

- Arria II GX (EP2AGX260FF35C4) devices

- Cyclone III (EP3C16F484C6) devices
- Stratix II (EP2S60F1020C3) devices
- Stratix III (EP3SL110F1152C2) devices
- Stratix IV (EP4SGX230HF35C2) devices

 The resource utilization for Arria and Stratix GX devices is similar to Stratix II devices.

**Table 1-7.** Resource Utilization in Arria II GX Devices *(Note 1)*

Memory Type	PHY Rate	Memory Width (Bits)	Combinational ALUTS	Logic Registers	M9K Blocks	Memory ALUTs
DDR3 SDRAM (without leveling)	Half	8	1,431	1,189	2	18
		16	1,481	1,264	4	2
		64	1,797	1,970	12	22
		72	1,874	2,038	13	2
DDR2/DDR SDRAM	Half	8	1,428	1,179	2	18
		16	1,480	1,254	4	2
		64	1,787	1,960	12	22
		72	1,867	2,027	13	2
	Full	8	1,232	975	0	35
		16	1,240	915	3	1
		64	1,287	1,138	7	41
		72	1,303	1,072	9	1

**Notes to Table 1-7:**

- (1) The listed resource utilization refers to resources used by the ALTMEMPHY megafunction with AFI only. Memory controller overhead is additional.

**Table 1-8.** Resource Utilization in Cyclone III Devices *(Note 1)*

Memory Type	PHY Rate	Memory Width (Bits)	Combinational LUTs	Logic Registers	M9K Blocks
DDR2/DDR SDRAM	Half	8	1,995	1,199	2
		16	2,210	1,396	3
		64	3,523	2,574	9
		72	3,770	2,771	9
	Full	8	1,627	870	2
		16	1,762	981	2
		64	2,479	1,631	5
		72	2,608	1,740	5

**Notes to Table 1-8:**

- (1) The listed resource utilization refers to resources used by the ALTMEMPHY megafunction with AFI only. Memory controller overhead is additional.

**Table 1-9.** Resource Utilization in Stratix II Devices (Note 1) and (2)

Memory Type	PHY Rate	Memory Width (Bits)	Combinational LUTs	Logic Registers	M512K Blocks	M4K Blocks
DDR2/DDR SDRAM	Half	8	1,444	1,201	4	1
		16	1,494	1,375	4	2
		64	1,795	2,421	5	7
		72	1,870	2,597	4	8

**Notes to Table 1-9:**

- (1) The listed resource utilization refers to resources used by the ALTMEMPHY megafunction with AFI only. Memory controller overhead is additional.
- (2) The resource utilization for Arria and Stratix GX devices is similar to Stratix II devices.

**Table 1-10.** Resource Utilization in Stratix III and Stratix IV Devices (Note 1)

Memory Type	PHY Rate	Memory Width (Bits)	Combinational ALUTs	Logic Registers	M9K Blocks	Memory ALUTs
DDR3 SDRAM (400 MHz, without leveling only)	Half	8	1,369	1,045	1	40
		16	1,428	1,196	1	80
		64	1,824	2,080	1	320
		72	1,920	2,226	1	360
DDR3 SDRAM (400 MHz, with leveling only)	Half	8	3,451	2,527	2	62
		16	4,049	3,057	2	124
		64	7,370	6,157	5	504
		72	7,977	6,676	5	567
DDR3 SDRAM (533 MHz with read and write deskew, with leveling only)	Half	8	3,886	2,650	2	62
		16	4,493	3,185	2	124
		64	7,808	6,257	5	504
		72	8,374	6,770	5	567
DDR2/DDR SDRAM	Half	8	1,356	1,040	1	40
		16	1,423	1,189	1	80
		64	1,805	2,072	1	320
		72	1,902	2,220	1	360
	Full	8	1,216	918	1	20
		16	1,229	998	1	40
		64	1,319	1,462	1	160
		72	1,337	1,540	1	180

**Notes to Table 1-10:**

- (1) The listed resource utilization refers to resources used by the ALTMEMPHY megafunction with AFI only. Memory controller overhead is additional.

## Latency

Latency is defined using the user (local) side frequency and absolute time (ns). There are two types of latencies that exist while designing with memory controllers—read and write latencies. Altera has the following definitions for read and write latencies:

- Read latency—the amount of time it takes for the read data to appear at the user (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.



For a half-rate controller, the user (local) side frequency is half of the memory interface frequency. For a full-rate controller, the user (local) side frequency is equal to the memory interface frequency.

Altera defines read and write latencies in terms of the local interface clock frequency and by the absolute time for the memory controllers. These latencies apply to supported device families ( see [Table 1-1 on page 1-1](#)) with the following memory controllers:

- Legacy DDR and DDR2 SDRAM controller
- Half-rate high-performance controller
- Full-rate high-performance controller

The latency defined in this section uses the following assumptions:

- The row is already open (there is no extra bank management needed)
- The controller is idle (there is no queued transaction pending, indicated by the `local_ready` signal asserted high)
- No refresh cycles occur before the transaction

### Legacy Integrated Static Datapath and Controller

[Table 1-11](#) lists the latency for the DDR2 SDRAM legacy controller for the Cyclone II, HardCopy II, Stratix II GX, and Stratix II device families. This document focuses on ALTMEMPHY-based variants so the legacy information is listed here as a convenience for comparison purposes.

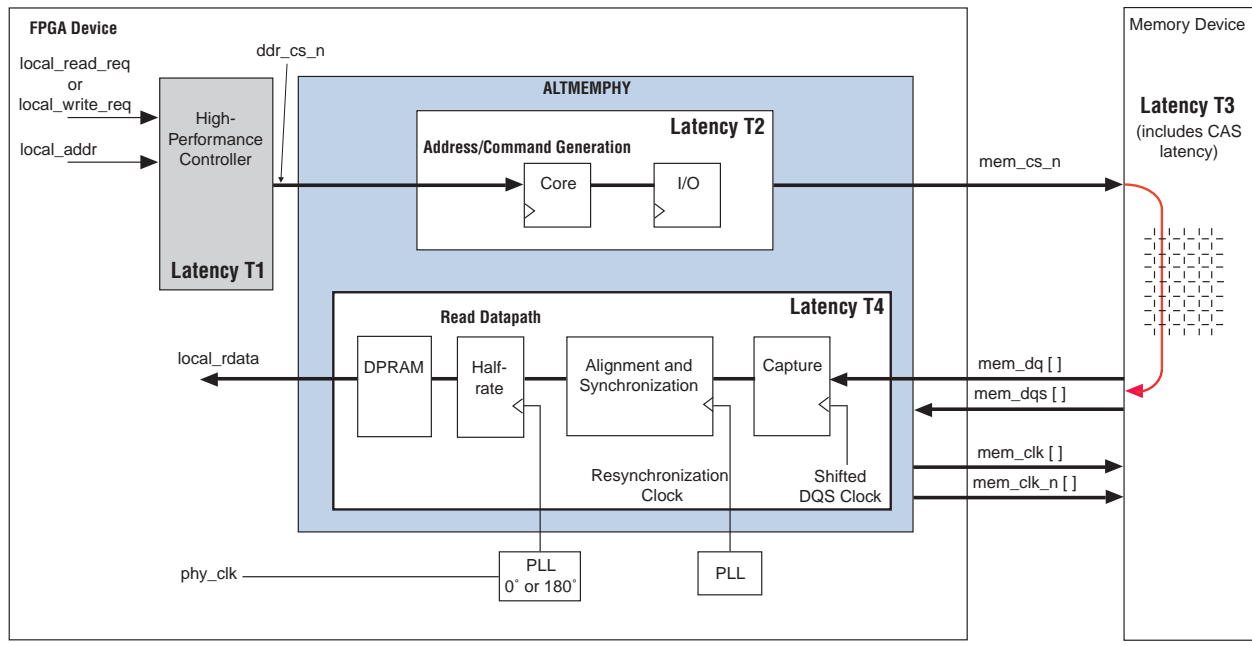
**Table 1-11.** Latency of DDR2 SDRAM Legacy Integrated Static PHY and Controllers

Full Rate	Frequency (MHz)	Latency (Local Clock Cycles)		Latency in Time (ns)	
		Read	Write	Read	Write
Cyclone II	167	13	9	78	54
HardCopy II	267	13	9	49	34
Stratix II GX	267	13	9	49	34
Stratix II	267	13	9	49	34

## High-Performance Controllers

The latency for the high-performance controller comprises many different stages of the memory interface. [Figure 1-3](#) shows a typical memory interface read latency path showing read latency from the time a `local_read_req` assertion is detected by the controller up to data available to be read from the dual-port RAM (DPRAM) module.

**Figure 1-3.** Typical Latency Path



[Table 1-12](#) shows the different stages that make up the whole read and write latency that [Figure 1-3](#) shows.

**Table 1-12.** High Performance Controller Latency Stages and Descriptions

Latency Number	Latency Stage	Description
T1	Controller	<code>local_read_req</code> or <code>local_write_req</code> signal assertion to <code>ddr_cs_n</code> signal assertion.
T2	Command Output	<code>ddr_cs_n</code> signal assertion to <code>mem_cs_n</code> signal assertion.
T3	CAS or WL	Read command to DQ data from the memory or write command to DQ data to the memory.
T4	ALTMEMPHY read data input	Read data appearing on the local interface.
T2 + T3	Write data latency	Write data appearing on the memory interface.

From [Figure 1-3](#), the read latency in the high-performance controllers is made up of four components:

$$\text{Read latency} = \text{controller latency} + \text{command output latency} + \text{CAS latency} + \text{PHY read data input latency} = T1 + T2 + T3 + T4$$



Similarly, the write latency in the high-performance controllers is made up of three components:

$$\text{Write latency} = \text{controller latency} + \text{write data latency} = T1 + T2 + T3$$

You can separate the controller and ALTMEMPHY read data input latency into latency that occurred in the I/O element (IOE) and latency that occurred in the FPGA fabric.

Table 1-13 through Table 1-22 show a typical latency that can be achieved in Arria GX, Arria II GX, Cyclone III, Stratix IV, Stratix III, Stratix II, and Stratix II GX devices. The exact latency for your memory controller depends on your precise configuration. You can obtain precise latency from simulation, but this figure can vary slightly in hardware because of the automatic calibration process.

The actual memory CAS and write latencies shown are halved in half-rate designs as the latency calculation is based on the local clock.

The read latency also depends on your board trace delay. The latency found in simulation can be different from that found in board testing as functional simulation does not take into account the board trace delays. For a given design on a given board, the latency may change by one clock cycle (for full-rate designs) or two clock cycles (for half-rate designs) upon resetting the board. Different boards could also show different latencies even with the same design.

The CAS and write latencies are different between DDR2 and DDR SDRAM interfaces. To calculate latencies for DDR SDRAM interfaces, use the numbers from DDR2 SDRAM listed below and replace the CAS and write latency with the DDR SDRAM values.

**Table 1-13.** Typical Read Latency in Arria GX High-Performance Controllers (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		CAS Latency (4)	Read Data Latency		Total Read Latency (5)	
				FPGA	I/O		FPGA	I/O	Local Clock Cycles	Time (ns)
DDR2 SDRAM	233	Half-rate	5	3	1	2	5.5	1	19	163
	167	Full-rate	4	2	1	4	8	1	21	126

**Notes to Table 1-13:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) CAS latency is per memory device specification and is programmable in the MegaWizard Plug-In.
- (5) Total read latency is the sum of controller, address and command, CAS, and read data latencies.

**Table 1-14.** Typical Write Latency in Arria GX High-Performance Controllers (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
DDR2 SDRAM	233	Half-rate	5	3	1	1.5	12	103
	167	Full-rate	4	2	1	3	11	66

**Notes to Table 1-14:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) Memory write latency is per memory device specification. The latency from when you provide the command to write to when you need to provide data at the memory device.
- (5) Total write latency is the sum of controller, address and command, and memory write latencies.

**Table 1-15.** Typical Read Latency in Arria II GX High-Performance Controllers (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		CAS Latency (4)	Read Data Latency		Total Read Latency (5)	
				FPGA	I/O		FPGA	I/O	Local Clock Cycles	Time (ns)
DDR3 SDRAM	400	Half-rate	5	3	1	3	6.5	1	19	95
DDR2 SDRAM	233	Half-rate	5	3	1	2.5	6.5	1	19	163
	167	Full-rate	4	2	1	4	8	1	21	16

**Notes to Table 1-15:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) CAS latency is per memory device specification and is programmable in the MegaWizard Plug-In.
- (5) Total read latency is the sum of controller, address and command, CAS, and read data latencies.

**Table 1-16.** Typical Write Latency in Arria II GX High-Performance Controllers (Part 1 of 2) (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
DDR3 SDRAM	400	Half-rate	5	4	1	2.5	14	68

**Table 1-16.** Typical Write Latency in Arria II GX High-Performance Controllers (Part 2 of 2) (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
DDR2 SDRAM	233	Half-rate	5	3	1	2.5	12	103
	167	Full-rate	4	2	1	4	11	66

**Notes to Table 1-16:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) Memory write latency is per memory device specification. The latency from when you provide the command to write to when you need to provide data at the memory device.
- (5) Total write latency is the sum of controller, address and command, and memory write latencies.

**Table 1-17.** Typical Read Latency in Cyclone III High-Performance Controllers (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		CAS Latency (4)	Read Data Latency		Total Read Latency (5)	
				FPGA	I/O		FPGA	I/O	Local Clock Cycles	Time (ns)
DDR2 SDRAM	200	Half-rate	5	3	1	2	5.5	1	19	190
	167	Full-rate	4	2	1	4	8	1	21	126

**Notes to Table 1-17:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) CAS Latency is per memory device specification and is programmable in the MegaWizard Plug-In.
- (5) Total read latency is the sum of controller, address and command, CAS, and read data latencies.

**Table 1-18.** Typical Write Latency in Cyclone III High-Performance Controllers (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
DDR2 SDRAM	200	Half-rate	5	3	1	1.5	12	120
	167	Full-rate	4	2	1	3	11	66

**Notes to Table 1-18:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) Memory write latency is per memory device specification. The latency from when you provide the command to write to when you need to provide data at the memory device.
- (5) Total write latency is the sum of controller, address and command, and memory write latencies.

**Table 1-19.** Typical Read Latency in Stratix IV and Stratix III High-Performance Controllers (Part 1 of 2) (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		CAS Latency (4)	Read Data Latency		Total Read Latency (5)	
				FPGA	I/O		FPGA	I/O	Local Clock Cycles	Time (ns)
DDR3 SDRAM (with leveling)	400	Half-rate	5	3	1	3	7	2	21	115
DDR3 SDRAM (without leveling)	400	Half-rate	5	2.5	1	3	8	1.5	21	115
DDR2 SDRAM	400	Half-rate	5	3	1	2.5	7.125	1.5	20	100
DDR2 SDRAM	267	Full-rate	4	2	1.5	4	9 (6)	1	21	85
QDRII+ SRAM (7)	350	Half-rate	5	2	0.66	1.25	8	1.2	20	110

**Table 1-19.** Typical Read Latency in Stratix IV and Stratix III High-Performance Controllers (Part 2 of 2) (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		CAS Latency (4)	Read Data Latency		Total Read Latency (5)	
				FPGA	I/O		FPGA	I/O	Local Clock Cycles	Time (ns)
QDRII SRAM (7)	350	Half-rate	5	2	0.66	0.75	8	1.2	19	107

**Notes to Table 1-19:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) CAS Latency is per memory device specification and is programmable in the MegaWizard Plug-In for DDR3/DDR2/DDR SDRAM device. Stratix IV and Stratix III only support QDRII+ SRAM with latency of 2.5 and QDRII SRAM with latency of 1.5.
- (5) Total read latency is the sum of controller, address and command, CAS, and read data latencies.
- (6) This latency is may be 10, because of the 720° calibration phase sweep.
- (7) Altera does not offer a high-performance controller for QDRII+/QDRII SRAM. The number shown here uses an example driver with non-deterministic latency in the QDRII+/QDRII SRAM ALTMEMPHY variation. QDRII+/QDRII SRAM devices also offer deterministic latency ALTMEMPHY variation as described in "QDRII+/QDRII SRAM Deterministic Latency" on page 2-23.

**Table 1-20.** Typical Write Latency in Stratix IV and Stratix III High-Performance Controllers (Part 1 of 2) (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
DDR3 SDRAM (with leveling)	400	Half-rate	5	3	1	2.5	13	68
DDR3 SDRAM (without leveling)	400	Half-rate	5	2.5	1	2.5	11	68
DDR2 SDRAM	400	Half-rate	5	3	1	2	12	60
DDR2 SDRAM	267	Full-rate	4	2	1.5	3	12	44
QDRII+ SRAM (6)	350	Half-rate	4	2	0.66	0.5	10	53

**Table 1-20.** Typical Write Latency in Stratix IV and Stratix III High-Performance Controllers (Part 2 of 2) (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
QDRII SRAM (6)	350	Half-rate	4	2	0.66	0.5	10	53

**Notes to Table 1-20:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) Memory write latency is per memory device specification. The latency from when you provide the command to write to when you need to provide data at the memory device.
- (5) Total write latency is the sum of controller, address and command, and memory write latencies.
- (6) Altera does not offer a high-performance controller for QDRII+/QDRII SRAM. The number shown here uses an example driver with non-deterministic latency ALTMEMPHY variation. QDRII+/QDRII SRAM devices also offer deterministic latency ALTMEMPHY variation as described in "QDRII+/QDRII SRAM Deterministic Latency" on page 2-23.

**Table 1-21.** Typical Read Latency in Stratix II and Stratix II GX High-Performance Controllers (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		CAS Latency (4)	Read Data Latency		Total Read Latency (5)	
				FPGA	I/O		FPGA	I/O	Local Clock Cycles	Time (ns)
DDR2 SDRAM	333	Half-rate	5	3	1	2	5.5	1	19	114
DDR2 SDRAM	267	Half-rate	5	3	1	2	5.5	1	19	143
DDR2 SDRAM	200	Full-rate	4	2	1	4	8	1	21	105

**Notes to Table 1-21:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) CAS Latency is per memory device specification and is programmable in the MegaWizard Plug-In.
- (5) Total read latency is the sum of controller, address and command, CAS, and read data latencies.

**Table 1-22.** Typical Write Latency in Stratix II and Stratix II GX High-Performance Controllers (Note 1), (2)

Memory Standard	Frequency (MHz)	Interface	Controller Latency (3)	Address and Command Latency		Memory Write Latency (4)	Total Write Latency (5)	
				FPGA	I/O		Local Clock Cycles	Time (ns)
DDR2 SDRAM	333	Half-rate	5	3	1	1.5	12	72
DDR2 SDRAM	267	Half-rate	5	3	1	1.5	12	90
DDR2 SDRAM	200	Full-rate	4	2	1	3	11	55

**Notes to Table 1-22:**

- (1) These are typical latency values using the assumptions listed in the beginning of the section. Your actual latency may be different than shown. Perform your own simulation for your actual latency.
- (2) Numbers shown may have been rounded up to the nearest higher integer.
- (3) The controller latency value is from the Altera high-performance controller.
- (4) Memory write latency is per memory device specification. The latency from when you provide the command to write to when you need to provide data at the memory device.
- (5) Total write latency is the sum of controller, address and command, and memory write latencies.




To see the latency incurred in the IOE for both read and write paths for ALTMEMPHY variations in Stratix IV and Stratix III devices refer to the IOE figures in the *External Memory Interfaces in Stratix III Devices* chapter of the *Stratix III Device Handbook* and the *External Memory Interfaces in Stratix IV Devices* chapter of the *Stratix IV Device Handbook*.





## System Requirements

The instructions in this section require the Quartus® II software version 9.0 or higher

 For Operating System (OS) support information, refer to:

[www.altera.com/support/software/os\\_support/oss-index.html](http://www.altera.com/support/software/os_support/oss-index.html)


You can implement an ALTMEMPHY megafunction using either one of the following flows:

- SOPC Builder flow
- MegaWizard Plug-In Manager flow

 For information on the SOPC Builder flow, refer to *AN517: Using High-Performance DDR, DDR2, DDR3 SDRAM with SOPC Builder*.

## Using the MegaWizard Plug-In Manager

Use the MegaWizard Plug-In Manager to create or modify design files that contain custom megafunction variations, which you can then instantiate in a design file. The MegaWizard Plug-In Manager provides a wizard that allows you to specify options for the ALTMEMPHY megafunction features in your design.

 For more information about how to use the MegaWizard Plug-In Manager, refer to Quartus II Help.

Always regenerate your ALTMEMPHY megafunction or high-performance controller in the latest Quartus II software version to ensure that you have the latest and most robust PHY or controller. When regenerating the ALTMEMPHY megafunction or controller, rerun the .tcl files generated by the MegaWizard Plug-In, in case there are changes in the logic assignments in the new software version. You may also need to remove the old assignments from previous versions of the Quartus II software that are no longer required in the current Quartus II software version.

Start the MegaWizard Plug-In Manager in one of the following ways:

- On the **Tools** menu, click **MegaWizard Plug-In Manager**.
- When working in **Block Editor**, from the **Edit** menu, click **Insert Symbol as Block**, or right-click in **Block Editor**, point to **Insert**, and click **Symbol as Block**. In the Symbol window, click **MegaWizard Plug-In Manager**.
- Start the stand-alone version of the MegaWizard Plug-In Manager by typing the following command at the command prompt:  
qmegawiz ↵

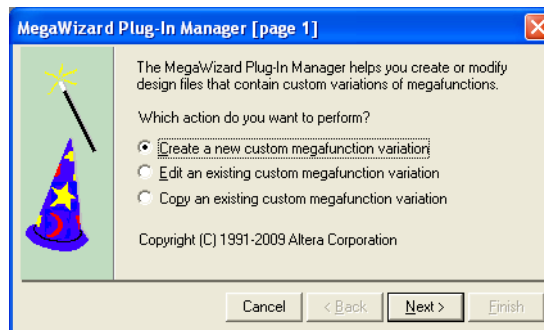
- The ALTMEMPHY megafunction is also instantiated when an Altera high-performance controller is generated using the MegaWizard Plug-In. There is no need to launch the ALTMEMPHY MegaWizard Plug-In separately. For more information about the high-performance controller, refer to the *DDR3 SDRAM High-Performance Controller User Guide* and the *DDR and DDR2 SDRAM High-Performance Controller User Guide*.

## MegaWizard Plug-In Manager Page Descriptions

This section provides descriptions of the options available on the ALTMEMPHY MegaWizard Plug-In pages. You can use the same GUI to instantiate the ALTMEMPHY megafunction for a DDR3, DDR2, DDR SDRAM, and QDRII+/QDRII SRAM interface.

On page 1 of the MegaWizard Plug-In Manager, select **Create a new custom megafunction variation** from the three available options, see [Figure 2-1](#), and click **Next**.

**Figure 2-1.** MegaWizard Plug-In Manager [page 1]

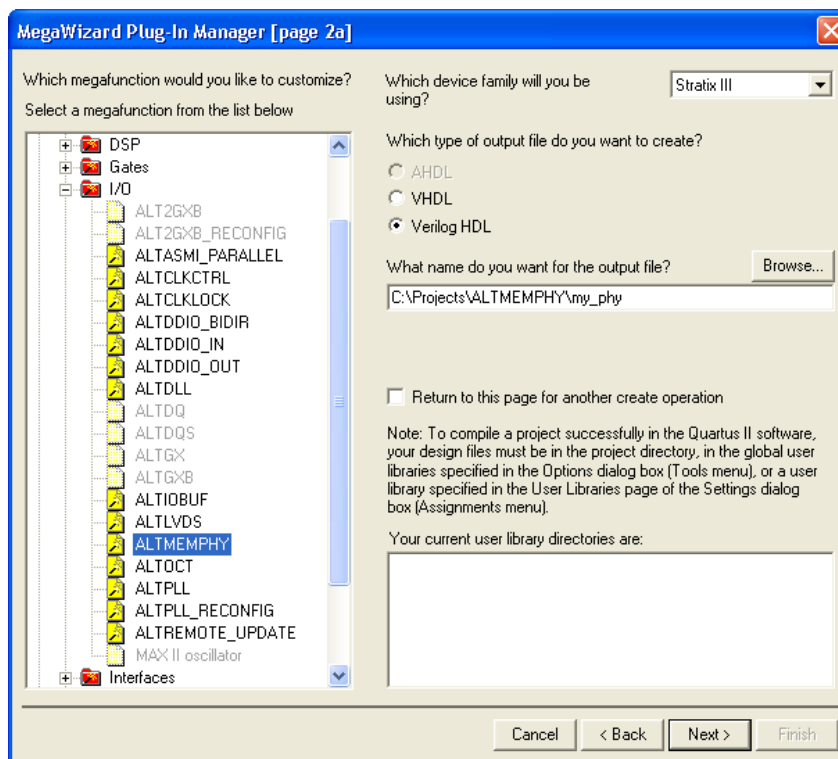


On page 2a of the MegaWizard Plug-In Manager, select **ALTMEMPHY** by expanding **I/O** in the **Megafunction** list. Select the appropriate device family, output file type, and name of your output file.

[Figure 2-2](#) shows an example of an ALTMEMPHY megafunction with the variation name **my\_phy.v**.

- The *<variation name>* must be a different name from the project name and the top-level design entity name.

Figure 2-2. MegaWizard Plug-In Manager [page 2a]

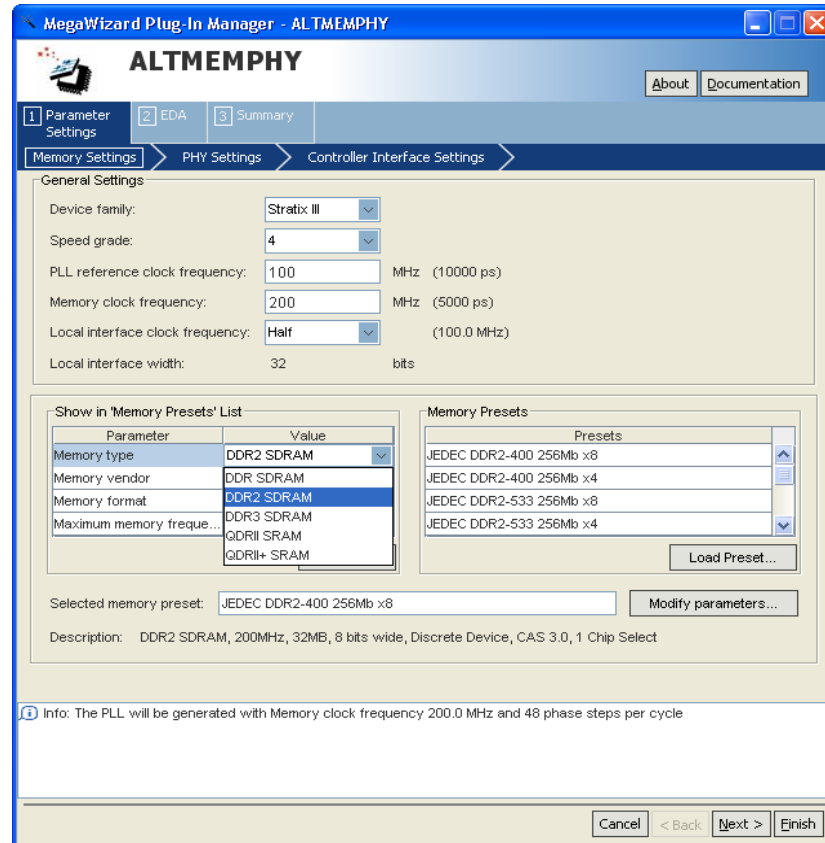


## ALTMEMPHY Parameter Settings Page

Click Next to launch the **ALTMEMPHY Parameter Settings** page, see [Figure 2-3](#). By clicking on the appropriate tab, this page allows you to parameterize:

- Memory settings
- PHY settings
- Controller interface settings

Figure 2-3. ALTMEMPHY Parameter Settings Page



The text window at the bottom of the MegaWizard Plug-In displays information about the memory interface, warnings (for example, if you are creating an interface above the maximum frequency supported), and errors if you are trying to create something that is not supported (for instance, a full-rate ALTMEMPHY megafunction for QDRII+/QDRII SRAM interfaces). The **Finish** button is disabled until you fix all the errors indicated in this window.

The following sections describe the three tabs of the **Parameter Settings** page in more details.

### Memory Settings

In the **Memory Settings** tab, you can choose the frequency of operation for the device and a particular memory device for your system. Under General Settings, you can choose the device family, speed grade, and clock information. In the middle of the page (left-side), see Figure 2-3, you can filter the available memory device listed on the right side of the **Memory Presets** page. If you cannot find the exact device that you are using, choose a device that has the closest specifications, then manually modify the parameters to match your actual device by clicking **Modify parameters...** next to the **Selected memory preset** field.

Table 2-1 describes the **General Settings** available on the **Memory Settings** page of the ALTMEMPHY MegaWizard Plug-In.

**Table 2-1.** General Settings

Parameter Name	Description
Device family	Targets device family (for example, Stratix III). Table 1-1 on page 1-1 shows supported device families. When targeting a HardCopy IV or HardCopy III device, target a Stratix IV or Stratix III device and choose the HardCopy IV or HardCopy III device as a companion device in the Quartus II Project Settings. The device family selected here must match the device family selected on MegaWizard page 2a, see Figure 2-2 on page 2-3.
Speed grade	Selects a particular speed grade of the device (for example, 2, 3, or 4 for the Stratix III device family).
PLL reference clock frequency	Determines the clock frequency of the external input clock to the PLL. Ensure that you use three decimal points if the frequency is not a round number (for example, 166.667 MHz or 100 MHz) to avoid a functional simulation or PLL locking issue on the board.
Memory clock frequency	Determines the memory interface clock frequency. If you are operating a memory device below its maximum achievable frequency, ensure that you enter the actual frequency of operation rather than the maximum frequency achievable by the memory device. Also, ensure that you use three decimal points if the frequency is not a round number (for example, 333.333 MHz or 400 MHz) to avoid a functional simulation or PLL locking issue on the board.
Local interface clock frequency	Sets the frequency of the controller to equal to either the memory interface frequency (full-rate) or half of the memory interface frequency (half-rate). The full-rate option is only available if you are creating an interface for DDR2 or DDR SDRAM devices.



There is an uneditable field called “Local interface width” under the **General Settings** tab. This field’s value changes with the memory device that you choose from the **Memory Presets** list.

Table 2-2 describes the options available to filter the **Memory Presets** that are displayed. This section is where you indicate whether you are creating a datapath for DDR3/DDR2/DDR SDRAM, or QDRII+/QDRII SRAM.

**Table 2-2.** Memory Presets List

Parameter Name	Description
Memory type	You can filter the type of memory to display (for example, DDR2 SDRAM). Currently the ALTMEMPHY megafunction supports DDR3 SDRAM, DDR2 SDRAM, DDR SDRAM, QDRII+ SRAM, and QDRII SRAM.
Memory vendor	You can filter the memory types by vendor. JEDEC is also one of the options, allowing you to choose the JEDEC specifications. If your chosen vendor is not listed, you can choose JEDEC (for DDR3, DDR2, and DDR SDRAM interfaces) or Other for QDRII+/QDRII SRAM interfaces. Then, pick a device that has similar specifications to your chosen device and check the values of each parameter. Make sure you change the each parameter value to match your device specifications.
Memory format	You can filter the type of memory by format (for example, components or DIMM packages). This option is only available for DDR3, DDR2, and DDR SDRAM interfaces.
Maximum frequency	You can filter the type of memory by the maximum operating frequency.

Pick a device in the **Memory Presets** list that is closest or the same as the actual memory device that you are using. Then, click the **Modify Parameters** button to parameterize the following settings in the **Preset Editor** page:

- Memory attributes—These are the settings that determine your system’s number of DQ, DQS, address, and memory clock pins.
- Memory initialization options—These settings are stored in the memory mode registers as part of the initialization process.
- Memory timing parameters—These are the parameters that create and time-constrain the PHY.



Even though the device you are using is listed in **Memory Presets**, ensure that the settings in the **Preset Editor** are accurate as some parameters may have been updated in the memory device datasheets.

You can change the parameters with a white background to reflect your system. You can also change the parameters with a gray background so the device parameters match the device you are using. These parameters in gray background are characteristics of the chosen memory device and changing them creates a new custom memory preset. If you click **Save As** (at the bottom left of the page) and save the new settings in the `<quartus_install_dir>\quartus\common\ip\altmemphy\lib\` directory, you can use this new memory preset in other Quartus II projects created in the same version of the software.

There are three different variations of the **Preset Editor** (described in the next sections):

- DDR3 SDRAM
- DDR2/DDR SDRAM
- QDRII+/QDRII SRAM

## DDR3 SDRAM Preset Editor Page

Figure 2-4 shows the **Preset Editor** page for the ALTMEMPHY variation for DDR3 SDRAM interfaces.

**Figure 2-4.** Preset Editor for the DDR3 SDRAM Variation

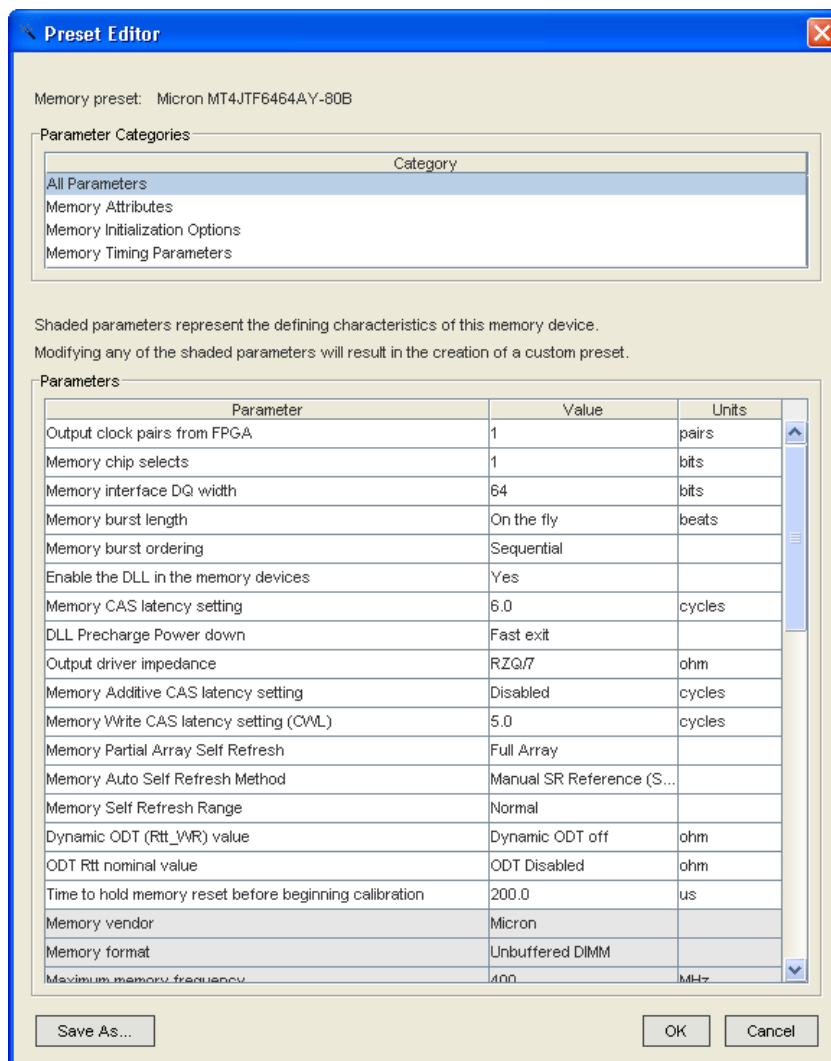


Table 2-3 through Table 2-5 describe the DDR3 SDRAM parameters available for memory attributes, initialization options, and timing parameters.

**Table 2-3.** DDR3 SDRAM Attributes Settings (Part 1 of 2)

Parameter Name	Range (1)	Units	Description
Output clock pairs from FPGA	1-6	pairs	Defines the number of differential clock pairs driven from the FPGA to the memory. The ALTMEMPHY MegaWizard Plug-In displays an error on the bottom of the window if you choose more than one for DDR3 SDRAM interfaces. Memory clock pins use the signal splitter feature in Arria II GX, Stratix IV and Stratix III devices for differential signaling.
Memory chip selects	1, 2, 4, or 8	bits	Sets the number of chip selects in your memory interface. The depth of your memory in terms of number of chips. You are limited to the range shown as the local side binary encodes the chip select address. You can set this value to the next higher number if the range does not meet your specifications. However, the highest address space of the ALTMEMPHY megafunction is not mapped to any of the actual memory addresses. The ALTMEMPHY megafunction calibrates against all chip select signals. The ALTMEMPHY MegaWizard Plug-In displays an error on the bottom of the window if you choose more than one for DDR3 SDRAM interfaces.
Memory interface DQ width	4-288	bits	Defines the total number of DQ pins on the memory interface. If you are interfacing with multiple devices, multiply the number of devices with the number of DQ pins per device. Even though the GUI allows you to choose 288-bit DQ width, DDR3 SDRAM variations are only supported up to 80-bit width due to restrictions in the board layout which affects timing at higher data width. Furthermore, the interface data width is limited by the number of pins on the device. For best performance, have the whole interface on one side of the device.
Memory vendor	Elpida, JEDEC, Micron, Samsung, Hynix, Nanya, other	—	Describes the name of the memory vendor for all supported memory standards.
Memory format	Discrete Device, Unbuffered DIMM	—	Specifies whether you are interfacing with devices or modules. SODIMM and MicroDIMM are supported under unbuffered DIMMs. The ALTMEMPHY megafunction for DDR3 SDRAM interfaces does not support registered DIMM format. Arria II GX devices only support DDR3 SDRAM components without leveling.
Maximum memory frequency	See the memory device datasheet	MHz	Sets the maximum frequency supported by the memory.
Column address width	10-12	bits	Defines the number of column address bits for your interface.



**Table 2-3.** DDR3 SDRAM Attributes Settings (Part 2 of 2)

Parameter Name	Range (1)	Units	Description
Row address width	12–16	bits	Defines the number of row address bits for your interface. If your DDR3 SDRAM device's row address bus is 12-bit wide, set the row address width to <b>13</b> and set the 13 <sup>th</sup> bit to logic-level low (or leave the 13 <sup>th</sup> bit unconnected to the memory device) in the top-level file.
Bank address width	3	bits	Defines the number of bank address bits for your interface.
Chip selects per DIMM	1 or 2	bits	Defines the number of chip selects on each DIMM in your interface. Currently, only one chip select per DIMM is supported for DDR3 SDRAM interfaces, so you can only interface with one single-rank DIMM. The ALTMEMPHY MegaWizard Plug-In Manager displays an error on the bottom of the window if you choose more than one for DDR3 SDRAM interfaces.
DQ bits per DQS bit	4 or 8	bits	Defines the number of data (DQ) bits for each data strobe (DQS) pin.
Drive DM pins from FPGA	Yes or No	—	Specifies whether you are using DM pins for write operation. Altera devices do not support DM pins with ×4 mode DDR3 SDRAM devices.
Maximum memory frequency for CAS latency 5.0	80–700	MHz	Frequency limits from the memory data sheet per given CAS latency. The ALTMEMPHY MegaWizard Plug-In generates a warning if the operating frequency with your chosen CAS latency exceeds this number. The lowest frequency supported by DDR3 SDRAM devices is 300 MHz.
Maximum memory frequency for CAS latency 6.0			
Maximum memory frequency for CAS latency 7.0			
Maximum memory frequency for CAS latency 8.0			
Maximum memory frequency for CAS latency 9.0			
Maximum memory frequency for CAS latency 10.0			

**Note to Table 2-3:**

(1) The range values depend on the actual memory device used.

**Table 2-4.** DDR3 SDRAM Initialization Options (Part 1 of 3)

Parameter Name	Range	Units	Description
Memory burst length	4, 8, on-the-fly	beats	Sets the number of words read or written per transaction.
Memory burst ordering	Sequential or Interleaved	—	Controls the order in which data is transferred between memory and the FPGA during a read transaction. For more information, refer to the memory device datasheet.
DLL precharge power down	Fast exit or Slow exit	—	Sets the mode register setting to disable ( <b>Slow exit</b> ) or enable ( <b>Fast exit</b> ) the memory DLL when CKE is disabled.

**Table 2-4.** DDR3 SDRAM Initialization Options (Part 2 of 3)

Parameter Name	Range	Units	Description
Enable the DLL in the memory devices	Yes or No	—	Enables the DLL in the memory device when set to <b>Yes</b> . Always enable the DLL in the memory device as Altera does not guarantee any ALTMEMPHY operation when the DLL is off because all timings from the memory devices are invalid if the DLL is off.
ODT $R_{tt}$ nominal value	ODT disable, RZQ/4, RZQ/2, RZQ/6	$\Omega$	RZQ in DDR3 SDRAM interfaces are set to 240 $\Omega$ . Sets the on-die termination (ODT) value to either 60 $\Omega$ ( <b>RZQ/4</b> ), 120 $\Omega$ ( <b>RZQ/2</b> ), or 40 $\Omega$ ( <b>RZQ/6</b> ). Set this to <b>ODT disable</b> if you are not planning to use ODT. For a single-ranked DIMM, set this to <b>RZQ/4</b> .
Dynamic ODT ( $R_{tt\_WR}$ ) value	Dynamic ODT off, RZQ/4, RZQ/2	$\Omega$	RZQ in DDR3 SDRAM interfaces are set to 240 $\Omega$ . Sets the memory ODT value during write operations to 60 $\Omega$ ( <b>RZQ/4</b> ) or 120 $\Omega$ ( <b>RZQ/2</b> ). As ALTMEMPHY only supports single rank DIMMs, you do not need this option (set to <b>Dynamic ODT off</b> ).
Output driver impedance	RZQ/6 (Reserved) or RZQ/7	$\Omega$	RZQ in DDR3 SDRAM interfaces are set to 240 $\Omega$ . Sets the output driver impedance from the memory device. Some devices may not have <b>RZQ/6</b> available as an option. Be sure to check the memory device datasheet before choosing this option.
Memory CAS Latency setting	5.0, 6.0, 7.0, 8.0, 9.0, 10.0	cycles	Sets the delay in clock cycles from the read command to the first output data from the memory.
Memory additive CAS latency setting	Disable, CL – 1, CL – 2	cycles	Allows you to add extra latency in addition to the CAS latency setting.
Memory write CAS latency setting (CWL)	5.0, 6.0, 7.0, 8.0	cycles	Sets the delay in clock cycles from the write command to the first expected data to the memory.
Memory partial array self refresh	Full array, Half array {BA[2:0]=000,001, 010,011}, Quarter array {BA[2:0]=000,001}, Eighth array {BA[2:0]=000}, Three Quarters array {BA[2:0]=010,011, 100,101,110,111}, Half array {BA[2:0]=100,101, 110,111}, Quarter array {BA[2:0]=110, 111}, Eighth array {BA[2:0]=111}	—	Determine whether you want to self-refresh only certain arrays instead of the full array. According to the DDR3 SDRAM specification, data located in the array beyond the specified address range are lost if self refresh is entered when you use this. This option is not supported by the DDR3 SDRAM High-Performance Controller MegaCore function, so set to <b>Full Array</b> if you are using the Altera controller.

**Table 2-4.** DDR3 SDRAM Initialization Options (Part 3 of 3)

Parameter Name	Range	Units	Description
Memory auto self refresh method	Manual SR reference (SRT) or ASR enable (Optional)	—	Sets the auto self refresh method for the memory device. The DDR3 SDRAM High-Performance Controller MegaCore function currently does not support the ASR option that you need for extended temperature memory self-refresh.
Memory self refresh range	Normal or Extended	—	Determines the temperature range for self refresh. You need to also use the optional auto self refresh option when using this option. The Altera controller currently does not support the extended temperature self-refresh operation.

**Table 2-5.** DDR3 SDRAM Timing Parameter Settings (Part 1 of 2) (Note 1)

Parameter Name	Range	Units	Description
Time to hold memory reset before beginning calibration	0–1000000	μs	Minimum time to hold the reset after a power cycle before issuing the MRS commands during the DDR3 SDRAM device initialization process.
$t_{\text{INIT}}$	0.001–1000	μs	Minimum memory initialization time. After reset, the controller does not issue any commands to the memory during this period.
$t_{\text{MRD}}$	2–39	ns	Minimum load mode register command period. The controller waits for this period of time after issuing a load mode register command before issuing any other commands.  $t_{\text{MRD}}$ is specified in ns but in terms of $t_{\text{CK}}$ cycles in Micron's device datasheet. Convert $t_{\text{MRD}}$ to ns by multiplying the number of cycles specified in the datasheet times $t_{\text{CK}}$ , where $t_{\text{CK}}$ is the memory operation frequency and not the memory device's $t_{\text{CK}}$ .
$t_{\text{RAS}}$	8–200	ns	Minimum active to precharge time. The controller waits for this period of time after issuing an active command before issuing a precharge command to the same bank.
$t_{\text{RCD}}$	4–65	ns	Minimum active to read-write time. The controller does not issue read or write commands to a bank during this period of time after issuing an active command.
$t_{\text{RP}}$	4–65	ns	Minimum precharge command period. The controller does not access the bank for this period of time after issuing a precharge command.
$t_{\text{REFI}}$	1–65534	μs	Maximum interval between refresh commands. The controller performs regular refresh at this interval unless user-controlled refresh is turned on.
$t_{\text{RFC}}$	14–1651	ns	Minimum autorefresh command period. The length of time the controller waits before doing anything else after issuing an auto-refresh command.
$t_{\text{WR}}$	4–65	ns	Minimum write recovery time. The controller waits for this period of time after the end of a write transaction before issuing a precharge command.
$t_{\text{WTR}}$	1–6	$t_{\text{CK}}$	Minimum write-to-read command delay. The controller waits for this period of time after the end of a write command before issuing a subsequent read command to the same bank. This timing parameter is specified in clock cycles and the value is rounded off to the next integer.
$t_{\text{AC}}$	0–750	ps	DQ output access time.
$t_{\text{DQSK}}$	50–750	ps	DQS output access time from CK/CK# signals.
$t_{\text{DQSQ}}$	50–500	ps	The maximum DQS to DQ skew; DQS to last DQ valid, per group, per access.

**Table 2-5.** DDR3 SDRAM Timing Parameter Settings (Part 2 of 2) (Note 1)

Parameter Name	Range	Units	Description
$t_{DQSS}$	0–0.3	$t_{CK}$	Positive DQS latching edge to associated clock edge.
$t_{DH}$	10–600	ps	DQ and DM input hold time relative to DQS, which has a derated value depending on the slew rate of the differential DQS and DQ/DM signals. Ensure that you are using the correct number and that the value entered is referenced to $V_{REF}(dc)$ , not $V_{IH}(dc)$ min or $V_{IL}(dc)$ max.
$t_{DS}$	10–600	ps	DQ and DM input setup time relative to DQS, which has a derated value depending on the slew rate of the differential DQS signals and DQ/DM signals. Ensure that you are using the correct number and that the value entered is referenced to $V_{REF}(dc)$ , not $V_{IH}(ac)$ min or $V_{IL}(ac)$ max.
$t_{DSH}$	0.1–0.5	$t_{CK}$	DQS falling edge hold time from CK.
$t_{DSS}$	0.1–0.5	$t_{CK}$	DQS falling edge to CK setup.
$t_{IH}$	50–1000	ps	Address and control input hold time, which has a derated value depending on the slew rate of the CK and CK# clocks and the address and command signals. Ensure that you are using the correct number and that the value entered is referenced to $V_{REF}(dc)$ , not $V_{IH}(dc)$ min or $V_{IL}(dc)$ max.
$t_{IS}$	65–1000	ps	Address and control input setup time, which has a derated value depending on the slew rate of the CK and CK# clocks and the address and command signals. Ensure that you are using the correct number and that the value entered is referenced to $V_{REF}(dc)$ , not $V_{IH}(ac)$ min or $V_{IL}(ac)$ max.
$t_{QHS}$	0–700	ps	The maximum data hold skew factor.
$t_{OH}$	0.1–0.6	$t_{CK}$	DQ output hold time.

**Note to Table 2-5:**

- (1) See the memory device data sheet for the parameter range. Some of the parameters may be listed in a clock cycle ( $t_{CK}$ ) unit. If the MegaWizard Plug-In requires you to enter the value in a time unit (ps or ns), convert the number by multiplying it with the clock period of your interface (and not the maximum clock period listed in the memory data sheet).

### DDR2/DDR SDRAM Preset Editor Page

Figure 2-5 shows the **Preset Editor** page for the ALTMEMPHY variation for DDR2/DDR SDRAM interfaces.

**Figure 2-5.** Preset Editor for DDR2/DDR SDRAM Interfaces

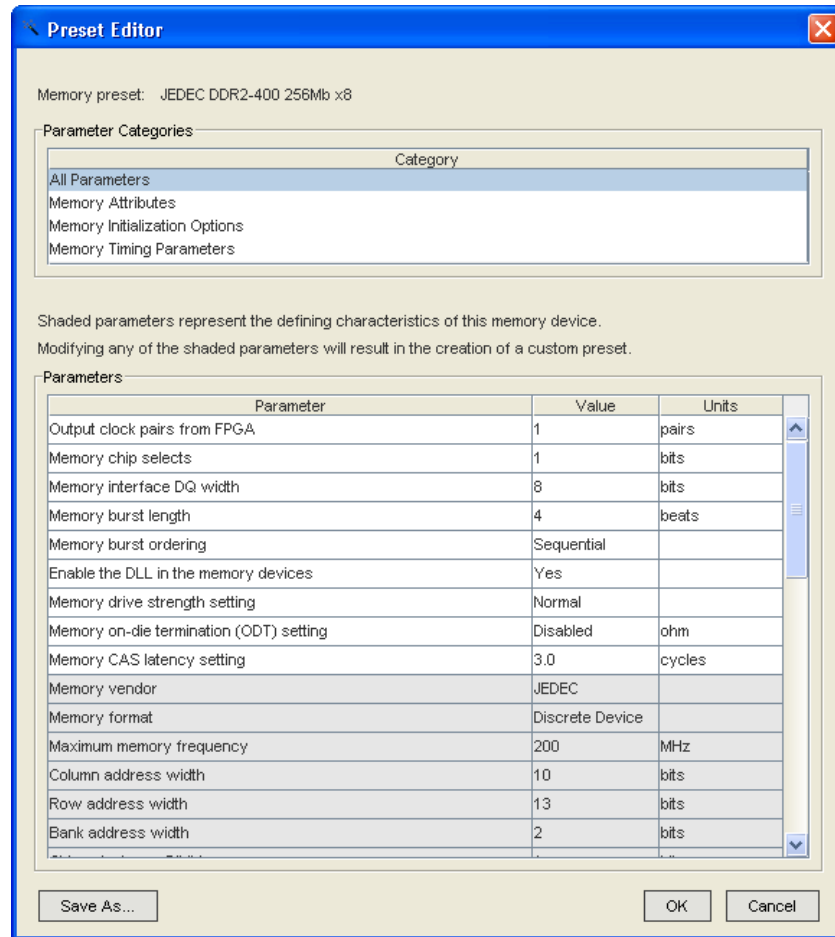


Table 2-6 through Table 2-8 describe the DDR2 SDRAM parameters available for memory attributes, initialization options, and timing parameters. DDR SDRAM has the same parameters, but their value ranges are different than DDR2 SDRAM. Use the MegaWizard Plug-In Manager to determine whether your chosen value is valid.

**Table 2-6.** DDR2 SDRAM Attributes Settings (Part 1 of 2)

Parameter Name	Range (1)	Units	Description
Output clock pairs from FPGA	1–6	pairs	Defines the number of differential clock pairs driven from the FPGA to the memory. More clock pairs reduce the loading of each output when interfacing with multiple devices. Memory clock pins use the signal splitter feature in Arria II GX, Stratix IV and Stratix III devices for differential signaling.
Memory chip selects	1, 2, 4, or 8	bits	Sets the number of chip selects in your memory interface. The depth of your memory in terms of number of chips. You are limited to the range shown as the local side binary encodes the chip select address. You can set this value to the next higher number if the range does not meet your specifications. However, the highest address space of the ALTMEMPHY megafunction is not mapped to any of the actual memory address. The ALTMEMPHY megafunction works with multiple chip selects and calibrates against all chip select <code>mem_cs_n</code> signals.
Memory interface DQ width	4–288	bits	Defines the total number of DQ pins on the memory interface. If you are interfacing with multiple devices, multiply the number of devices with the number of DQ pins per device. Even though the GUI allows you to choose 288-bit DQ width, the interface data width is limited by the number of pins on the device. For best performance, have the whole interface on one side of the device.
Memory vendor	JEDEC, Micron, Qimonda, Samsung, Hynix, Elpida, Nanya, other	—	Describes the name of the memory vendor for all supported memory standards.
Memory format	Discrete Device, Unbuffered DIMM, Registered DIMM	—	Specifies whether you are interfacing with devices or modules. SODIMM is supported under unbuffered or registered DIMMs.
Maximum memory frequency	See the memory device datasheet	MHz	Sets the maximum frequency supported by the memory.
Column address width	9–11	bits	Defines the number of column address bits for your interface.
Row address width	13–16	bits	Defines the number of row address bits for your interface.
Bank address width	2 or 3	bits	Defines the number of bank address bits for your interface.
Chip selects per DIMM	1 or 2	bits	Defines the number of chip selects on each DIMM in your interface.

**Table 2-6.** DDR2 SDRAM Attributes Settings (Part 2 of 2)

Parameter Name	Range (1)	Units	Description
DQ bits per DQS bit	4 or 8	bits	Defines the number of data (DQ) bits for each data strobe (DQS) pin.
Precharge address bit	8 or 10	bits	Selects the bit of the address bus to use as the precharge address bit.
Drive DM pins from FPGA	Yes or No	—	Specifies whether you are using DM pins for write operation. Altera devices do not support DM pins in ×4 mode.
Maximum memory frequency for CAS latency 3.0	80–533	MHz	See (2).
Maximum memory frequency for CAS latency 4.0	80–533	MHz	See (2).
Maximum memory frequency for CAS latency 5.0	80–533	MHz	See (2).
Maximum memory frequency for CAS latency 6.0	80–533	MHz	See (2).

**Note to Table 2-6:**

- (1) The range values depend on the actual memory device used.
- (2) Frequency limits from the memory data sheet per given CAS latency. The ALTMEMPHY MegaWizard Plug-In Manager generates a warning if the operating frequency with your chosen CAS latency exceeds this number.

**Table 2-7.** DDR2/DDR SDRAM Initialization Options

Parameter Name	Range	Units	Description
Memory burst length	4 or 8 (DDR2 SDRAM) 2, 4, or 8 (DDR SDRAM)	beats	Sets the number of words read or written per transaction. Memory burst length of four equates to local burst length of one in half-rate designs and to local burst length of two in full-rate designs.
Memory burst ordering	Sequential or Interleaved	—	Controls the order in which data is transferred between memory and the FPGA during a read transaction. Refer to the memory device datasheet for more information.
Enable the DLL in the memory devices	Yes or No	—	Enables the DLL in the memory device when set to <b>Yes</b> . You must always enable the DLL in the memory device as Altera does not guarantee any ALTMEMPHY operation when the DLL is off because all timings from the memory devices are invalid when the DLL is off.
Memory drive strength setting	Normal or Reduced	—	Controls the drive strength of the memory device's output buffers. Reduced drive strength is not supported on all memory devices. The default option is normal.
Memory ODT setting	Disabled, 50, 75, 150	Ohms	Sets the memory ODT value. Not available in DDR SDRAM interfaces.
Memory CAS latency setting	3, 4, 5, 6 (DDR2 SDRAM) 2, 2.5, 3 (DDR SDRAM)	Cycles	Sets the delay in clock cycles from the read command to the first output data from the memory.

**Table 2-8.** DDR2/DDR SDRAM Timing Parameter Settings (*Note 1*) (Part 1 of 2)

Parameter Name	Range	Units	Description
$t_{INIT}$	0.001–1000	$\mu s$	Minimum memory initialization time. After reset, the controller does not issue any commands to the memory during this period.
$t_{MRD}$	2–39	ns	Minimum load mode register command period. The controller waits for this period of time after issuing a load mode register command before issuing any other commands. $t_{MRD}$ is specified in ns in the DDR2 SDRAM High-Performance Controller and in terms of $t_{CK}$ cycles in Micron's device datasheet. You need to convert $t_{MRD}$ to ns by multiplying the number of cycles specified in the datasheet times $t_{CK}$ . Where $t_{CK}$ is the memory operation frequency and not the memory device's $t_{CK}$ .
$t_{RAS}$	8–200	ns	Minimum active to precharge time. The controller waits for this period of time after issuing an active command before issuing a precharge command to the same bank.
$t_{RCD}$	4–65	ns	Minimum active to read-write time. The controller does not issue read or write commands to a bank during this period of time after issuing an active command.
$t_{RP}$	4–65	ns	Minimum precharge command period. The controller does not access the bank for this period of time after issuing a precharge command.
$t_{REFI}$	1–65534	$\mu s$	Maximum interval between refresh commands. The controller performs regular refresh at this interval unless user-controlled refresh is turned on.
$t_{RFC}$	14–1651	ns	Minimum autorefresh command period. The length of time the controller waits before doing anything else after issuing an auto-refresh command.
$t_{WR}$	4–65	ns	Minimum write recovery time. The controller waits for this period of time after the end of a write transaction before issuing a precharge command.
$t_{WTR}$	1–3	$t_{CK}$	Minimum write-to-read command delay. The controller waits for this period of time after the end of a write command before issuing a subsequent read command to the same bank. This timing parameter is specified in clock cycles and the value is rounded off to the next integer.
$t_{AC}$	300–750	ps	DQ output access time from CK/CK# signals.
$t_{DQSQ}$	100–750	ps	DQS output access time from CK/CK# signals.
$t_{DQSQ}$	100–500	ps	The maximum DQS to DQ skew; DQS to last DQ valid, per group, per access.
$t_{DQSS}$	0–0.3	$t_{CK}$	Positive DQS latching edge to associated clock edge.
$t_{DS}$	10–600	ps	DQ and DM input setup time relative to DQS, which has a derated value depending on the slew rate of the DQS (for both DDR2 and DDR SDRAM interfaces) and whether DQS is single-ended or differential (for DDR2 SDRAM interfaces). Ensure that you are using the correct number and that the value entered is referenced to $V_{REF}(dc)$ , not $V_{IH}(ac)$ min or $V_{IL}(ac)$ max.
$t_{DH}$	10–600	ps	DQ and DM input hold time relative to DQS, which has a derated value depending on the slew rate of the DQS (for both DDR2 and DDR SDRAM interfaces) and whether DQS is single-ended or differential (for DDR2 SDRAM interfaces). Ensure that you are using the correct number and that the value entered is referenced to $V_{REF}(dc)$ , not $V_{IH}(dc)$ min or $V_{IL}(dc)$ max.
$t_{DSH}$	0.1–0.5	$t_{CK}$	DQS falling edge hold time from CK.
$t_{DSS}$	0.1–0.5	$t_{CK}$	DQS falling edge to CK setup.



**Table 2-8.** DDR2/DDR SDRAM Timing Parameter Settings (Note 1) (Part 2 of 2)

Parameter Name	Range	Units	Description
$t_{IH}$	100–1000	ps	Address and control input hold time, which has a derated value depending on the slew rate of the CK and CK# clocks and the address and command signals. Ensure that you are using the correct number and that the value entered is referenced to $V_{REF}(dc)$ , not $V_{IH}(dc)$ min or $V_{IL}(dc)$ max.
$t_{IS}$	100–1000	ps	Address and control input setup time, which has a derated value depending on the slew rate of the CK and CK# clocks and the address and command signals. Ensure that you are using the correct number and that the value entered is referenced to $V_{REF}(dc)$ , not $V_{IH}(ac)$ min or $V_{IL}(ac)$ max.
$t_{QHS}$	100–700	ps	The maximum data hold skew factor.

**Note to Table 2-8:**

- (1) See the memory device data sheet for the parameter range. Some of the parameters may be listed in a clock cycle ( $t_{CK}$ ) unit. If the MegaWizard Plug-In Manager requires you to enter the value in a time unit (ps or ns), convert the number by multiplying it with the clock period of your interface (and not the maximum clock period listed in the memory data sheet).

**QDRII+/QDRII SRAM Preset Editor Page**

Figure 2-6 shows the **Preset Editor** page for the ALTMEMPHY variation for QDRII+/QDRII SRAM interfaces.

**Figure 2-6.** Preset Editor for QDRII+/QDRII SRAM Interfaces

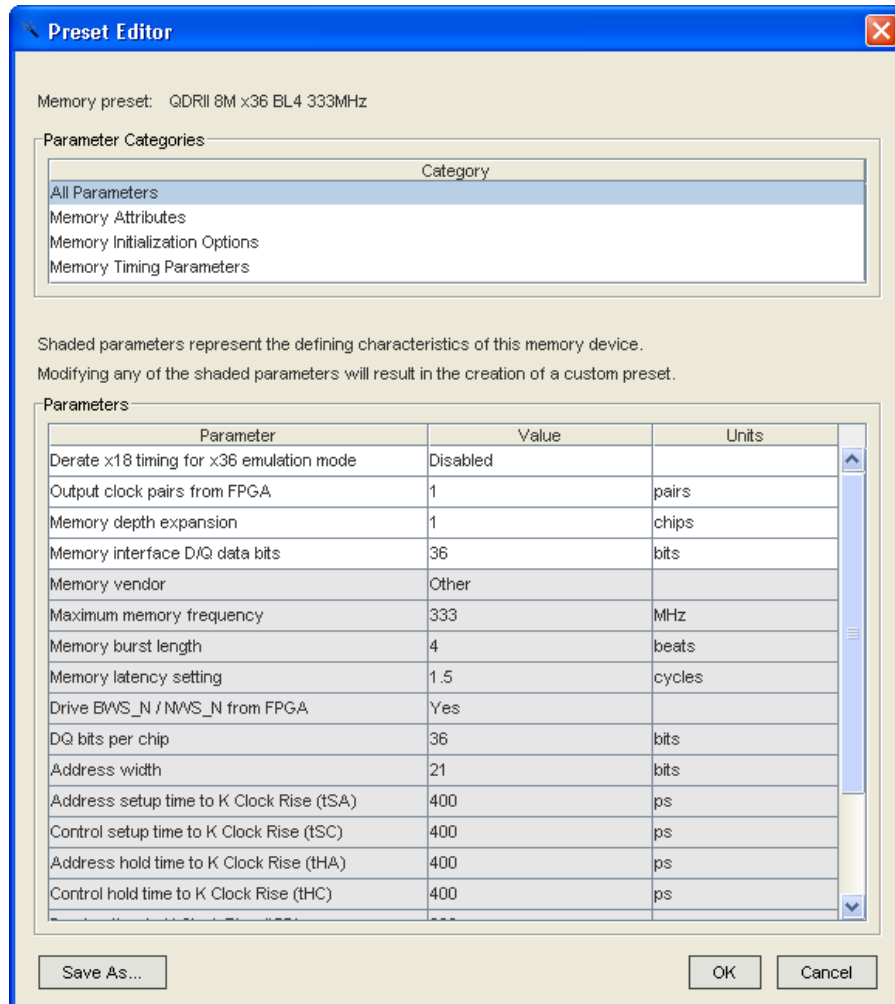


Table 2-9 through Table 2-11 describe the QDRII+/QDRII SRAM parameters available for memory attributes, initialization options, and timing parameters. QDRII+ SRAM devices have the same parameters as QDRII SRAM devices, but their value ranges can differ. Confirm that the value that you have chosen is valid in the ALTMEMPHY MegaWizard Plug-In.

**Table 2-9.** QDRII+/QDRII SRAM Attribute Settings

Parameter Name	Range (1)	Units	Description
De-rate $\times 18$ timing for emulation $\times 36$ mode	Enabled or Disabled	—	Allows the ALTMEMPHY megafunction to derate the timing calculation when creating $\times 36$ QDRII+/QDRII SRAM interfaces by using two $\times 18$ DQS/DQ groups. For more information on $\times 36$ emulation, refer to “ <a href="#">Creating an Emulated <math>\times 36</math> QDRII+/QDRII SRAM ALTMEMPHY Variation</a> ” on page 2-37.
Output clock pairs from FPGA	1-16	pairs	Selects the number of differential clock pairs driven from the FPGA to the memory. More clock pairs reduce the loading of each output when interfacing with multiple memory devices. Memory clock pins use the signal splitter feature in Stratix III and Stratix IV devices for differential signaling.
Memory depth expansion	1-2	chips	Picks the number of chip selects of memory supported. This option is for memory depth expansion.
Memory interface D/Q data bits	8-288	bits	Defines the width of external memory read and write data bus. Multiply the number of devices with the number of DQ pins per device when you create width-expanded memory interfaces. Even though the GUI allows you to choose 288-bit DQ width, the interface data width is limited by the number of pins on the device. For best performance, have the whole interface on one side of the device.
Memory vendor	Others	—	Displays the name of the memory vendor for all supported memory standards. The ALTMEMPHY megafunction only has generic QDRII+/QDRII SRAM data sheet information listed under vendor as Other.
Maximum memory frequency	See the memory device data sheet	MHz	Defines the maximum frequency supported by the memory.
Drive BWS_N/NWS_N from FPGA	Yes or No	—	Enables the use of the write select pins for write operations when set to <b>Yes</b> .
DQ bits per chip	8, 9, 18, 36	bits	Defines the width of D and Q data bus on each QDRII SRAM chip.
Address width	15-25	bits	Sets the number of address bits.
I/O standard	QDRII+ SRAM: 1.5 V HSTL Class I; QDRII SRAM: 1.8 V HSTL Class I or 1.5 V HSTL Class I	—	Selects the I/O standard to be applied to the memory interface pins.

**Note to Table 2-9:**

(1) The range values depend on the actual memory device used.

**Table 2-10.** QDRII+/QDRII SRAM Initialization Options

Parameter Name	Range	Units	Description
Memory burst length	4	beats	Sets the memory burst length for the interface. As the QDRII+/QDRII SRAM ALTMEMPHY megafunction only supports half-rate designs, only a memory burst length of four is supported, which equates to a local burst length of one.
Memory latency setting	1.5 (QDRII SRAM) or 2.5 (QDRII+ SRAM)	Cycles	Sets the memory latency. Altera devices only support latency of 2.5 for QDRII+ SRAM and 1.5 for QDRII SRAM. QDRII+ SRAM with latency of 2.0 is not supported with Altera devices, even though the ALTMEMPHY MegaWizard Plug-In Manager shows this as an option.

**Table 2-11.** QDRII+/QDRII SRAM Timing Parameter Settings

Parameter Name	Range	Units	Description
$t_{SA}$	200–500	ps	Address setup time to K clock rise.
$t_{SC}$	200–500	ps	Control setup time to K clock rise.
$t_{HA}$	200–500	ps	Address hold time to K clock rise.
$t_{HC}$	200–500	ps	Control hold time after K clock rise.
$t_{SD}$	200–500	ps	D setup time to K clock rise.
$t_{HD}$	200–500	ps	D hold time to K clock rise.
$t_{CQHqV}$	200–500	ps	Echo clock high to data valid.
$t_{CQHqX}$	200–500	ps	Echo clock high to data invalid.
$t_{CQHcQnH}$ (1)	0–2,000	ps	Echo clock high to inverted echo clock high.
$t_{CQH}$ (1)	0–2,000	ps	Echo clock high.

**Note to Table 2-11:**

(1) This parameter is available for QDRII+ SRAM interfaces only.

**PHY Settings**

Click **Next** or the **PHY Settings** tab (Figure 2-7) to set the options described in Table 2-12. The options are editable if they apply to the Altera device that you have chosen for your interface. Otherwise, the options are disabled.

Figure 2-7. ALTMEMPHY PHY Parameter Settings Page

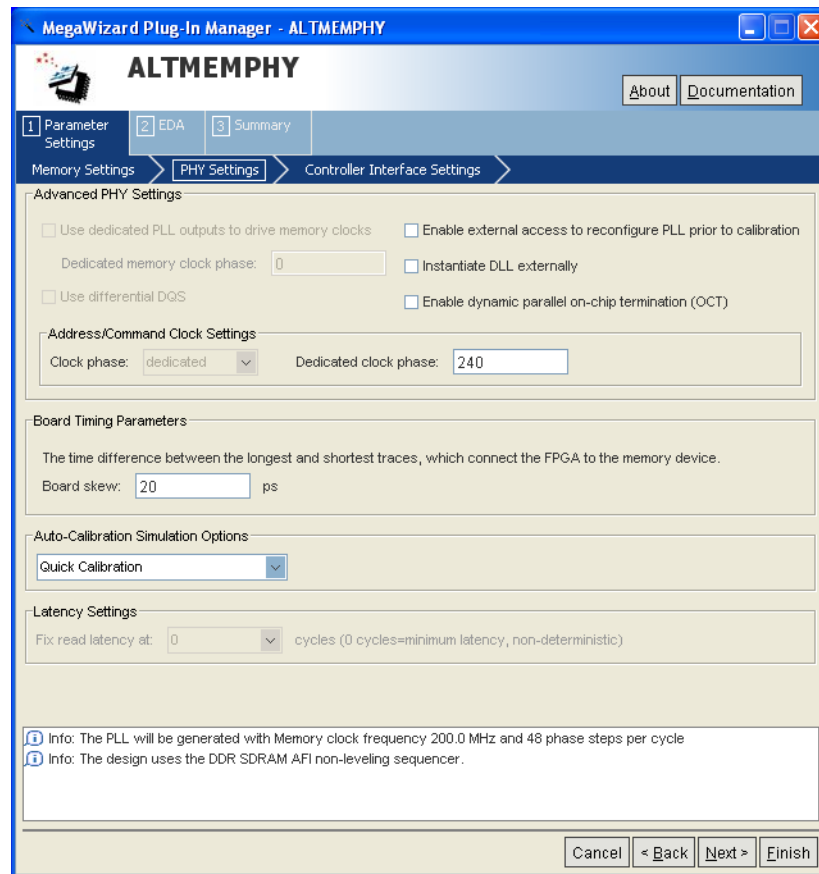


Table 2-12. ALTMEMPHY PHY Settings (Part 1 of 3)

Parameter Name	Applicable Device Families	Description
Use dedicated PLL outputs to drive memory clocks	HardCopy II and Stratix II (prototyping for HardCopy II)	Turn on to use dedicated PLL outputs to generate the external memory clocks, which is required for HardCopy II ASICs and their Stratix II FPGA prototypes. When turned off, the DDIO output registers generate the clock outputs.  When you use the DDIO output registers for the memory clock, both the memory clock and the DQS signals are well aligned and easily meets the tDQSS specification. However, when the dedicated clock outputs are for the memory clock, the memory clock and the DQS signals are not aligned properly and requires a positive phase offset from the PLL to align the signals together.
Dedicated memory clock phase	HardCopy II and Stratix II (prototyping for HardCopy II)	The required phase shift to align the CK/CK# signals with DQS/DQS# signals when using dedicated PLL outputs to drive memory clocks.
Use differential DQS	Arria II GX, Stratix IV, and Stratix III	Enable this feature for better signal integrity. Recommended for operation at 333 MHz or higher. An option for DDR2 SDRAM only, as DDR SDRAM does not support differential DQS and DDR3 SDRAM requires differential DQS.

**Table 2-12.** ALTMEMPHY PHY Settings (Part 2 of 3)

Parameter Name	Applicable Device Families	Description
Enable external access to reconfigure PLL prior to calibration	HardCopy II and Stratix II (prototyping for HardCopy II)	<p>When enabling this option for Stratix II and HardCopy II devices, the inputs to the ALTPLL_RECONFIG megafunction are brought to the top level for debugging purposes.</p> <p>This option allows you to reconfigure the PLL before calibration to adjust, if necessary, the phase of the memory clock (<code>mem_clk_2x</code>) before the start of the calibration of the resynchronization clock on the read side. The calibration of the resynchronization clock on the read side depends on the phase of the memory clock on the write side.</p>
Instantiate DLL externally	All supported device families, except for Cyclone III devices	<p>Use this option with Stratix III, Stratix IV, or HardCopy III, or HardCopy IV devices, if you want to apply a non-standard phase shift to the DQS capture clock. The ALTMEMPHY DLL offsetting I/O can then be connected to the external DLL and the Offset Control Block.</p> <p>As Cyclone III devices do not have DLLs, this feature is not supported.</p>
Enable dynamic parallel on-chip termination	Stratix IV and Stratix III	<p>This option provides I/O impedance matching and termination capabilities. The ALTMEMPHY megafunction enables parallel termination during reads and series termination during writes with this option checked. Only applicable for DDR3, DDR2, and DDR SDRAM interfaces where DQ and DQS are bidirectional. Using the dynamic termination requires that you use the OCT calibration block, which may impose a restriction on your DQS/DQ pin placements depending on your RUP/RDN pin locations.</p> <p>Although DDR SDRAM does not support ODT, dynamic OCT is still supported in Altera FPGAs.</p> <p>For more information, refer to <a href="#">“Dynamic OCT Support” on page 4-27</a> or refer to either the <i>External Memory Interfaces in Stratix III Devices</i> chapter in volume 1 of the <i>Stratix III Device Handbook</i> or the <i>External Memory Interfaces in Stratix IV Devices</i> chapter in volume 1 of the <i>Stratix IV Device Handbook</i>.</p>
Clock phase	Arria GX, Arria II GX, Cyclone III, HardCopy II, Stratix II, and Stratix II GX	<p>Adjusting the address and command phase can improve the address and command setup and hold margins at the memory device to compensate for the propagation delays that vary with different loadings. You have a choice of 0°, 90°, 180°, and 270°, based on the rising and falling edge of the <code>phy_clk</code> and <code>write_clk</code> signals. In Stratix IV and Stratix III devices, the clock phase is set to <b>dedicated</b>.</p>
Dedicated clock phase	Stratix IV and Stratix III	<p>When you use a dedicated PLL output for address and command, you can choose any legal PLL phase shift to improve setup and hold for the address and command signals. You can set this value to between 180° and 359° (the default is 240°). However, generally PHY timing requires a value of greater than 240° for half-rate designs (270° for full-rate designs).</p>

**Table 2-12.** ALTMEMPHY PHY Settings (Part 3 of 3)

Parameter Name	Applicable Device Families	Description
Board skew	All supported device families	Maximum skew across any two memory interface signals for the whole interface from the FPGA to the memory (either a component or a DIMM). This parameter includes all types of signals (data, strobe, clock, address, and command signals). You need to input the worst-case skew, whether it is within a DQS/DQ group, or across all groups, or across the address and command and clocks signals. This parameter generates the timing constraints in the <b>.sdc</b> file.
Autocalibration simulation options	All supported device families	Choose between <b>Full Calibration</b> (long simulation time), <b>Quick Calibration</b> , or <b>Skip Calibration</b> (DDR3 SDRAM with leveling only). For more information, see <a href="#">“Simulating your Design” on page 2-51</a> .
Fix read latency at	QDRII+/QDRII SRAM interfaces in Stratix IV and Stratix III devices	You can ask the ALTMEMPHY megafunction for a specific read latency of your system (if you choose any numbers other than 0). If you chose 0, the ALTMEMPHY calibration determines the read latency. If you want to set a fixed read latency for your system, choose from one of the following: 15, 16, 17, 18, 19, 20, 21, 22 to pick read latency in number of PHY clock cycles. If the PHY cannot achieve the latency you have chosen, the sequencer creates a flag and uses the best-case latency found. For more information, refer to <a href="#">“QDRII+/QDRII SRAM Deterministic Latency” on page 2-23</a> .

### QDRII+/QDRII SRAM Deterministic Latency

There is a setting in the **PHY Settings** tab for you to choose between:

- The minimum read latency achievable equal to the worst-case latency measured out of all of the DQS groups, which is referred to as non-deterministic latency, when the option is set to **0**.
- A user-defined predetermined latency from 15 to 22 half-rate PHY clock cycles, irrespective of the memory read latency, which is referred to as deterministic latency.

With non-deterministic latency, the sequencer sets up the system with a latency value equal to the worst-case latency found for any of the DQS groups. However, the latency value can change from one board to another—this value is not known until calibration has completed.

Specify a read latency value that must consistently be the same from one board to another with deterministic latency. An exception is if the latency you requested is too low and cannot be achieved, then the system defaults to non-deterministic mode and an output signal `p_user_defined_latency_ok` is deasserted to show that the deterministic latency option has failed.

The non-deterministic latency can result in a lower read latency than using deterministic. For example, for a given system with non-deterministic latency, a read latency value of 13 can be achieved. For the same system with deterministic latency, the minimum latency selectable in the MegaWizard Plug-In is 15.

## Controller Interface Settings

Figure 2-8 shows the **Controller Interface Settings** tab. This tab allows you to choose between the native interface and the default Avalon® Memory-Mapped (Avalon-MM) interface for your local interface as required by the ALTMEMPHY megafunction for DDR2/DDR SDRAM. The options are disabled when you are creating an ALTMEMPHY megafunction for DDR3 SDRAM or QDRII+/QDRII SRAM interface, or if you select **AFI** for the controller to PHY interface protocol. The Avalon-MM interface is the only local interface supported in these variations.



Altera recommends that you use the AFI for new designs; only use the non-AFI for existing designs.

**Figure 2-8.** Controller Interface Settings



Native interface is a superset of the Avalon-MM interface, containing the following additional signals in addition to the Avalon-MM interface signals:

- `local_rdvalid_in_n`
- `local_init_done`
- `local_refresh_req`
- `local_refresh_ack`
- `local_wdata_req`



These signals provide extra information and control that is not possible in the Avalon-MM bus protocol.


The other difference between the native and the Avalon-MM local interface is in the write transaction. In an Avalon-MM interface, the write data is presented along with the write request. In native local interfaces, the write data (and byte enables) are presented in the clock cycle after the `local_wdata_req` signal is asserted. Avalon-MM interfaces do not use the `local_wdata_req` signal.

 There is no difference in latency between the native and Avalon-MM interfaces.

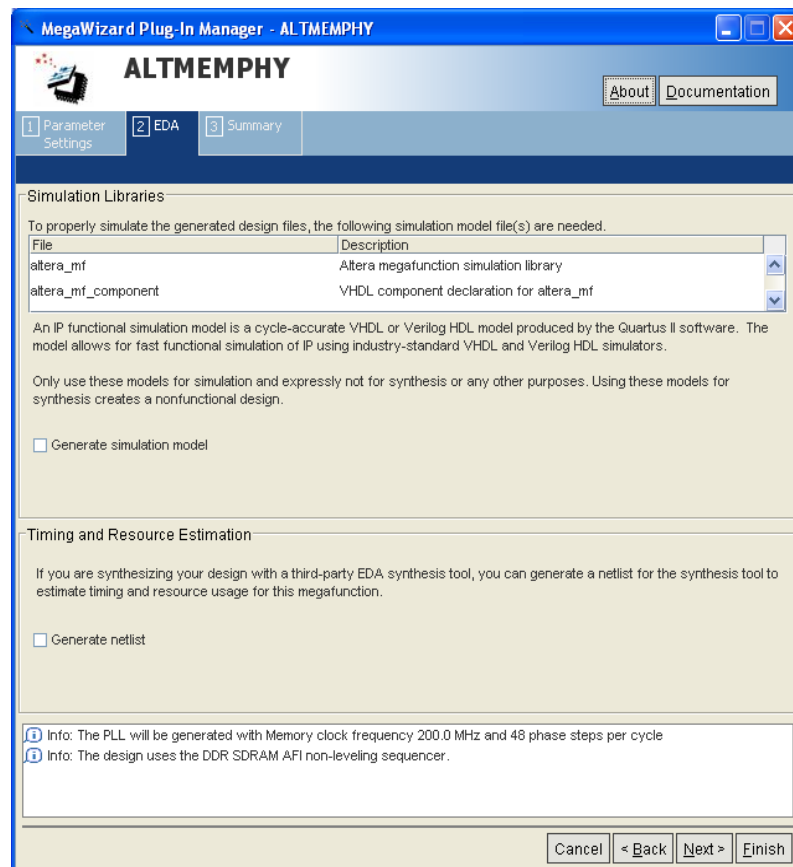
 For example waveforms on native and Avalon-MM local interfaces, refer to the *DDR and DDR2 SDRAM High-Performance Controller User Guide*.

## ALTMEMPHY EDA Page

Click **Next** or click the **EDA** tab (Figure 2-9) to set your **Simulation Model** settings. Turn on **Generate Simulation Model** if you want to generate a simulation model for the encrypted sequencer module of the ALTMEMPHY variation.

 The ALTMEMPHY megafunction only supports functional simulation. You cannot perform timing or gate-level simulation when using ALTMEMPHY megafunction.

**Figure 2-9.** ALTMEMPHY Simulation Model Generation Page



If you turn on **Generate netlist** to generate a synthesis area and timing estimation netlist in the MegaWizard Plug-In, the wizard generates an additional netlist file (`<variation_name>_gb.v`). The netlist file is a representation of the customized logic used in the Quartus II software. The file provides the connectivity of the architectural elements in the megafunction but may not represent true functionality. This information enables certain third-party synthesis tools to better report area and timing estimates. In addition, synthesis tools can use the timing information to focus timing-driven optimizations and improve the quality of results.

You need to generate your simulation model in the same language that you are using to generate your megafunction variation. The generated simulation model is called `<project_dir>\<variation_name>_alt_mem_phy_seq_wrapper.vo/.vho`, and is used during the RTL NativeLink simulation. For more information, refer to [“Simulating your Design” on page 2-51](#).



When targeting a VHDL simulation model, the ALTMEMPHY MegaWizard Plug-In still generates the `<variation_name>_alt_mem_phy.v` file for the Quartus II synthesis. Do not use this file for simulation. Use the `<variation_name>.vho` file for simulation instead.



Use the simulation model output files for simulation only. Using these files for synthesis creates a nonfunctional design.

Simulating the ALTMEMPHY megafunction requires simulating the calibration process also, which may take a significant time, unless you chose **Skip Calibration**, which is available for DDR3 SDRAM interfaces.

Click **Next** to go to the **Summary** page or click the **Summary** tab.

## ALTMEMPHY Summary Page

On the **Summary** page ([Figure 2-10](#)) of the MegaWizard Plug-In, specify the files you wish to generate for your custom megafunction. The grayed-out boxes indicate files that always generate; the other files are optional and generate only if selected (indicated by a green check mark). [Table 2-13](#) lists the files listed on this page.

Figure 2-10. ALTMEMPHY Summary Page

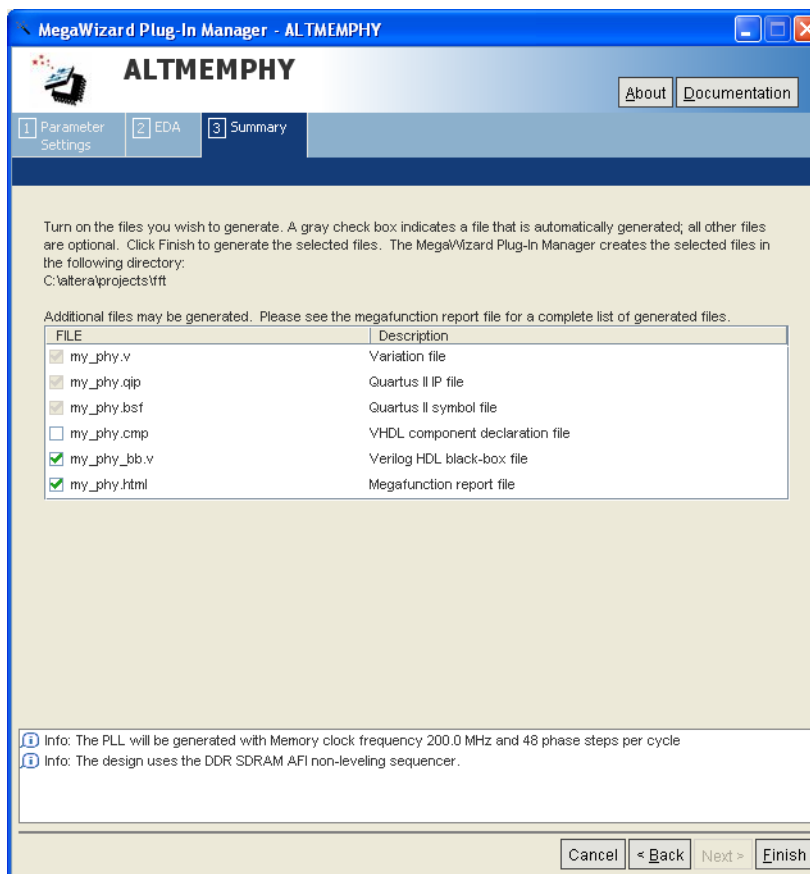



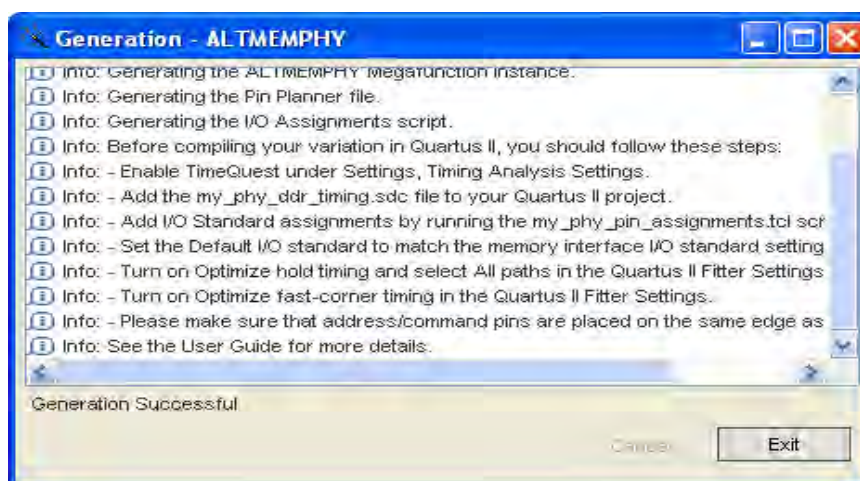
Table 2-13. ALTMEMPHY Summary Page

File Name	Description	Usage
<variation_name>.v or .vhd	Variation file	The top-level file of your megafunction variation.
<variation_name>.qip	Quartus II IP file	Contains names of the files for your megafunction variation to be added to your Quartus II project. This file is generated by the MegaWizard Plug-In for all megafunctions in the Quartus II software.
<variation_name>.bsf	Quartus II symbol file	Use if you are using a block description file (.bdf) to instantiate the megafunction. You can also create this file using <b>Create/Update</b> under the Quartus II software <b>File</b> menu.
<variation_name>.cmp	VHDL component declaration file	Use if you are using VHDL to instantiate the megafunction. You can also create this file using <b>Create/Update</b> under the Quartus II software <b>File</b> menu.
<variation_name>.bb.v	Verilog HDL black-box file	Use if you are using Verilog HDL to instantiate the megafunction. You can also create this file using <b>Create/Update</b> under the Quartus II software <b>File</b> menu.
<variation_name>.html	Megafunction report file	Lists the variation name, HDL language used, directory used, files generated, and input and output ports of the interface.

Click **Finish** to generate the megafunction variation files. [Figure 2-11](#) shows the ALTMEMPHY megafunction **Generation** window. This window also shows the settings that you must apply to change in your Quartus II project before compiling the ALTMEMPHY megafunction design. Click **Exit** to close the window, and click **Yes** on the **Quartus II IP Files** message.

 The Quartus II IP File (**.qip**) is a file generated by the MegaWizard interface that contains information about a generated IP core. You are prompted to add this **.qip** file to the current Quartus II project at the time of file generation. In most cases, the **.qip** file contains all of the necessary assignments and information required to process the core or system in the Quartus II compiler. Generally, a single **.qip** file is generated for each megafunction.

**Figure 2-11.** ALTMEMPHY Megafunction Generation Window



## Generated Files

Clicking **Finish** on the ALTMEMPHY MegaWizard Plug-In generates the RTL files and the simulation files (if you choose to generate the simulation model), pin assignment, timing constraints, and timing report script files required for the interface.

[Table 2-14](#) shows all the files that the ALTMEMPHY MegaWizard Plug-In generates.

**Table 2-14.** Generated Files (Part 1 of 3)

File Name	Description
<b>alt_mem_phy_defines.v</b>	Contains constants used in the interface. This file is always in Verilog HDL regardless of the language you chose in the MegaWizard Plug-In Manager.
<b>&lt;variation_name&gt;.html</b>	Lists the top-level files created and ports used in the megafunction.
<b>&lt;variation_name&gt;.ppf</b>	Pin planner file for your ALTMEMPHY variation.
<b>&lt;variation_name&gt;.qip</b>	Quartus II IP file for your ALTMEMPHY variation, containing the files associated with this megafunction.

**Table 2-14.** Generated Files (Part 2 of 3)

File Name	Description
<code>&lt;variation_name&gt;.v/.vhd</code>	Top-level file of your ALTMEMPHY variation, generated based on the language you chose in the MegaWizard Plug-In.
<code>&lt;variation_name&gt;.vho</code>	Contains functional simulation model for VHDL only.
<code>&lt;variation_name&gt;_alt_mem_phy_delay.vhd</code>	Includes a delay module for simulation. This file is only generated if you choose VHDL as the language of your MegaWizard Plug-In output files.
<code>&lt;variation_name&gt;_alt_mem_phy_dq_dqs.vhd</code> or <code>.v</code>	Generated file that contains DQ/DQS I/O atoms interconnects and instance. Arria II GX devices only.
<code>&lt;variation_name&gt;_alt_mem_phy_dq_dqs_clearbox.txt</code>	Specification file that generates the <code>&lt;variation_name&gt;_alt_mem_phy_dq_dqs</code> file using the clearbox flow. Arria II GX devices only.
<code>&lt;variation_name&gt;_alt_mem_phy_pll.qip</code>	Quartus II IP file for the PLL that your ALTMEMPHY variation uses that contains the files associated with this megafunction.
<code>&lt;variation_name&gt;_alt_mem_phy_pll.v/.vhd</code>	The PLL megafunction file for your ALTMEMPHY variation, generated based on the language you chose in the MegaWizard Plug-In.
<code>&lt;variation_name&gt;_alt_mem_phy_pll_bb.v/.cmp</code>	Black box file for the PLL used in your ALTMEMPHY variation. Typically unused.
<code>&lt;variation_name&gt;_alt_mem_phy_reconfig.qip</code>	Quartus II IP file for the PLL reconfiguration block. Only generated when targeting Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.
<code>&lt;variation_name&gt;_alt_mem_phy_reconfig.v/.vhd</code>	PLL reconfiguration block module. Only generated when targeting Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.
<code>&lt;variation_name&gt;_alt_mem_phy_reconfig_bb.v/cmp</code>	Blackbox file for the PLL reconfiguration block. Only generated when targeting Arria GX, HardCopy II, Stratix II, and Stratix II GX devices.
<code>&lt;variation_name&gt;_alt_mem_phy_seq.vhd</code> or <code>&lt;variation_name&gt;_alt_mem_phy_qdrii_sequencer.vhd</code>	Contains the sequencer used during calibration. This file is encrypted and is always in VHDL language regardless of the language you chose in the MegaWizard Plug-In. In QDR11+/QDR11 SRAM interfaces, the file name includes prefix <b>qdrii</b> .
<code>&lt;variation_name&gt;_alt_mem_phy_seq_wrapper.v/.vhd</code> or <code>&lt;variation_name&gt;_alt_mem_phy_qdrii_sequencer_wrapper.v/.vhd</code>	A wrapper file, for compilation only, that calls the sequencer file, created based on the language you chose in the MegaWizard Plug-In. In QDR11+/QDR11 SRAM interfaces, the file name includes the text <b>qdrii</b> .
<code>&lt;variation_name&gt;_alt_mem_phy_seq_wrapper.vo/.vho</code> or <code>&lt;variation_name&gt;_alt_mem_phy_qdrii_sequencer_wrapper.vo/.vho</code>	A wrapper file, for simulation only, that calls the sequencer file, created based on the language you chose in the MegaWizard Plug-In. In QDR11+/QDR11 SRAM interfaces, the file name includes the text <b>qdrii</b> .

**Table 2-14.** Generated Files (Part 3 of 3)

File Name	Description
<code>&lt;variation_name&gt;_alt_mem_phy.v</code>	Contains all modules of the ALTMEMPHY variation except for the sequencer. This file is always in Verilog HDL language regardless of the language you chose in the MegaWizard Plug-In. The DDR3 SDRAM sequencer is included in the <code>&lt;variation_name&gt;_alt_mem_phy_seq.vhd</code> file.
<code>&lt;variation_name&gt;_bb.v/.cmp</code>	Black box file for your ALTMEMPHY variation, depending whether you are using Verilog HDL or VHDL language.
<code>&lt;variation_name&gt;_ddr_pins.tcl</code>	Contains procedures used in the <code>&lt;variation_name&gt;_ddr_timing.sdc</code> and <code>&lt;variation_name&gt;_report_timing.tcl</code> files.
<code>&lt;variation_name&gt;_ddr_timing.sdc</code>	Contains timing constraints for your ALTMEMPHY variation.
<code>&lt;variation_name&gt;_pin_assignments.tcl</code>	Contains I/O standard, drive strength, output enable grouping, DQ/DQS grouping, and termination assignments for your ALTMEMPHY variation. If your top-level design pin names do not match the default pin names or a prefixed version, edit the assignments in this file.
<code>&lt;variation_name&gt;_report_timing.tcl</code>	Script that reports timing for your ALTMEMPHY variation during compilation.

Table 2-15 shows the modules that are instantiated in the `<variation_name>_alt_mem_phy.v/vhd` file. A particular ALTMEMPHY variation may or may not use any of the modules, depending on the memory standard that you specify.

**Table 2-15.** Modules in `<variation_name>_alt_mem_phy.v` File (Part 1 of 2)

Module Name	Usage	Description
<code>&lt;variation_name&gt;_alt_mem_phy_addr_cmd</code>	All ALTMEMPHY variations	Generates the address and command structures.
<code>&lt;variation_name&gt;_alt_mem_phy_clk_reset</code>	All ALTMEMPHY variations	Instantiates PLL, DLL, and reset logic.
<code>&lt;variation_name&gt;_alt_mem_phy_dp_io</code>	All ALTMEMPHY variations	Generates the DQ, DQS, DM, and QVLD I/O pins.
<code>&lt;variation_name&gt;_alt_mem_phy_mimic</code>	DDR3/DDR2/DDR SDRAM ALTMEMPHY variation	Creates the VT tracking mechanism for DDR3, DDR2, and DDR SDRAM PHYs.
<code>&lt;variation_name&gt;_alt_mem_phy_oct_delay</code>	DDR3/DDR2/DDR SDRAM ALTMEMPHY variation when dynamic OCT is enabled.	Generates the proper delay and duration for the OCT signals.
<code>&lt;variation_name&gt;_alt_mem_phy_postamble</code>	DDR3/DDR2/DDR SDRAM ALTMEMPHY variations	Generates the postamble enable and disable scheme for DDR3, DDR, and DDR SDRAM PHYs.
<code>&lt;variation_name&gt;_alt_mem_phy_read_dp</code>	All ALTMEMPHY variations (unused for Stratix III or Stratix IV devices)	Takes read data from the IO through a read path FIFO, to transition from the resynchronization clock to the PHY clock.

**Table 2-15.** Modules in `<variation_name>_alt_mem_phy.v` File (Part 2 of 2)

Module Name	Usage	Description
<code>&lt;variation_name&gt;_alt_mem_phy_read_dp_group</code>	DDR3/DDR2/DDR SDRAM ALTMEMPHY variations (Stratix III and Stratix IV devices only)	A per DQS group version of <code>&lt;variation_name&gt;_alt_mem_phy_read_dp</code> .
<code>&lt;variation_name&gt;_alt_mem_phy_rdata_valid</code>	DDR3/DDR2/DDR SDRAM ALTMEMPHY variations	Generates read data valid signal to sequencer and controller.
<code>&lt;variation_name&gt;_alt_mem_phy_seq_wrapper</code> or <code>&lt;variation_name&gt;_alt_mem_phy_qdrii_sequencer_wrapper</code>	All ALTMEMPHY variations	Generates sequencer for DDR3, DDR2, and DDR SDRAM. File with <code>_qdrii</code> is for QDRII+/QDRII SRAM.
<code>&lt;variation_name&gt;_alt_mem_phy_write_dp</code>	All ALTMEMPHY variations	Generates the demultiplexing of data from half-rate to full-rate DDR data.
<code>&lt;variation_name&gt;_alt_mem_phy_write_dp_fr</code>	DDR2/DDR SDRAM ALTMEMPHY variations	A full-rate version of <code>&lt;variation_name&gt;_alt_mem_phy_write_dp</code> .

The ALTMEMPHY variation top-level file instantiates the required submodules depending on the device family and target memory technology.

## Instantiating Multiple ALTMEMPHY Instances

This section discusses the following topics:

- “Clock Sharing”
- “Stratix III and Stratix IV Devices”

### Clock Sharing

The ALTMEMPHY sources a system clock, for use in your design, to clock your ALTMEMPHY interface. This same clock is used by Altera High-Performance controllers, SOPC Builder systems, and the Avalon-MM interface connected to the memory controller. If you use more than one similarly configured ALTMEMPHY in a design, you can share these system clocks, which allows the ALTMEMPHY interfaces to operate synchronously to each other. Clock sharing has the following benefits:

- Although a PLL instance is used for each ALTMEMPHY instance, the static clocks are shared, which uses fewer clocking resources
- The multiple ALTMEMPHYs operate on the same system clock so can be directly connected to logic on the same clock (for example, Avalon-MM interface) without any clock domain crossing logic required, which causes an increase in logic usage and read latency

This clock sharing forces the merging of PLLs and allows ALTMEMPHYs that are operating at the same frequency to connect synchronously to the same Avalon-MM interface.



This clock sharing is based on a Quartus II PLL merging assignment.

You can only use clock sharing when your multiple ALTMEMPHY instances observe the following rules:

- All ALTMEMPHYs are full-rate or all half-rate designs
- All ALTMEMPHYs have the same PLL input and output frequencies
- The ALTMEMPHY memory types are compatible. DDR and DDR2 SDRAM can be mixed in the same system; DDR3 SDRAM controllers are not compatible with DDR or DDR2 SDRAM
- The `ref_clk` input of all instances must be connected to the same signal



For more information instantiating multiple memory controllers, refer to [AN 462: Implementing Multiple Memory Controllers Using the ALTMEMPHY Megafunction](#).

To use clock sharing, follow these steps:

1. Generate two compatible ALTMEMPHY (or high performance controller) instances using the Mega Wizard Plug-In.
2. Create a design instantiating your two ALTMEMPHYs or controllers. If you are using high performance controllers, you can derive a design from the generated `<variation name>_example_top.vhd` or `.v` files.
3. Manually add the two PLL merging assignments. You can use the Assignments Editor or directly update the `.qsf` file.

Table 2-16 shows a list of PLL merging assignments.



SOPC Builder systems apply these assignments automatically.



**Table 2-16.** PLL Merging Assignments (Part 1 of 3)

Device (1)	Rate	Assignment (2)
Arria II GX	Half	<pre>set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll::*:altpll_component *:auto_generated clk[0] -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll::*:altpll_component *:auto_generated clk[0]  set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll::*:altpll_component *:auto_generated clk[3] -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll::*:altpll_component *:auto_generated clk[3]</pre>
	Full	<pre>set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll::*:altpll_component *:auto_generated clk[1] -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll::*:altpll_component *:auto_generated clk[1]  set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll::*:altpll_component *:auto_generated clk[3] -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll::*:altpll_component *:auto_generated clk[3]</pre>

**Table 2-16.** PLL Merging Assignments (Part 2 of 3)

Device (1)	Rate	Assignment (2)
Cyclone III	Half	<pre> set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll:*:altpll_component *:auto_generated clk[0] -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll:*:altpll_component *:auto_generated clk[0]  set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll:*:altpll_component *:auto_generated clk[2] -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll:*:altpll_component *:auto_generated clk[2] </pre>
	Full	<pre> set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll:*:altpll_component *:auto_generated clk[1] -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll:*:altpll_component *:auto_generated clk[1]  set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll:*:altpll_component *:auto_generated clk[2] -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll:*:altpll_component *:auto_generated clk[2] </pre>

**Table 2-16.** PLL Merging Assignments (Part 3 of 3)

Device (1)	Rate	Assignment (2)
Stratix II	Half	<pre>set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll:*:altpll_component *clk0 -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll:*:altpll_component *clk0  set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll:*:altpll_component *clk2 -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll:*:altpll_component *clk2</pre>
	Full	<pre>set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll:*:altpll_component *clk1 -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll:*:altpll_component *clk1  set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk &lt;SLAVE&gt;_phy_alt_mem_ phy_pll:*:altpll_component *clk2 -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk &lt;MASTER&gt;_phy_alt_me m_phy_pll:*:altpll_component *clk2</pre>
Stratix III	Half and full	<pre>set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk write_clk_2x - to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk write_clk_2x  set_instance_assignment -name FORCE_MERGE_PLL_FANOUTS ON - from * &lt;SLAVE&gt;_phy_alt_mem_phy_clk_reset:clk phy_clk_1x -to * &lt;MASTER&gt;_phy_alt_mem_phy_clk_reset:clk phy_clk_1x</pre>

**Notes to Table 2-16:**

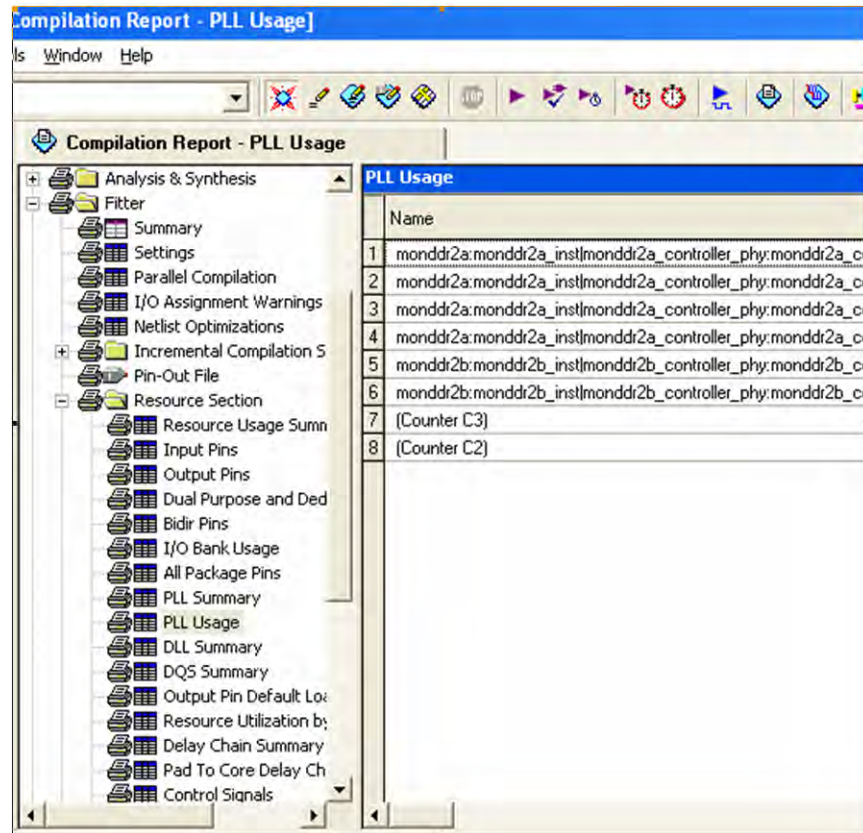
- (1) Cyclone III includes Cyclone III devices; Stratix II includes Stratix II, Stratix II GX, Arria GX or HardCopy II devices; Stratix III includes Stratix III, Stratix IV, HardCopy III or HardCopy IV devices.
- (2) In these assignments, if you are using the ALTMEMPHY megafunction, you need to replace <SLAVE>\_phy and <MASTER>\_phy by the variation names of your two variations. If you are using the high performance controller, you need to replace <SLAVE> and <MASTER> by the variation names of your two variations.


For some Stratix III and Stratix IV devices (F780 and larger), fitter errors may arise because you cannot route a clock from the single shared PLL to the two individual DLLs for each of the controllers.

To avoid these errors, configure the PHYs to share a DLL, by using the **Instantiate DLL externally** option on one of them, and connect its `dqs_delay_ctrl_import` port to the `dqs_delay_ctrl_export` output of the other PHY.

To check that the Quartus II software has applied the assignments, read the Compilation Report - PLL Usage (available only if the fitter step is successful), which shows the two clocks merged. This report allows you to compare the PLL usage before PLL merging and after PLL merging. Figure 2-12 shows the PLL usage.

**Figure 2-12.** Compilation Report PLL Usage



 If the report does not show the clocks merged as expected you should check the `FORCE_MERGE_PLL_FANOUTS` assignment carefully for incorrect clock names. You can also open your `<projectname>_fit.rpt` file and look for Merged PLL.

## Stratix III and Stratix IV Devices

A keep assignment is applied to the `ctl_clk` (`phy_clk` for non-AFI designs) to preserve the name so that it is easy to identify. However, the global clock network assignment that is also made consumes one global clock resource per ALTMEMPHY. For single ALTMEMPHY designs this issue is irrelevant, but if your design instantiates more than one ALTMEMPHY, you may find your design does not fit. You need to allow the Quartus II software the freedom to use regional clock networks for the `ctl_clk`, such that global clock network resources are not used unnecessarily and depleted.

To remove the keep and global assignment from `ctl_clk` (`phy_clk`), follow these steps:

1. Locate the PHY Verilog HDL file (*<variation\_name>\_alt\_mem\_phy.v*) and search for the following line:

```
(* keep, altera_attribute = "-name global_signal      global_clock"  
*) wire phy_clk_1x;
```

2. Edit this line so that it reads:

```
wire phy_clk_1x;
```

3. Save the *<variation\_name>\_alt\_mem\_phy.v* file.
4. Recompile the design.

The Quartus II software now has the flexibility to place the `phy_clk` on any clock network, so the design functions correctly.



If you generated the multiple ALTMEMPHY design using SOPC Builder and you selected **Use clocks from another controller**, the associated `FORCE_MERGE_PLL_FANOUTS` assignments that implement this feature can no longer find the `ctl_clk` (`phy_clk`). You must update these assignments with the new name of the `ctl_clk` (`phy_clk`) path in the Quartus II Assignment Editor. To discover the new `ctl_clk` (`phy_clk`) path, use the **Node Finder**, navigate to the **my\_phy\_name\_alt\_mem\_phy\_pll** instance and select the `c0` output for half-rate designs or the `c2` output for full-rate designs.

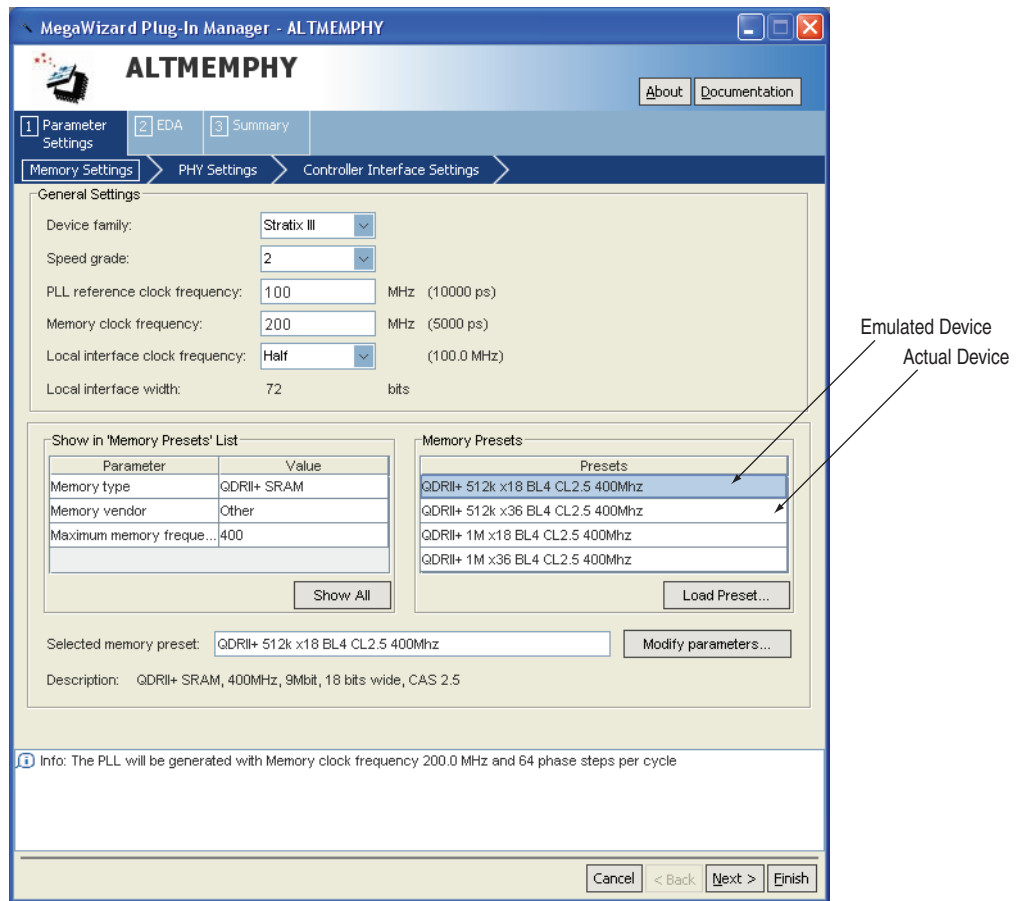
## Creating an Emulated x36 QDRII+/QDRII SRAM ALTMEMPHY Variation

From software implementation point of view, creating a x36 emulated QDRII+/QDRII SRAM interface is exactly the same as implementing an interface with two x18 QDRII+/QDRII SRAM devices. In the **Memory Settings** page of the ALTMEMPHY MegaWizard Plug-In, select a x18 QDRII+/QDRII SRAM with the same timing specifications as x36 QDRII+/QDRII SRAM device (see [Figure 2-13](#)).



For more information on x36 emulation, refer to [“x36 Emulation for QDRII+/QDRII SRAM Interfaces”](#) on page 4-20.

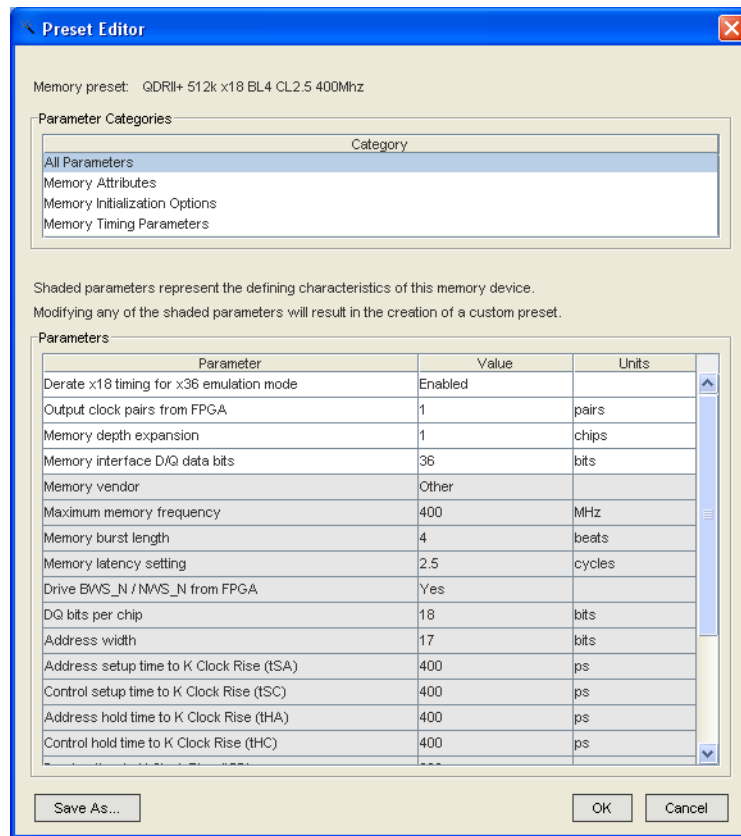
Figure 2-13. Select a x18 Device



To indicate that you are interfacing with 36-bit read and 36-bit write data follow these steps:

1. On the **Memory Settings** page click **Modify Parameters**, to open the **Preset Editor**, see [Figure 2-14](#).

Figure 2-14. Preset Editor



- For **Derate x18 timing for x36 emulation mode**, select **Enabled**. This setting tells the `report_timing.tcl` script to use the derating factor for x36 emulation. If you have modified the board such that the slew rate of the x36 emulated (double-loaded) CQ/CQn signal is comparable to a non-emulated (single-loaded) CQ/CQn signal, you can leave this option as **Disabled**, as there is no slew rate degradation in your design.
- For **Output clock pairs from FPGA** select **1**. Only one `mem_clk` and `mem_clk_n` pair connect to the QDRII+/QDRII SRAM device's K and Kn ports..
- Memory interface D/Q data bits select **36**, which is the data bus width for x36 QDRII+/QDRII SRAM interfaces.

After generation, for devices with F780 and F1152 packages that do not have x18 DQ groups necessary to fit the write data bus, follow these steps to modify the `<variation_name>_pin_assignments.tcl` file to change the assignments to use x9 DQ groups:

- Remove the memory interface data pin group assignment of 18 for write data bus and DM pins in the Assignment Editor.
- Find the following assignments in the `<variation_name>_pin_assignments.tcl`.

```
set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 18 -to
${d_pin_name}\[0..17\] -from ${d_pin_name}\[0\]
```

```

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 18 -to
${d_pin_name}\[0..1\] -from ${d_pin_name}\[0\]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 18 -to
${d_pin_name}\[18..35\] -from ${d_pin_name}\[18\]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 18 -to
${dm_pin_name}\[2..3\] -from ${d_pin_name}\[18\]

```

3. Replace with the following assignment:

```

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 9 -from
${d_pin_name}[0] -to ${d_pin_name}[0..8]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 9 -from
${d_pin_name}[0] -to ${dm_pin_name}[0]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 9 -from
${d_pin_name}[18] -to ${d_pin_name}[18..26]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 9 -from
${d_pin_name}[18] -to ${dm_pin_name}[2]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 9 -from
${d_pin_name}[9] -to ${d_pin_name}[9..17]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 9 -from
${d_pin_name}[9] -to ${dm_pin_name}[1]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 9 -from
${d_pin_name}[27] -to ${d_pin_name}[27..35]

set_instance_assignment -name MEMORY_INTERFACE_DATA_PIN_GROUP 9 -from
${d_pin_name}[27] -to ${dm_pin_name}[3]

```

4. Save the `<variation_name>_pin_assignments.tcl` file.

The rest of the RTL implementation is similar to a regular QDRII+/QDRII SRAM interface. You just need to add a couple of extra timing constraints, see [“Determining the CQ/CQn Arrival Time Skew” on page 4–26](#).

## Inferring Megafunctions from HDL Code

The ALTMEMPHY megafunction cannot be inferred from the HDL code.



If you use a third-party EDA synthesis tool, you can instantiate the megafunction variation file as a black box for synthesis. Use the VHDL component declaration or Verilog HDL module declaration black box file to define the function in your synthesis tool, and then include the megafunction variation file in your Quartus II project.

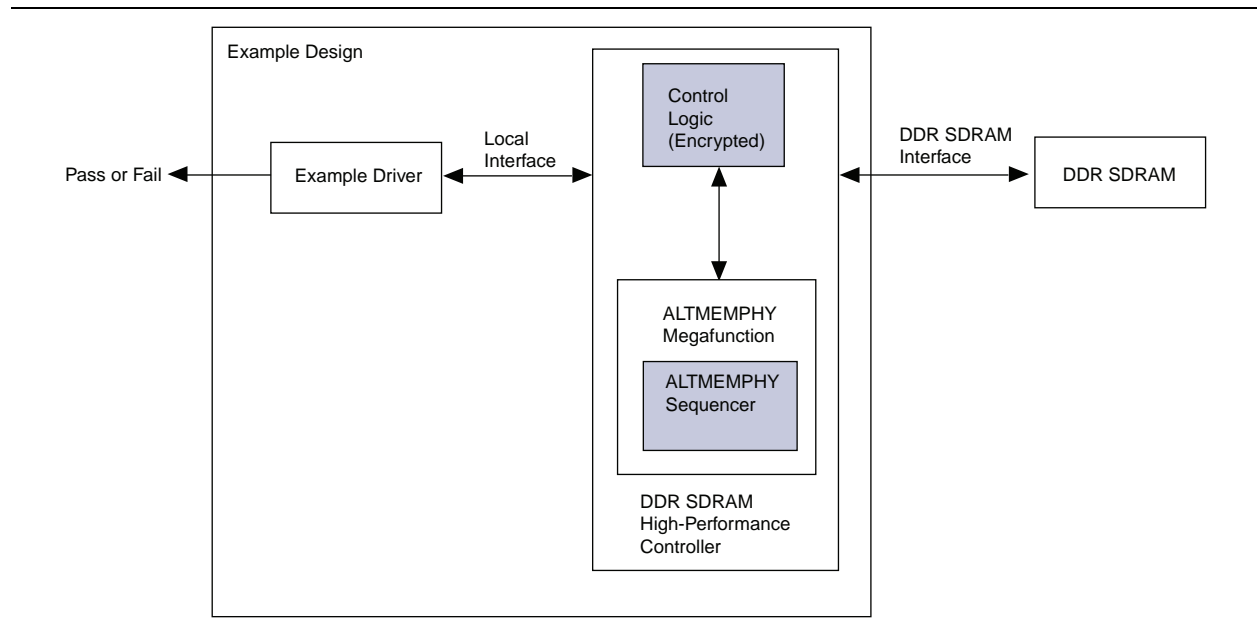
## Compiling in the Quartus II Software

You cannot compile the ALTMEMPHY variation as a stand-alone top-level design, as the generated `.sdc` requires that the ALTMEMPHY variation to be part of a larger design (with a controller and/or example driver). If you want to check whether the ALTMEMPHY variation meets your required target frequency before your memory controller is ready, create a top-level file that instantiates this ALTMEMPHY variation.



Figure 2-15 shows the top-level view of the Altera DDR SDRAM High-Performance Controller design as an example on how your final design looks after you integrate the controller and the user logic. The top-level view of the Altera DDR3 and DDR2 SDRAM High-Performance Controller design looks similar to Figure 2-15.

Figure 2-15. DDR SDRAM High-Performance Controller System-Level Diagram



Before compiling a design with the ALTMEMPHY variation, you must edit some project settings, include the .sdc file, and make I/O assignments. I/O assignments include I/O standard, pin location, and other assignments, such as termination and drive strength settings. Some of these tasks are listed at the ALTMEMPHY **Generation** window, see Figure 2-11 on page 2-28. For most systems, Altera recommends that you use the **Advanced I/O Timing** feature by using the **Board Trace Model** command in the Quartus II software to set the termination and output pin loads for the device.

You can either use the `<variation_name>_pin_assignments.tcl` or the `<variation_name>.ppf` file to apply the I/O assignments generated by the ALTMEMPHY MegaWizard Plug-In. Using the .ppf file and the Pin Planner gives you the extra flexibility to add a prefix to your memory interface pin names. You can edit the assignments either in the Assignment Editor or Pin Planner.

If your design contains pin names that do not match the design, edit the `<variation name>_pin_assignments.tcl` file before you run the script. Follow these steps:

1. Open `<variation name>_pin_assignments.tcl` file.
2. Based on the flow you are using, set the `sopc_mode` value to **Yes** or **No**.

```
if {[info exists socp_mode]} {set socp_mode NO}
```

3. Type your preferred prefix in the `pin_prefix` variable. For example, to add the prefix `my_mem`:

```
if {[info exists set_prefix]}{set pin_prefix "my_mem_"}
```

After setting the prefix, the pin names are expanded, for example:

```
my_mem_cs_n[0]
```



If your top-level design does not use single bit bus notation for the single-bit memory interface signals (for example, `mem_dqs` rather than `mem_dqs[0]`), in the Tcl script you should change `set single_bit {[0]}` to `set single_bit {}`.



For more information on how to compile a design with ALTMEMPHY variation, refer to the appropriate document in “References” on page Info-3.



For more information on how to specify I/O standard assignment for pins, refer to the “Compile the Example Design” section in the *DDR and DDR2 SDRAM Controller Compiler User Guide*.

## Identifying a Megafunction After Compilation

During compilation with the Quartus II software, analysis and elaboration are performed to build the structure of your design. To locate your megafunction in the Project Navigator window, expand the compilation hierarchy and find the megafunction by its name. To search for node names within the megafunction (using Node Finder), click **Browse** in the **Look in** dialog box and select the megafunction in the **Hierarchy** dialog box.

## Analyzing Timing


Because ALTMEMPHY timing is critical to your system functionality, the MegaWizard Plug-In generates timing constraining and timing reporting scripts, allowing you to perform detailed timing analysis of both core and I/O timing paths, see Table 2-17. You need to use ALTMEMPHY-generated timing constraint file (`<variation_name>_ddr_timing.sdc`) and the TimeQuest Timing Analyzer to perform the timing analysis.



The `.sdc` file may not be applied correctly if the ALTMEMPHY variation top-level file is the top-level file of the project. You must have the top-level file of the project instantiate the ALTMEMPHY variation top-level file.

Timing analysis of ALTMEMPHY is supported only by TimeQuest Timing Analyzer, for the following reasons:

- The timing constraint scripts generated by the ALTMEMPHY megafunction only support the TimeQuest Timing Analyzer.
- The Classic Timing Analyzer does not offer analysis of source-synchronous outputs. For example, write data, and address and command outputs. The Classic Timing Analyzer does not support detailed rise and fall delay analysis.

 For more information about timing analysis for Cyclone III or Stratix III FPGAs, refer to *AN 438: Constraining and Analyzing Timing for External Memory Interfaces in Stratix IV, Stratix III and Cyclone III Devices*.


## Timing Constraints

The ALTMEMPHY MegaWizard Plug-In generates the following files for timing constraints and reporting scripts:

1. `<variation_name>_ddr_timing.sdc`

This file sets the following constraints for the ALTMEMPHY variation:


- Creates all the necessary clocks with proper clock periods
- Sets all the board and memory parameter settings
- Sets the output delay on the DQS pins for write analysis (Arria GX, Stratix II, Stratix II GX, and HardCopy II devices only)
- Sets the input delay on the DQS pins for read analysis (Arria GX, Stratix II, Stratix II GX, and HardCopy II devices only)

 For Arria II GX, Stratix III, and Stratix IV devices, when you use the **pin\_assignments.sdc** script, the design meets the macro timing assumptions, so there is no need to explicitly set delays.

- Sets clock uncertainty for DQS pins versus CK pins timing analysis
- Sets the output delay on the address and command pins
- Sets necessary multicycle-path and false-path assignments

You only need one **.sdc** file, even when you instantiate the same variation of the ALTMEMPHY megafunction multiple times in your top-level design.

 The high-performance controller MegaWizard Plug-In generates an extra **.sdc** file for the example driver design.

 For more information on the **.sdc** file, refer to *AN 438: Constraining and Analyzing Timing for External Memory Interfaces in Stratix IV, Stratix III and Cyclone III Devices*.

2. `<variation_name>_report_timing.tcl`

This script reports the timing slacks for your ALTMEMPHY variation. It is run automatically during compilation. You can also run this script with the Report DDR task in the TimeQuest Timing Analyzer window. This script is run for every instance of the same ALTMEMPHY variation.

3. `<variation_name>_ddr_pins.tcl`

This script includes all the functions and procedures required by the `<variation_name>_report_timing.tcl` and `<variation_name>_ddr_timing.sdc` scripts. It finds all the ALTMEMPHY variation instances in the design and the associated clock, register, and pin names of each instances. The results are saved in the same directory as the **.sdc** and `<variation_name>_report_timing.tcl` as `<variation_name>_autodetectedpins.tcl`.



Because this `.tcl` file traverses the design for the project pin names, you do not need to keep the same port names on the top level of the design.

## Timing Analysis Using the TimeQuest Timing Analyzer

The `report_timing.tcl` script is run automatically during compilation as long as there is a top-level module instantiating the ALTMEMPHY variation. The script runs the timing analysis under all available timing models and then reports the worst-case margin for each path. In the TimeQuest Timing Analyzer window, you can run the `report_timing.tcl` script that generates the same timing report after a compilation by double-clicking on the **Report DDR** task.

For Arria II GX, Cyclone III, Stratix IV, and Stratix III devices, some of the timing calculations in the `report_timing.tcl` script use the following assumptions:

1. The memory interface pins are connected properly as recommended in the device family handbook.
2. I/O standard assignments are correct for the interface.
3. The `pll_ref_clk` pin is a dedicated clock input to the PLL used in the interface or a dedicated clock input to the PLL neighboring the PLL used in the interface.
4. The PLL is in the correct operation mode, which is **With no compensation** in Arria II GX, Stratix III and Stratix IV devices and **Normal mode** in Cyclone III devices. There are no PLL compensation requirements in Arria GX, Stratix II, or Stratix II GX devices.



For optimal performance, you need to have the PLL for the ALTMEMPHY megafunction located on the same side of the device as the memory interface.

For Stratix III devices, PLL cascading uses the following assumptions, if another PLL feeds the input clock to the ALTMEMPHY PLL:

- A dedicated input pin must feed the source PLL.
- The source PLL must use **With no compensation** operation mode.
- The source PLL must be configured with **Low** bandwidth, and the ALTMEMPHY PLL must be configured with **High** bandwidth.
- The ALTMEMPHY PLL input clock must come from the dedicated PLL to PLL cascade path.

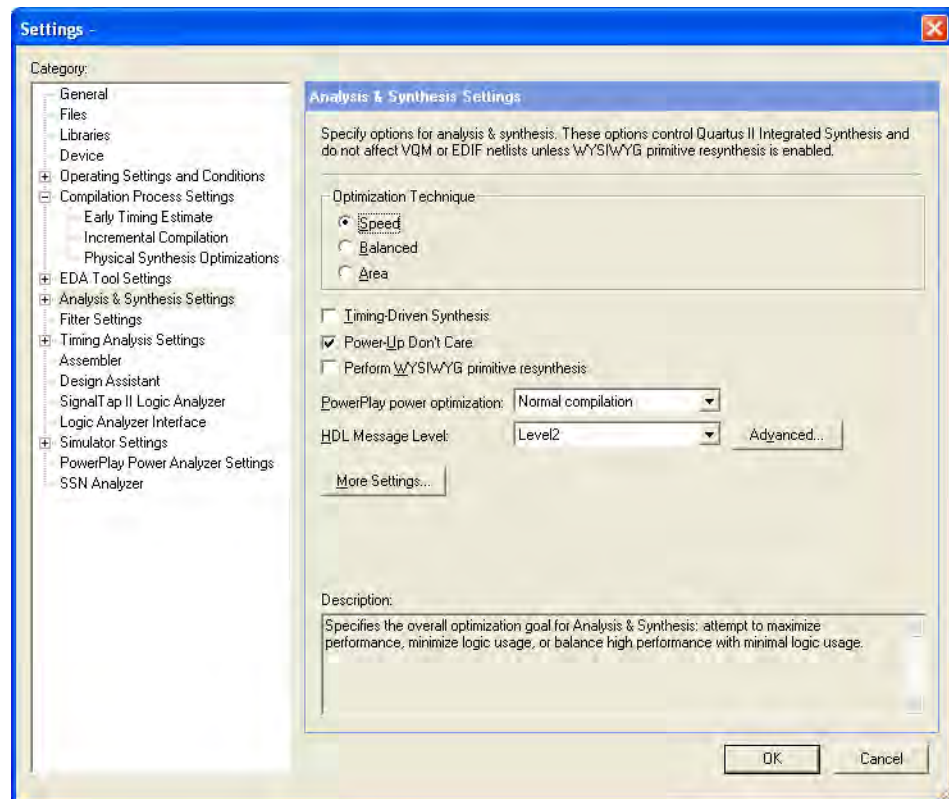
If you do not adhere to these restrictions in this section, you receive critical warnings when the TimeQuest Timing Analyzer is run during compilation or when you run the **Report DDR** task. Your timing results are not going to be accurate as you did not follow the correct restrictions stated above.

## Optimizing Timing

For full-rate designs you may need to use some of the Quartus II 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** (see [Figure 2-16](#)).

Figure 2-16. Optimization Technique



2. In the **Category** list, click **Physical Synthesis Optimizations**. Specify the following options:

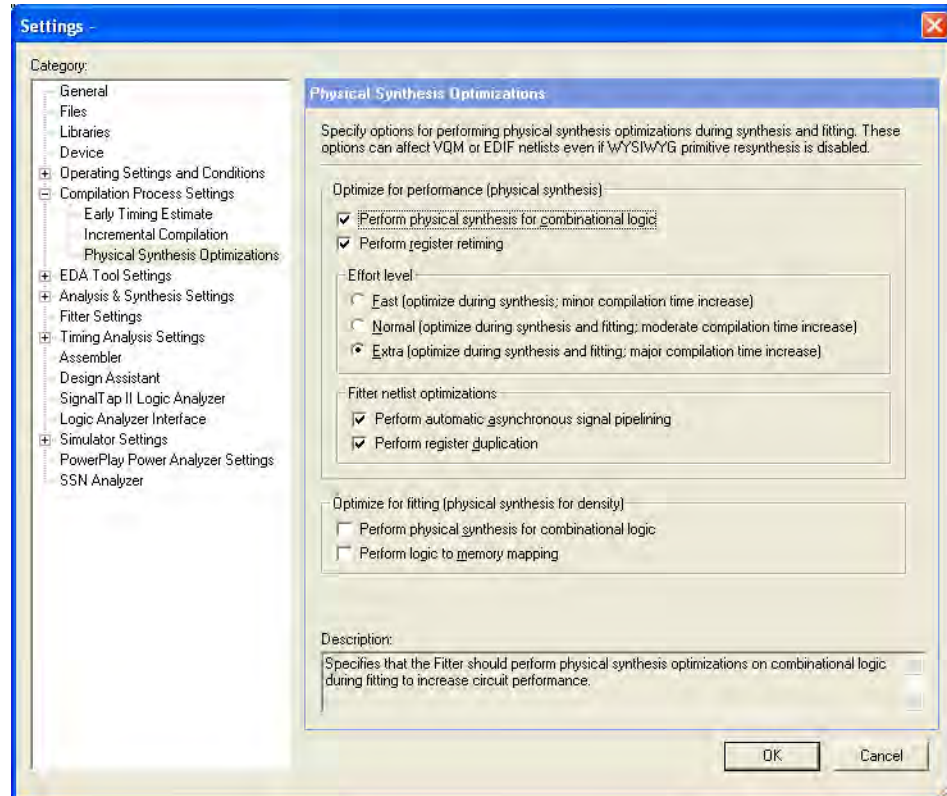
- Turn on **Perform physical synthesis for combinational logic**.

 For more information on physical synthesis, refer to the [Netlist and Optimizations and Physical Synthesis](#) chapter in the *Quartus II 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** (see [Figure 2-17](#)).

Figure 2-17. Physical Synthesis Optimizations



## Analyzed Timing Paths

Table 2-17 lists the timing paths that are analyzed by the **Report DDR** in the TimeQuest Timing Analyzer.

**Table 2-17.** ALTMEMPHY Timing Paths (Part 1 of 5) (Note 1)

Timing Path	Applicable ALTMEMPHY Variation(s)	Applicable Clock (2)	Description
Address and command	All ALTMEMPHY variations	mem_clk ( <i>&lt;variation_name&gt;</i> _ck_n_mem_clk_n[ <i>i</i> ] <sub>ac_rise</sub> <i>&lt;variation_name&gt;</i> _ck_p_mem_clk[ <i>i</i> ] <sub>ac_rise</sub> <i>&lt;variation_name&gt;</i> _ck_n_mem_clk_n[ <i>i</i> ] <sub>ac_fall</sub> <i>&lt;variation_name&gt;</i> _ck_p_mem_clk[ <i>i</i> ] <sub>ac_fall</sub> )	Setup and hold margin for all address and command pins (in QDR II SRAM variation or in DDR2/DDR SDRAM variation in full-rate mode) or for mem_cke, mem_cs_n, and mem_odt pins (in DDR3/DDR2/DDR SDRAM variation in half-rate mode)
PHY	All ALTMEMPHY variations	All clocks in the PHY that drive ALTMEMPHY registers	Internal timing of the ALTMEMPHY megafunction.

**Table 2-17.** ALTMEMPHY Timing Paths (Part 2 of 5) (Note 1)

Timing Path	Applicable ALTMEMPHY Variation(s)	Applicable Clock (2)	Description
PHY reset	All ALTMEMPHY variations	All clocks in the PHY that drive ALTMEMPHY registers	Internal timing of the asynchronous reset signals to the ALTMEMPHY megafunction.
DQS vs. CK	DDR2/DDR SDRAM	$\langle variation\_name \rangle\_ck\_n\_mem\_clk\_n[i]_tDQSS$ $\langle variation\_name \rangle\_ck\_p\_mem\_clk[i]_tDQSS$ $\langle variation\_name \rangle\_ck\_n\_mem\_clk\_n[i]_tDSS$ $\langle variation\_name \rangle\_ck\_p\_mem\_clk[i]_tDSS$	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.
Half-rate address and command	DDR3/DDR2/DDR SDRAM in half-rate mode	$mem\_clk$ $(\langle variation\_name \rangle\_ck\_n\_mem\_clk\_n[i]_ac\_rise$ $\langle variation\_name \rangle\_ck\_p\_mem\_clk[i]_ac\_rise$ $\langle variation\_name \rangle\_ck\_n\_mem\_clk\_n[i]_ac\_fall$ $\langle variation\_name \rangle\_ck\_p\_mem\_clk[i]_ac\_fall)$	Setup and hold margin for the address and command pins (except for $mem\_cs\_n$ , $mem\_cke$ , and $mem\_odt$ pins) with respect to the $mem\_clk$ clock at the memory when the PHY is in half-rate mode.
Mimic	DDR2/DDR SDRAM variation in Arria GX, Arria II GX, Cyclone III, HardCopy II, Stratix II, and Stratix II GX Devices	$\langle variation\_name \rangle\_ck\_n\_mem\_clk[0]$	The setup margin for the VT tracking mechanism.

**Table 2-17.** ALTMEMPHY Timing Paths (Part 3 of 5) (Note 1)

Timing Path	Applicable ALTMEMPHY Variation(s)	Applicable Clock (2)	Description
Read capture	All ALTMEMPHY variations	dqs ( <i>&lt;variation_name&gt;</i> _ddr_dqs_in_mem_dqs)	<p>Setup and hold margin for the DQ pins with respect to DQS strobe at the FPGA capture registers.</p> <p>In Arria II GX, Stratix III, and Stratix IV devices, the margin is reported via an equation based on the applicable sampling window (SW) value for the configuration.</p> <p>Refer to the <i>&lt;variation_name&gt;</i>_report_timing.tcl file, for the equation or refer to the “DC and Switching Characteristics” chapter in the relevant device handbook.</p> <p>In Cyclone III devices, it is a calibrated path. The resulting setup and hold margin, in these three device families, is always balanced as it is calculated as the total margin divided by two. Your actual setup and hold margin on the board can differ.</p>
Read Postamble	DDR2/DDR SDRAM variation in Arria GX, Stratix II, and Stratix II GX Devices	Postamble clock ( <i>&lt;variation_name&gt;</i> _ddr_postamble)	<p>Setup and hold time margin for the postamble path that is calibrated with the resynchronization clock phase.</p> <p>The setup and hold margin is always balanced as it is calculated as the total margin divided by two. Your actual setup and hold margin on the board can differ.</p>
Read Postamble Enable	DDR2/DDR SDRAM variation in Arria GX, Stratix II, and Stratix II GX Devices	DQS clocks ( <i>&lt;variation_name&gt;</i> _ddr_dqs_in_*)	The setup and hold margin for the postamble logic that enables and disables the DQS signal going to the DQ registers.



**Table 2-17.** ALTMEMPHY Timing Paths (Part 4 of 5) (Note 1)


Timing Path	Applicable ALTMEMPHY Variation(s)	Applicable Clock (2)	Description
Read Resynchronization	DDR3/DDR2/ DDR SDRAM	Resynchronization clock ( <variation_name>_ddr_resync )	<p>Setup and hold margin for the DQ data with respect to resynchronization and the postamble clock at the resynchronization and the postamble registers.</p> <p>In Arria II GX, Stratix III, and Stratix IV devices, the margin is reported via an equation based on the applicable SW value for the configuration.</p> <p>Refer to the &lt;variation_name&gt;_report_timing.tcl file, for the equation.</p> <p>The setup and hold margin is always balanced as it is calculated as the total margin divided by two. Your actual setup and hold margin on the board can differ.</p>
Write datapath	All ALTMEMPHY variations	DQS ( <variation_name>_ddr_dqsout_mem_dqs )	<p>Setup and hold margin for the DQ pins with respect to DQS strobe at the memory.</p> <p>In Arria II GX, Cyclone III, Stratix III, and Stratix IV devices, the margin is reported via an equation based on the applicable channel-to-channel skew (TCCS) value for the configuration.</p> <p>The setup and hold margin is always balanced as it is calculated as the total margin divided by two. Your actual setup and hold margin on the board can differ.</p>

**Table 2-17.** ALTMEMPHY Timing Paths (Part 5 of 5) (Note 1)

Timing Path	Applicable ALTMEMPHY Variation(s)	Applicable Clock (2)	Description
Write leveling $t_{DQSS}$	DDR3 SDRAM (except Arria II GX devices)	CK/CK# clocks ( <i>&lt;variation_name&gt;_ck_n_&lt;CK# pin name&gt;_tDQSS</i> ) or ( <i>&lt;variation_name&gt;_ck_p_&lt;CK pin name&gt;_tDQSS</i> )	Skew margin for the arrival time of the DQS strobe at the memory with respect to the arrival time of CK/CK# at the memory.  This path is a calibrated path, such that the margin is reported via an equation. The setup and hold margin is always balanced as it is calculated as the total margin divided by two. Your actual setup and hold margin on the board can differ.
Write leveling $t_{DSS}/t_{DSH}$	DDR3 SDRAM (except Arria II GX devices)	CK/CK# clocks ( <i>&lt;variation_name&gt;_ck_n_&lt;CK# pin name&gt;_tDQSS</i> ) or ( <i>&lt;variation_name&gt;_ck_p_&lt;CK pin name&gt;_tDQSS</i> )	Setup and hold margin for the DQS falling edge with respect to the CK clock at the memory.  A calibrated path, such that the margin is reported via an equation. The setup and hold margin is always balanced as it is calculated as the total margin divided by two. Your actual setup and hold margin on the board can differ.

**Note to Table 2-17:**

- (1) You cannot see the details of the timing nodes for any calibrated path (such as the read postamble and read resynchronization paths) in Stratix II devices. You also cannot see the details of the timing nodes for read and write paths in Cyclone III, Stratix III, and Stratix IV devices as these paths are calculated using the sampling window and channel-to-channel skew published in the “DC and Switching Characteristics” chapter in the device family handbook.
- (2) [ i ] refers to the clock number—1, 2, or 3.

 For more information about these timing paths, refer to *AN 438: Constraining and Analyzing Timing for External Memory Interfaces in Stratix IV, Stratix III and Cyclone III Devices*.



## Timing Closure

This section describes potential timing closure issues that can be found when using ALTMEMPHY. For possible timing closure issues with ALTMEMPHY variations, refer to the *Quartus II 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 II Software Release Notes* list common timing issues that can be encountered in a particular version of the Quartus II software.

For Stratix III, Stratix IV, HardCopy III, and HardCopy 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.


-  For more information about closing timing for DDR2 and DDR SDRAM interfaces in Stratix II devices, refer to *AN 328: Interfacing DDR2 SDRAM with Stratix II, Stratix II GX, and Arria GX Devices*.
-  For information about ALTMEMPHY timing for Stratix IV, Stratix III, and Cyclone III devices, refer to *AN 438: Constraining and Analyzing Timing for External Memory Interfaces in Stratix IV, Stratix III and Cyclone III Devices*.


## Simulating your Design

The ALTMEMPHY megafunction cannot be simulated alone. To simulate the ALTMEMPHY megafunction, you must use all of the following blocks:

- Memory controller
- Example driver (to initiate read and write transactions)
- Testbench and a suitable memory model

Simulating the whole memory interface is a good way to find the latency for 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 ALTMEMPHY megafunction only supports functional simulation; it does not support gate-level simulation, for the following reasons:
  - The ALTMEMPHY is a calibrated interface. Therefore, gate-level simulation time can be very slow and take up to several hours to complete.
  - Gate-level timing annotations, and the phase sweeping that the calibration uses, give setup and hold violations. Because of the effect of X (unknown value) propagation within the atom simulation models, this action causes gate-level simulation failures in some cases.
  - Memory interface timing methodology does not match timing annotation that gate-level simulations use. Therefore the gate-level simulation does not accurately match real device behavior.

 Altera recommends that you validate the functional operation of your design via RTL simulation, and the timing validation of your design using TimeQuest Timing Analysis.

## DDR2/DDR SDRAM, DDR3 SDRAM (without leveling), and QDRII+/QDRII SRAM Simulation

In ALTMEMPHY variations for DDR2/DDR SDRAM, DDR3 SDRAM without leveling, and QDRII+/QDRII SRAM interfaces, you have the following simulation options:


- Quick calibration—Performs a calibration on a single pin and chip select.

 You may see memory model warnings about initialization times.

- Full calibration—Across all pins and chip selects.

In ALTMEMPHY designs targeting DDR2/DDR SDRAM interfaces for Arria GX, HardCopy II, Stratix II, and Stratix II GX devices, you must add the following line in the project .qsf file to allow NativeLink simulation to find the correct .mif file for the PLL reconfiguration module:

```
set_global_assignment -name EDA_TEST_BENCH_FILE
<variation_name>_phy_alt_mem_phy_pll_sii.mif -section_id
<testbench_name>
```


 This line is not needed for Stratix III, Stratix IV, and Arria II GX devices, as the design uses the PLL phase-stepping feature not the full-blown reconfiguration.

The ALTMEMPHY datapath is in Verilog HDL; the sequencer is in VHDL. For ALTMEMPHY designs with the AFI, to allow the Verilog HDL simulator to simulate the design after modifying the VHDL sequencer, follow these steps:

1. On the View menu, point to **Utility Windows**, and click **TCL console**.
2. Enter the following command in the console:

```
quartus_map --read_settings_file=on --write_settings_file=off --
source=<variation_name>_phy_alt_mem_phy_seq.vhd --
source=<variation_name>_phy_alt_mem_phy_seq_wrapper.v --simgen --
simgen_parameter=CBX_HDL_LANGUAGE=verilog
<variation_name>_phy_alt_mem_phy_seq_wrapper -c
<variation_name>_phy_alt_mem_phy_seq_wrapper
```

The Quartus II software regenerates `<variation_name>_phy_alt_mem_phy_seq_wrapper.vo` and uses this file when the simulation runs.

 For more information on how to perform simulation, refer to the *DDR and DDR2 SDRAM High-Performance Controller User Guide* or the application notes listed in “References” on page Info-3.


## DDR3 SDRAM (with leveling) Simulation

In ALTMEMPHY variations for DDR3 SDRAM interfaces, you have the following three simulation options:

- Skip calibration—Available for ×4 and ×8 DDR3 SDRAM (with leveling). Skip calibration simulation is for 300 MHz through 400 MHz (for frequencies between 400 and 533 MHz, use quick calibration or full calibration). There is no calibration in this simulation mode. The ALTMEMPHY megafunction is statically configured to provide the correct write and read latencies. Skip calibration provides the fastest simulation time for DDR3 SDRAM interfaces. Use the generated or vendor DDR3 SDRAM simulation models for this simulation option. Skip calibration simulation only supports CAS latency of 6 and a CAS write latency of 5.
- Quick calibration—Available for ×4 and ×8 DDR3 SDRAM (with leveling). In quick calibration simulation mode, the sequencer only does clock cycle calibration. So there must be no delays (DDR3 DIMM modeling for example) in the testbench, because no phase calibration is performed. Quick calibration mode can be used between 300 MHz and 533 MHz. Both the generated or vendor DDR3 SDRAM simulation models support burst length on-the-fly changes during the calibration sequence.
- Full calibration—Available for ×4 and ×8 DDR3 SDRAM (with leveling) between 300 MHz and 533 MHz. You cannot use the wizard-generated memory model, if you select **Full Calibration**. You must use a memory-vendor provided memory model that supports write leveling calibration.

If the following warning messages appear in DDR3 SDRAM skip calibration and quick calibration simulation mode, you can change the values of the two parameters (`tinit_tck` and `tinit_rst`) in the `<variation_name>_phy_alt_mem_phy_seq_wrapper.vo` or `<variation_name>_phy_alt_mem_phy_seq_wrapper.vho` files to match the parameters in `<variation_name>_phy_alt_mem_phy_seq_wrapper.v`. These warning messages do not appear in full calibration mode.


```
WARNING: 200 us is required before RST_N goes inactive
WARNING: 500 us is required after RST_N goes inactive before CKE goes active
```

 You may see the following warning and error messages with full calibration simulation during write leveling:

```
Warning: tWLS violation on DQS bit 0 positive edge. Indeterminate CK capture is possible
Warning: tWLH violation on DQS bit 0 positive edge. Indeterminate CK capture is possible.
ERROR: tDQSH violation on DQS bit 0
```

 You may see the following warning messages at time 0 (before reset) of simulation, which you can ignore:

```
Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
Warning: NUMERIC_STD.TO_INTEGER: metavalue detected, returning 0
```

 You may see the following warning and error messages during reset, which you can ignore.

```
Error : clock switches from 0/1 to X (Unknown value) on DLL instance
```

Warning : Duty Cycle violation DLL instance Warning: Input clock duty cycle violation.



For more information on how to perform simulation, refer to the *DDR3 SDRAM High-Performance Controller User Guide* or the application notes listed in “References” on page Info-3.

## Simulation Files

The Quartus II software automatically creates the required files list and autocompilation script for any supported RTL simulation tool if you specify the tool and language settings in the EDA Tool **Settings-Simulation** page of the **Setting** dialog box.

In general, you need the following files to simulate:

- Library files from the `<Quartus II install path>/quartus/eda/sim_lib/` directory:
  - **220model**
  - **altera\_primitives**
  - **altera\_mf**
  - **sgate**
  - **stratixiv\_atoms / stratixiii\_atoms / cycloneiii\_atoms / stratixii\_atoms / stratixiigx\_atoms** (device dependent)



If you are targeting Stratix IV devices, you need both the Stratix IV and Stratix III files (**stratixiv\_atoms** and **stratixiii\_atoms**) to simulate in your simulator, unless you are using NativeLink.

- The sequencer wrapper file (in **.vo** or **.vho** format)
- PLL file (for example `<variation_name>_alt_mem_phy_pll.v` or `.vhd`)
- The ALTMEMPHY modules (in the `<variation_name>_alt_mem_phy.v`)
- The top-level file
- User logic, or a driver for the PHY
- Testbench
- Memory model

### Introduction

This chapter describes the general specifications of the ALTMEMPHY megafunctions that are common between memory standards or Altera device families. It also describes calibration process, read and write data mapping, and half-rate address and command clocking.

### Calibration

This section describes the calibration that the sequencer performs, to find the optimal clock phase for the memory interface. The calibration sequence is similar across families, but different depending on the following target memory interface:

- “DDR2/DDR SDRAM and DDR3 SDRAM (Component)”
- “DDR3 SDRAM (DIMM)”
- “QDRII+/QDRII SRAM”

#### DDR2/DDR SDRAM and DDR3 SDRAM (without leveling)

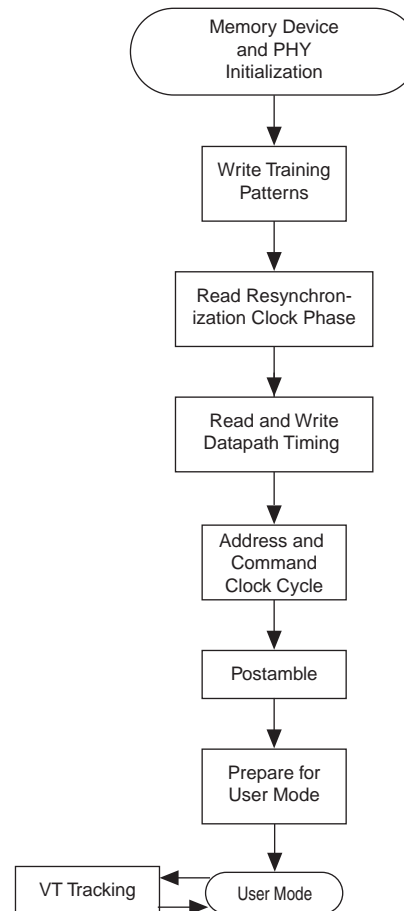
The ALTMEMPHY variation for DDR2/DDR SDRAM and DDR3 SDRAM without leveling interfaces has a similar calibration process for the AFI and non-AFI.

The calibration process for the DDR2/DDR SDRAM and DDR3 SDRAM PHY without leveling includes the following steps:

- “Step 1: Memory Device Initialization”
- “Step 2: Write Training Patterns”
- “Step 3: Read Resynchronization (Capture) Clock Phase”
- “Step 4: Read and Write Datapath Timing”
- “Step 5: Address and Command Clock Cycle”
- “Step 6: Postamble”
- “Step 7: Prepare for User Mode”

Figure 3-1 shows the calibration flow.

**Figure 3-1.** Calibration Flow—DDR2/DDR SDRAM and DDR3 SDRAM (without leveling)




### Step 1: Memory Device Initialization

This step initializes the memory device according to the DDR, DDR2 or DDR3 SDRAM specification. The initialization procedure includes resetting the memory device (DDR3 SDRAM only), setting up the mode registers, setting up memory device ODT setting (DDR2 and DDR3 SDRAM only) and initializing the memory device DLL. Calibration requires some overrides of user-specified mode register settings, which are reverted in “Step 7: Prepare for User Mode”.

### Step 2: Write Training Patterns


In this step, a pattern is written to the memory to be read in later calibration stages. The matched trace lengths to DDR SDRAM devices mean that after memory initialization, write capture works. The pattern is 0x30F5 and comprises the following separately written patterns:


 This pattern is required to match the characterization behavior for non-DQS capture-based schemes (for example, the Cyclone III devices).

- All 0: `b0000 - DDIO high and low bits held at 0



- All 1: 'b1111 - DDIO high and low bits held at 1
- Toggle: 'b0101 - DDIO high bits held at 0 and DDIO low bits held at 1
- Mixed: 'b0011 - DDIO high and low bits have to toggle

 Loading a mixed pattern is complex, because write latency is unknown at this time. Two sets of write and read operations (single pin resynchronization (capture) clock phase sweeps, (“[Step 3: Read Resynchronization \(Capture\) Clock Phase](#)”) are required to accurately write the mixed pattern to memory.

 Memory bank 0, row 0, and column addresses 0 to 55 store calibration data.

### Step 3: Read Resynchronization (Capture) Clock Phase

This step adjusts the phase of the resynchronization (or capture) clock to determine the optimal phase that gives the greatest margin. For DQS-based capture schemes, the resynchronization clock captures the outputs of DQS capture registers (DQS is the capture clock). In a non-DQS capture-based scheme, the capture clock captures the input DQ pin data (the DQS signal is unused, and there is no resynchronization clock).

To correctly calibrate resynchronization (or capture) clock phase, based on a data valid window, requires the following degrees of phase sweep:

- 720° for all half-rate interfaces and full-rate DQS-based capture PHY
- 360° for a full-rate non-DQS capture PHY

### Step 4: Read and Write Datapath Timing

In this step, the sequencer calculates the calibrated write latency (the `ctl_wlat` signal) between write commands and write data. The sequencer also calculates the calibrated read latency (the `ctl_rlat` signal) between the issue of a read command and valid read data. Both read and write latencies are output to a controller. In addition to advertising the read latency, the sequencer calibrates a read data valid signal to the delay between a controller issuing a read command and read data returning. The controller can use the read data valid signal in place of the advertised read latency, to determine when the read data is valid.

### Step 5: Address and Command Clock Cycle

For half-rate interfaces, this step also optionally adds an additional memory clock cycle of delay from the address and command path. This delay aligns write data to memory commands given in the controller clock domain. If you require this delay, this step reruns the calibration (“[Step 2: Write Training Patterns](#)” to “[Step 4: Read and Write Datapath Timing](#)”) to calibrate to the new setting.

### Step 6: Postamble

This step sets the correct clock cycle for the postamble path. The aim of the postamble path is to eliminate false DQ data capture because of postamble glitches on the DQS signal, through an override on DQS. This step ensures the correct clock cycle timing of the postamble enable (override) signal.

 Postamble is only required for DQS-based capture schemes.

## Step 7: Prepare for User Mode

In this step, the PHY applies user mode register settings and performs periodic VT tracking.

### VT Tracking

VT tracking is a background process that tracks the voltage and temperature variations to maintain the relationship between the resynchronization or capture clock and the data valid window that are achieved at calibration.

When the data calibration phase is completed, the sequencer issues the mimic calibration sequence every 128 ms.

During initial calibration, the mimic path is sampled using the measure clock (`measure_clk` has a `_1x` or `_2x` suffix, depending whether the ALTMEMPHY is a full-rate or half-rate design). The sampled value is then stored by the sequencer. After a sample value is stored, the sequencer uses the PLL reconfiguration logic to change the phase of the measure clock by one VCO phase tap. The control sequencer then stores the sampled value for the new mimic path clock phase. This sequence continues until all mimic path clock phase steps are swept. After the control sequencer stores all the mimic path sample values, it calculates the phase which corresponds to the center of the high period of the mimic path waveform. This reference mimic path sampling phase is used during the VT tracking phase.

In user mode, the sequencer periodically performs a tracking operation as defined in the tracking calibration description. At the end of the tracking calibration operation, the sequencer compares the most recent optimum tracking phase against the reference sampling phase. If the sampling phases do not match, the mimic path delays have changed due to voltage and temperature variations.

When the sequencer detects that the mimic path reference and most recent sampling phases do not match, the sequencer uses the PLL reconfiguration logic to change the phase of the resynchronization clock by the VCO taps in the same direction. This allows the tracking process to maintain the near-optimum capture clock phase setup during data tracking calibration as voltage and temperature vary over time.


The relationship between the resynchronization or capture clock and the data valid window is maintained by measuring the mimic path variations due to the VT variations and applying the same variation to the resynchronization clock.

### Mimic Path

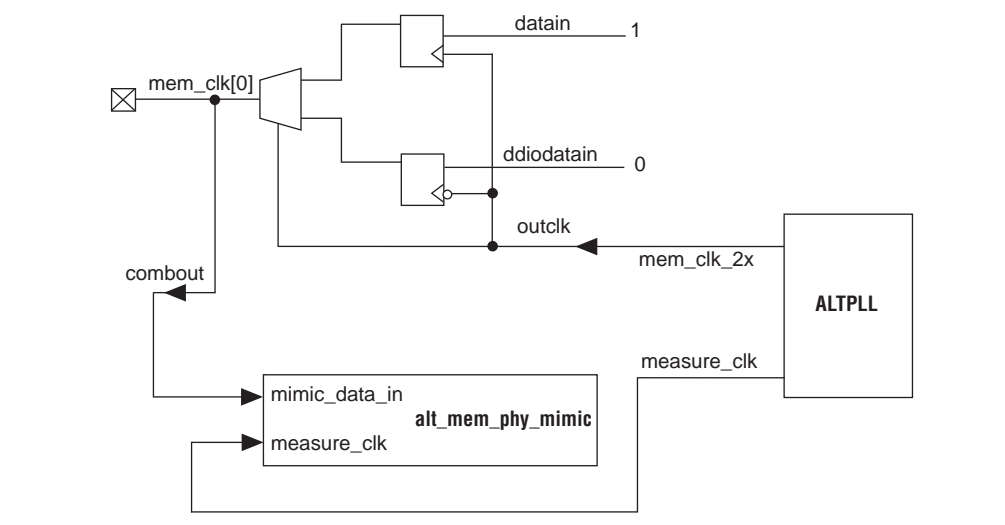
The mimic path mimics the FPGA portion of the elements of the round-trip delay, which enables the calibration sequencer to track delay variation due to VT changes during the memory read and write transactions without interrupting the operation of the ALTMEMPHY megafunction.

The assumption made about the mimic path is that the VT variation on the round trip delay path that resides outside of the FPGA is accounted for in the board skew and memory parameters entered in the MegaWizard Plug-In. For the write direction, any VT variation in the memory devices is accounted for by timing analysis.

Figure 3–2 shows the mimic path in Arria GX, Cyclone III, Stratix II, and Stratix II GX devices, which mimics the delay of the clock outputs to the memory as far as the pads of the FPGA and the delay from the input DQS pads to a register in the FPGA core. During the tracking operation, the sequencer measures the delay of the mimic path by varying the phase of the measure clock. Any change in the delay of the mimic path indicates a corresponding change in the round-trip delay, and a corresponding adjustment is made to the phase of the resynchronization or capture clock.


 The mimic path in Arria II GX, Stratix III and Stratix IV devices is similar to Figure 3–2. The only difference is that the `mem_clk[0]` pin is generated by DDIO register; `mem_clk_n[0]` is generated by signal splitter.

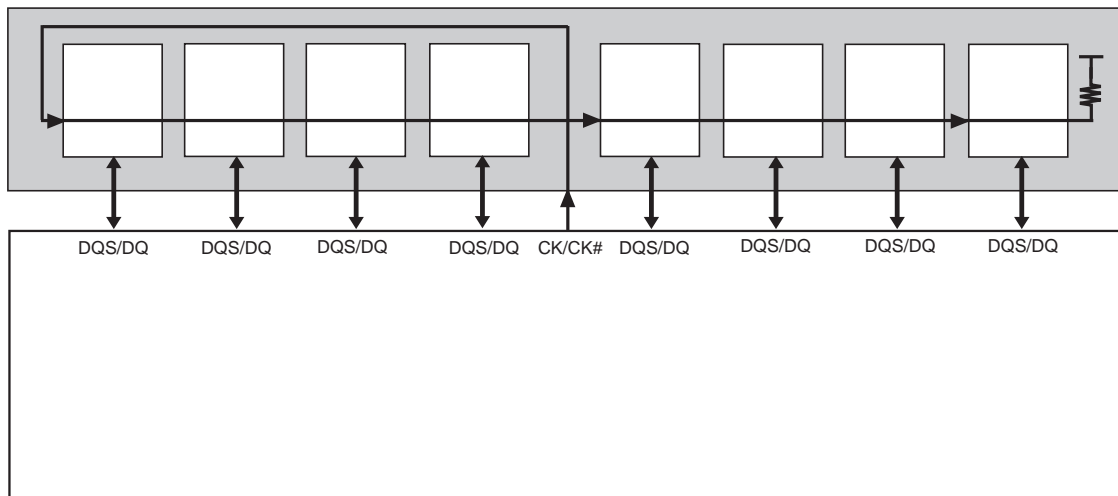
**Figure 3–2.** Mimic Path in Arria GX, Arria II GX, Cyclone III, Stratix II GX, and Stratix II Devices



## DDR3 SDRAM (with leveling)

The calibration process for the DDR3 SDRAM PHY (with leveling) assumes an interface in an unbuffered DIMM format, where the clock uses a fly-by termination, see Figure 3–3.

 ALTMEMPHY does not support DDR3 SDRAM RDIMM.

**Figure 3-3.** DDR3 SDRAM Unbuffered Module Clock Topology

With fly-by termination, each DDR3 SDRAM device on the DIMM sees the CK/CKn edges at different times. Therefore, the sequencer must adjust the clock to launch the DQS/DQSn and DQ signals so that it is appropriately aligned to the CK/CKn signals on each device.

The DDR3 SDRAM leveling sequencer during calibration writes to the following locations:

- Banks 0, 1, and 2
- Row 0
- All columns

Bank 0 is written to for the block training pattern and clock cycle calibration (DQ\_1T and AC\_1T). Bank 1 is written to for write deskew (DQ). Bank 2 is written to for write deskew (DM). For each bank, only row 0 is accessed. The number of columns accessed can vary, but you should avoid writing to all columns in these banks and row 0.

The calibration process for the DDR3 SDRAM PHY with leveling includes the following steps:

- “Step 1: Memory Device Initialization”
- “Step 2: Write Leveling”
- “Step 3: Write Training Patterns”
- “Step 4: Read Resynchronization”
- “Step 5: Address and Command Path Clock Cycle”
- “Step 7: Write Clock Path Setup”
- “Step 8: Prepare for User Mode”



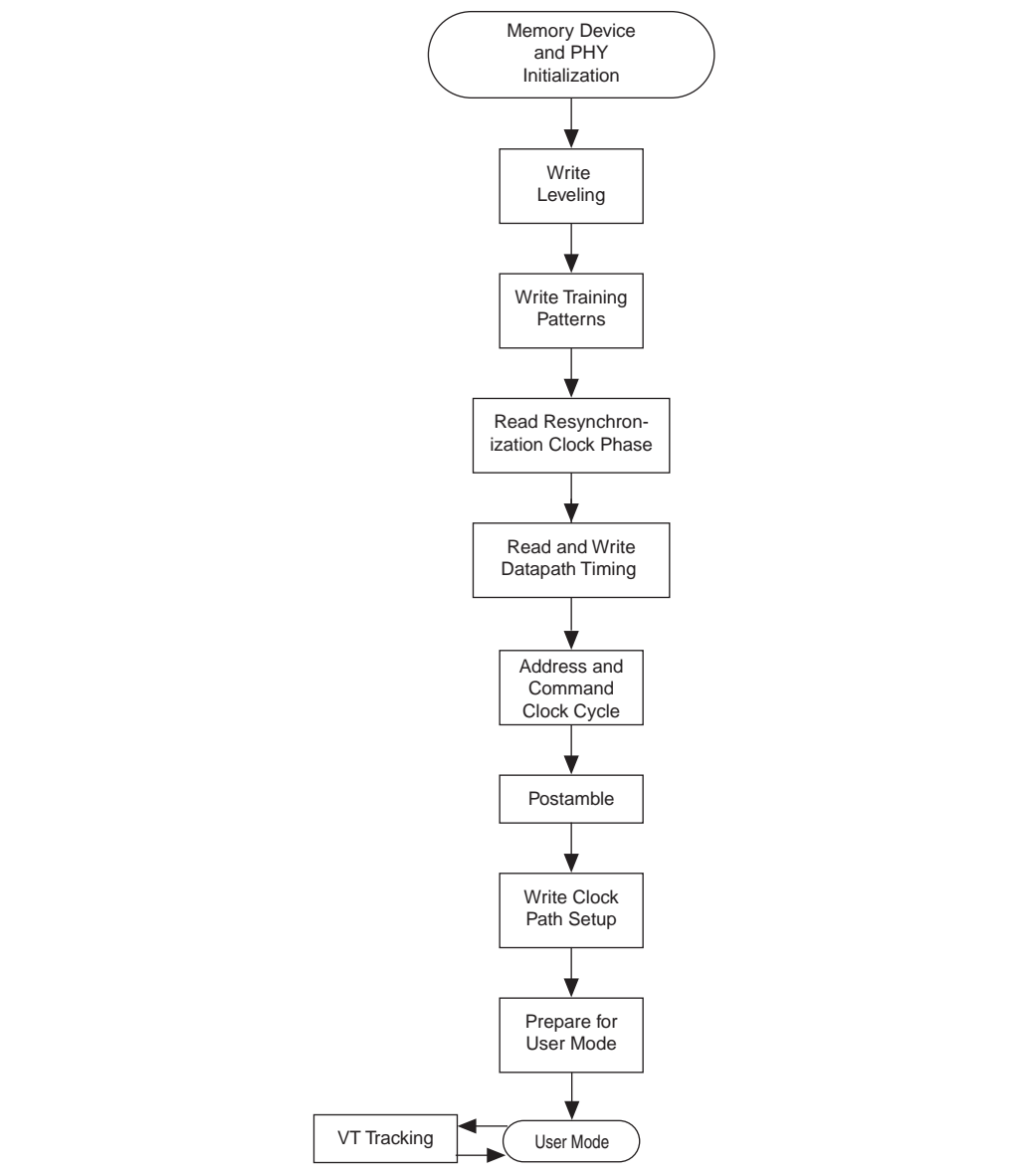
No steps can be bypassed. Therefore, even if you are using only one DDR3 SDRAM DIMM, all the calibration sequences are performed.

The calibration assumes that the skew for all the DQS launch times is one clock period maximum.

The VT tracking portion of the DDR3 SDRAM sequencer is similar to that of the DDR2/DDR SDRAM sequencer.

Figure 3-4 shows the calibration flow.

**Figure 3-4.** Calibration Flow—DDR3 SDRAM (with leveling)



**Step 1: Memory Device Initialization**

This step initializes the memory device per the DDR3 SDRAM specification. The initialization procedure includes resetting the memory device, setting up the mode registers, setting up memory device ODT setting and initializing the memory device DLL. Some of these settings may be different to those you set; however, these are changed to the correct values (at the end of calibration) in “[Step 8: Prepare for User Mode](#)”.

**Step 2: Write Leveling**

This step aligns the DQS edge with the CK edge at each memory device in the DIMM, which includes calibrating the write-leveling delay chains, programmable output delay chain, and using the  $t_{DQS}$ -margin register in the DDR3 SDRAM to monitor the relationship between the DQS edge and the CK edge. The calibration uses one DQ pin per DQS group (prime DQ) for write leveling calibration.

**Step 3: Write Training Patterns**

This step only allows you to write a pattern to be read later to calibrate the read path.

To satisfy the DDR3 SDRAM JEDEC specification, DQ is held constant during this step to ensure that there are no DQ timing violations. The DQS is then toggled, followed by a write command. A combination of burst length of four and burst length of eight operations ensure that it is correctly written. There are three different DQ patterns written in this step:

- All 0: 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
- All 1: 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF
- Mixed: 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0xFF

**Step 4: Read Resynchronization**

This step adjusts the phase of the resynchronization clock to determine the optimal clock phase that gives the most margin, similar to the resynchronization calibration done in DDR2/DDR SDRAM PHYs.

This step uses the read-leveling delay chain and the PLL reconfigurable clock output to adjust the resynchronization clock phase for each DQS group.

**Step 5: Address and Command Path Clock Cycle**

This step word-aligns the read data within and between each DQS group so that data can be presented in one clock cycle.

**Step 6: Postamble**

This step sets the correct clock cycle and clock phase shift for the postamble path. With the read resynchronization process, the sequencer can approximate when the postamble enable must be asserted. The sequencer then tries to incrementally assert the postamble enable signal (per DQS group) earlier until there is a read failure. This ensures the optimal clock phase for the system’s postamble enable signal.

### Step 7: Write Clock Path Setup

After the sequencer has the optimum settings for read capture and resynchronization setup, the sequencer calibrates the write datapath by configuring the alignment registers in the IOE and the DQ and DQS phase shift per DQS group. This step ensures that the write data can be presented on the same clock cycle from controller, but launched at the appropriate time for each DQS group to the DDR3 SDRAM memory devices.

### Step 8: Prepare for User Mode

In this step, the sequencer sends the calibrated write latency between command and write data (the `ctl_wlat` signal) to the controller. The PHY then applies user mode register settings and performs setup for periodic VT tracking.



Deskew is automatically enabled above 400.000 MHz.

### VT Tracking



For information on VT tracking for DDR3 SDRAM with leveling, refer to “VT Tracking” on page 3-4.

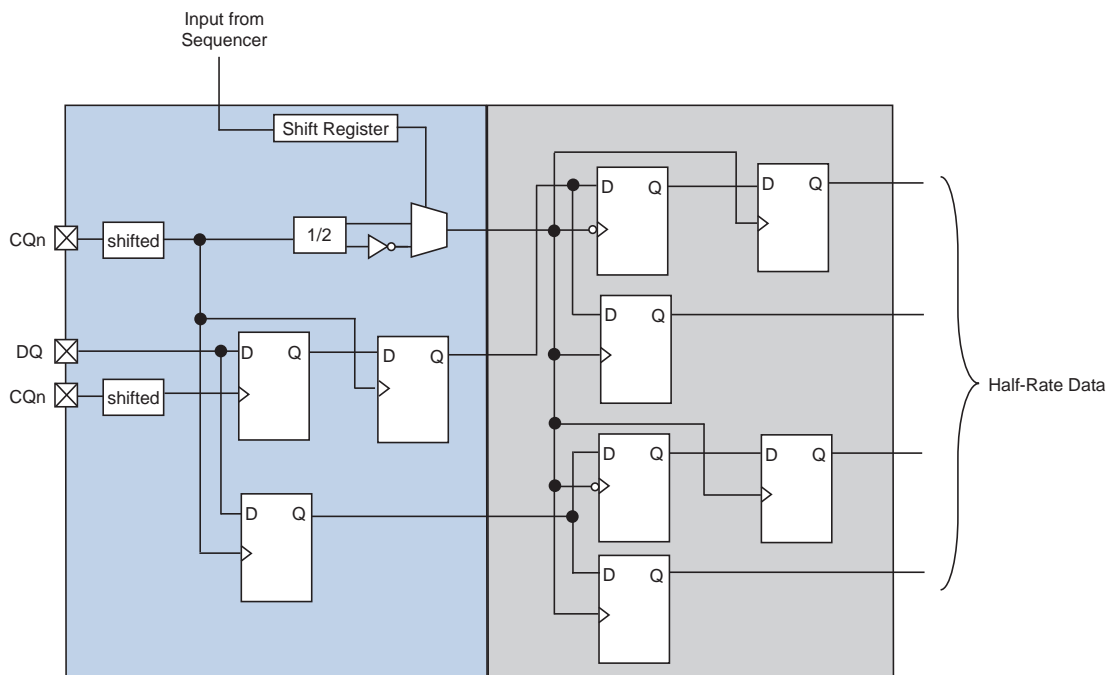
### Mimic Path



For information on mimic path for DDR3 SDRAM with leveling, refer to “Mimic Path” on page 3-4.

## QDRII+/QDRII SRAM

The calibration process of a QDRII+/QDRII SRAM device is considerably simpler than that of a calibration process for a DDR2/DDR SDRAM device. The calibration process involves selecting the right phase of the resynchronization clock to capture the read data at half rate. [Figure 3-5](#) shows the generation of the resynchronization clock which then clocks the HDR registers in the IOE. During calibration, the sequencer determines whether to use the half-rate clock or the inverted half-rate clock to capture the half-rate data.

**Figure 3-5.** Resynchronization Clock in QDRII+/QDRII SRAM ALTMEMPHY Megafunction

The clock CQ coming from the QDRII+/QDRII SRAM is delayed and is divided by two to generate a half-rate clock (`resync_clk_1x`). Data is captured on the rising edges of the shifted CQ and shifted CQn signals. There is one `resync_clk_1x` per DQS group. QDRII+/QDRII SRAM devices can only have one DQS group per device, which means that there is one `resync_clk_1x` signal associated with each memory device. This signal clocks the registers doing the full-rate to half-rate conversion. It also clocks the front side of the read datapath clock-crossing FIFO. There is one FIFO per DQS group (or memory device).

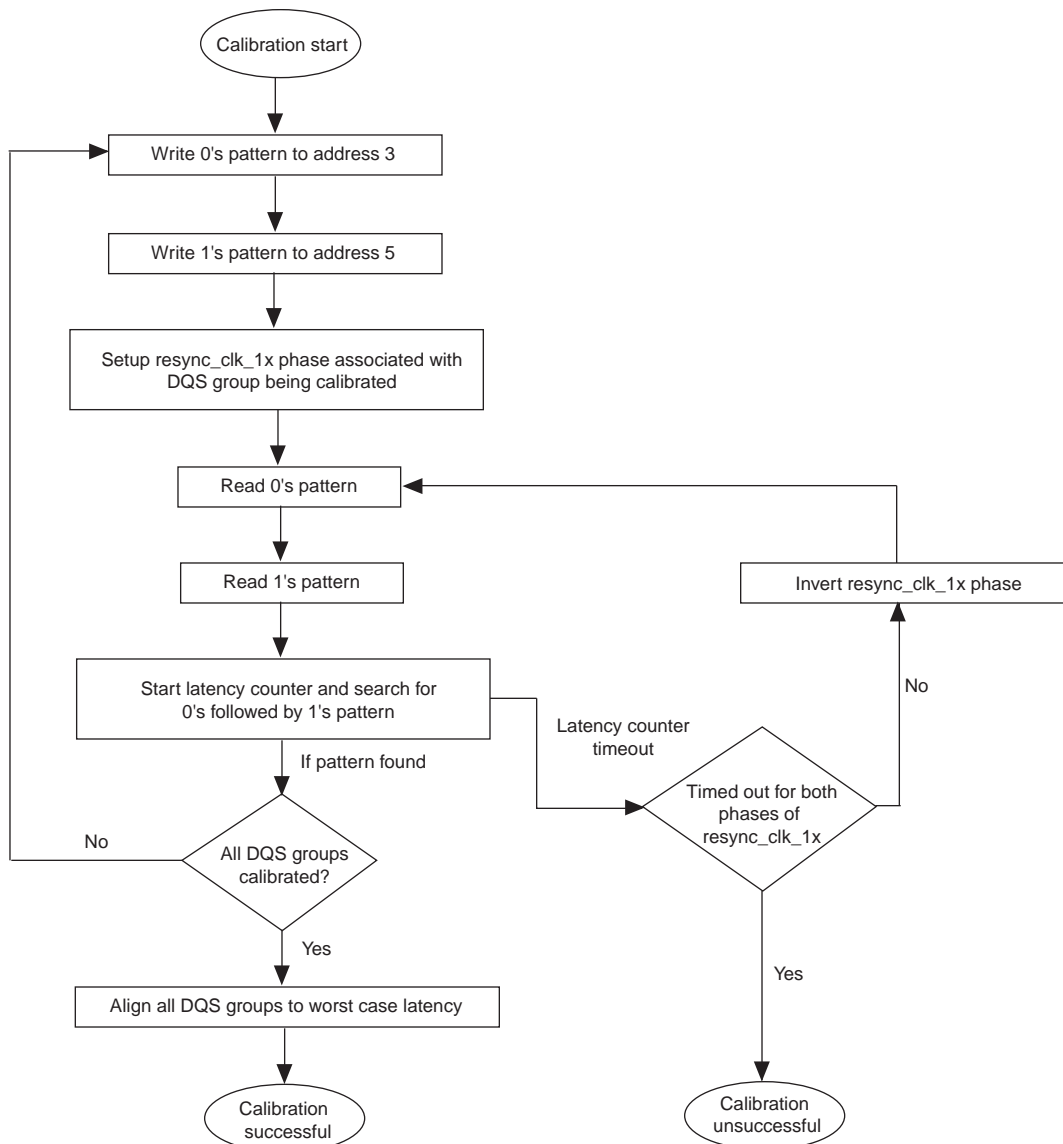
The `resync_clk_1x` signal can be inverted or not inverted. You can transfer data in the correct byte order with one of these options. The main objective of calibration is to find out whether the `resync_clk_1x` signal requires inversion, which is done by loading the shift register, see [Figure 3-5](#) (at most twice per QDRII+/QDRII SRAM device). Each memory device is calibrated one after the other.



### Calibration Process

Figure 3-6 shows the QDRII+/QDRII SRAM calibration flowchart.


Figure 3-6. QDRII+/QDRII SRAM Calibration Flowchart



During the QDRII+/QDRII SRAM calibration, the sequencer first writes all 0s to address space 3 in the external memory, followed by all 1s to address space 5. It then loads the scan chain for the first time, to program the first setting for `resync_clk_1x`.


The sequencer then starts reading 0s from address space 3 several times, followed by a single read from address space 5 and starts the latency counter. If a pattern of all 0s followed by all 1s is read before the latency counter reaches its time-out value (31 clock cycles), the latency value for that memory device is stored.

If all 0s followed by all 1s is not found when reading back from memory and the latency count has reached the time-out value, the sequencer loads the scan chains a second time to invert the `resync_clk_1x` signal. The sequencer then starts reading from address space 3 several times, followed by a single read from address space 5 as before and starts the latency counter again. If all 0s followed by all 1s are read back from memory, the latency value for that memory device is stored.

 The training pattern must be read back correctly on this second iteration (if it was not already read correctly on the first iteration). If it is not read back correctly, it indicates an underlying problem in the system.

The previous process is repeated until all memory devices are tested and a latency value obtained for each device.

The latency values found for the different devices are compared with each other. If necessary, they are aligned to the worst case latency (or to the user-requested deterministic latency value if this option is used), which done by adjusting address pointers in order to add latency to some of the read datapath RAMs inside the ALTMEMPHY megafunction until the latency associated with all of the memory devices is aligned to the worst case latency measured.

 You cannot have a latency difference of more than two PHY clock cycles between all the QDRII+/QDRII SRAM devices in non-deterministic latency mode.

When calibration has finished, the sequencer hands over control to the driver/user logic, and generates the `p_rdata_out_valid` flag to indicate when read data is valid. The sequencer also outputs the following signals upon completion of calibration:


- `p_ready`—Indicates completion of the calibration process (but does not mean calibration was successful). This signal is renamed as the `ctl_usr_mode_rdy` signal at the ALTMEMPHY top-level file.
- `p_calibration_successful`—Indicates calibration was successfully completed. This port is renamed `resynchronisation_successful` port at the ALTMEMPHY top-level file.
- `p_user_defined_latency_ok`—Indicates that the read latency requested by the user was achievable, when using deterministic latency. This port is not instantiated at the top-level of the file. Currently this signal exists at the `sequencer_wrapper` file level only.
- `p_detectedlatency`—Specifies the read latency achieved in `phy_clk` clock cycles. This port is renamed `ctl_rlat` port at the ALTMEMPHY top-level file.

VT tracking is not required because the read strobe from the QDRII+/QDRII SRAM memory is continuous. So all registers in the I/O to the read RAM path are clocked using a clock that is derived from the QDRII+/QDRII SRAM read clock.

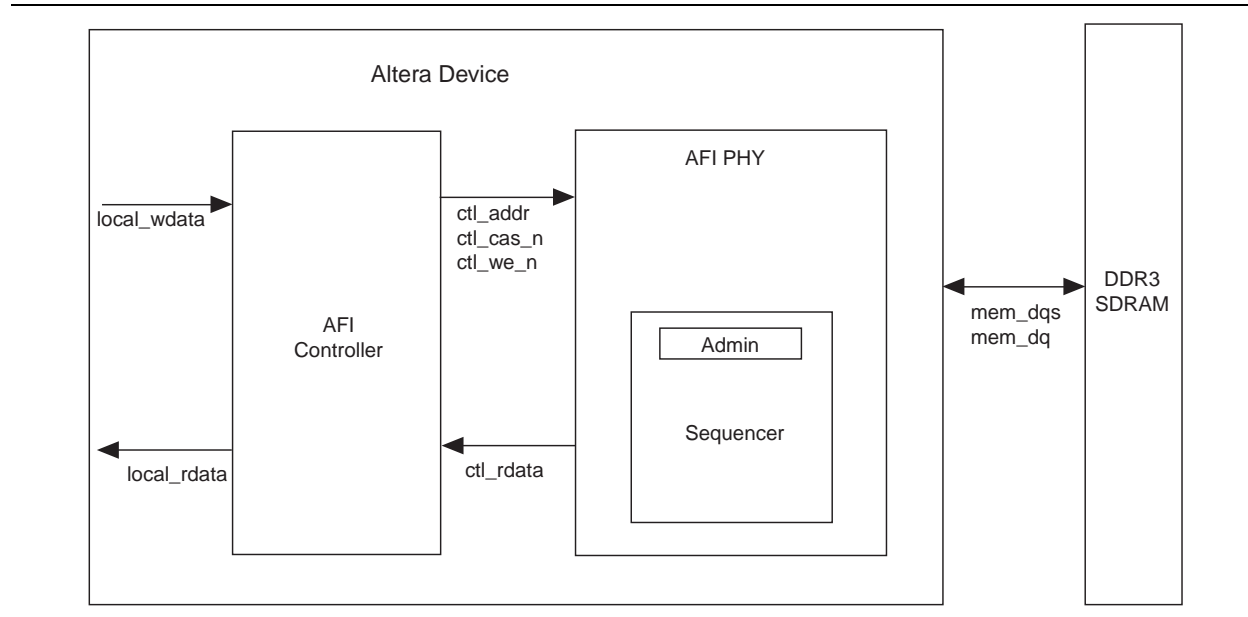
## PHY-to-Controller Interfaces

The following section describes the typical modules that are connected to the ALTMEMPHY variation and the port name prefixes each module uses. This section describes the AFI; for non-AFI information, refer to [Appendix A, Non-AFI](#).

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 includes an administration block that configures the memory for calibration and performs necessary mode registers accesses to configure the memory as required (these calibration processes are different). [Figure 3-7](#) shows an overview of the connections between the PHY, the controller, and the memory device.

 Altera recommends that you use the AFI for new designs.

**Figure 3-7.** AFI PHY Connections



For half-rate designs, the address and command signals in the ALTMEMPHY megafunction 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, drive both input bits (of the address and command signal) identically in half-rate designs.

For DDR3 SDRAM with the AFI, the read and write control signals are on a per-DQS group basis. The controller can calibrate and use a subset of the available DDR3 SDRAM devices. For example, two devices out of a 64- or 72-bit DIMM, for better debugging mechanism.

For half-rate designs, the AFI allows the controller to issue reads and writes that are aligned to either half-cycle of the half-rate `phy_clk`, which means that the datapaths can support multiple data alignments—word-unaligned and word-aligned writes and reads. [Figure 3-8](#) and [Figure 3-9](#) display the half-rate write operation.

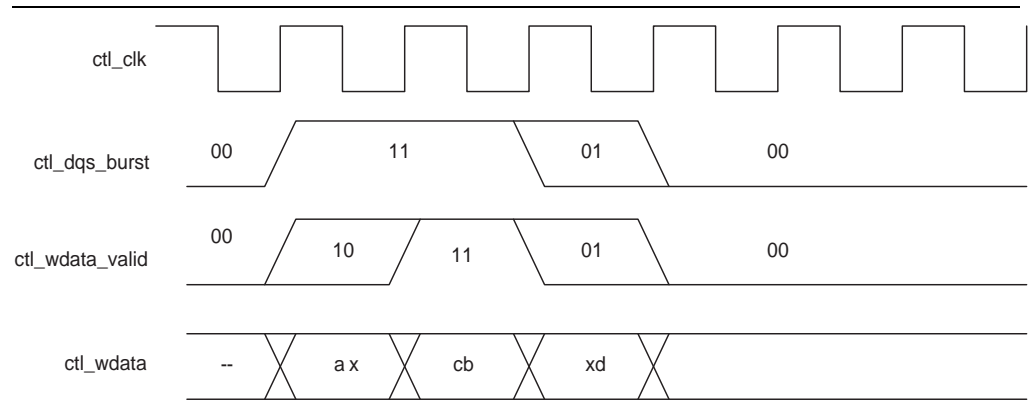
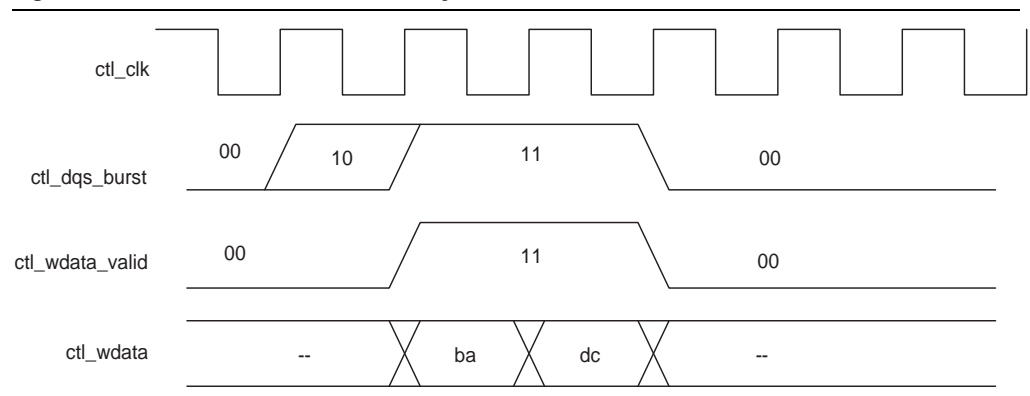
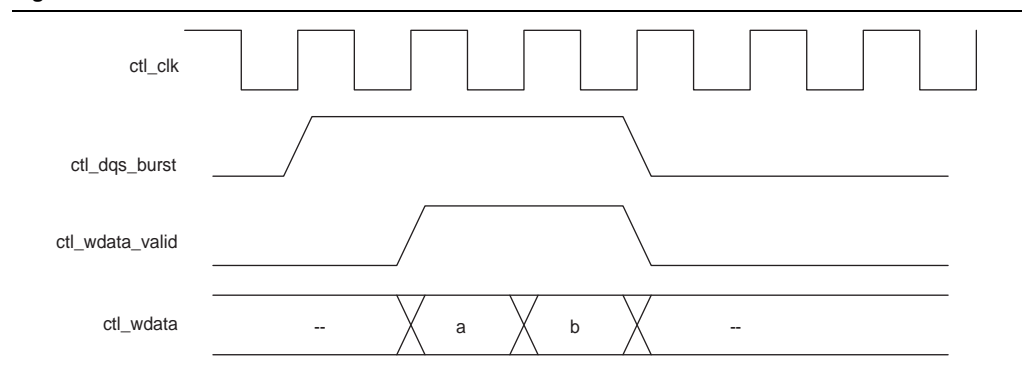
**Figure 3-8.** Half-Rate Write with Word-Unaligned Data**Figure 3-9.** Half-Rate Write with Word-Aligned Data

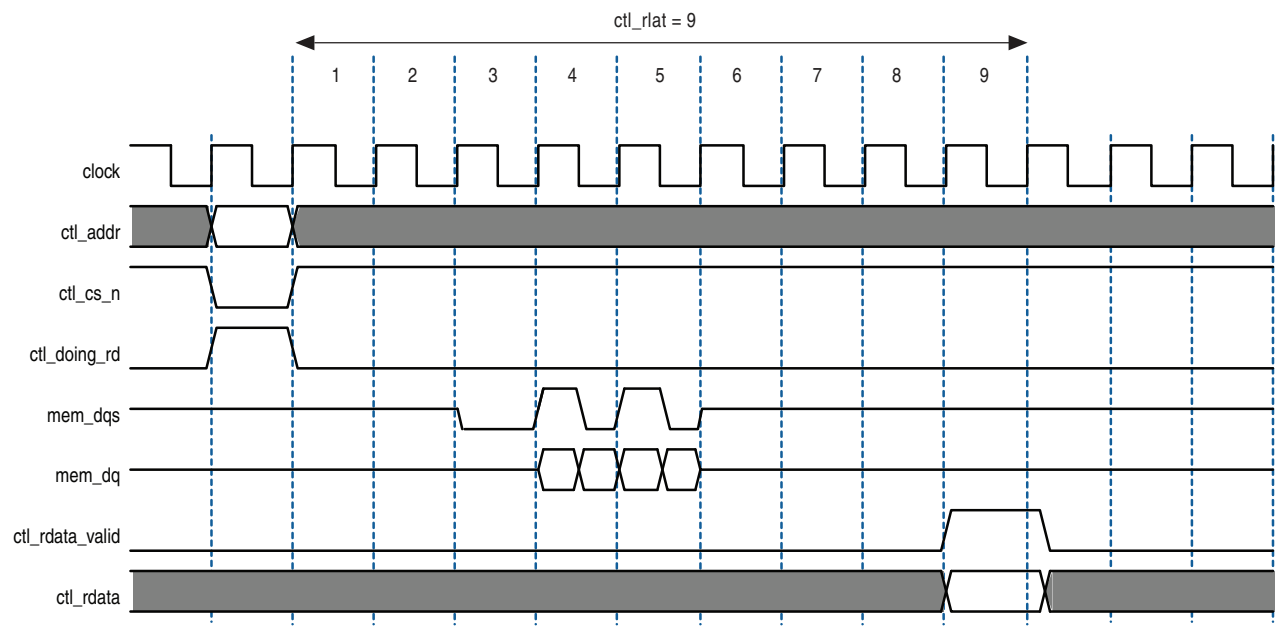
Figure 3-10 shows a full-rate write.

**Figure 3-10.** Full-Rate Write

After calibration completes, the sequencer sends the write latency in number of clock cycles to the controller.

Figure 3-11 shows full-rate reads; Figure 3-12 shows half-rate reads.

**Figure 3-11.** Full-Rate Reads



**Figure 3-12.** Half-Rate Reads

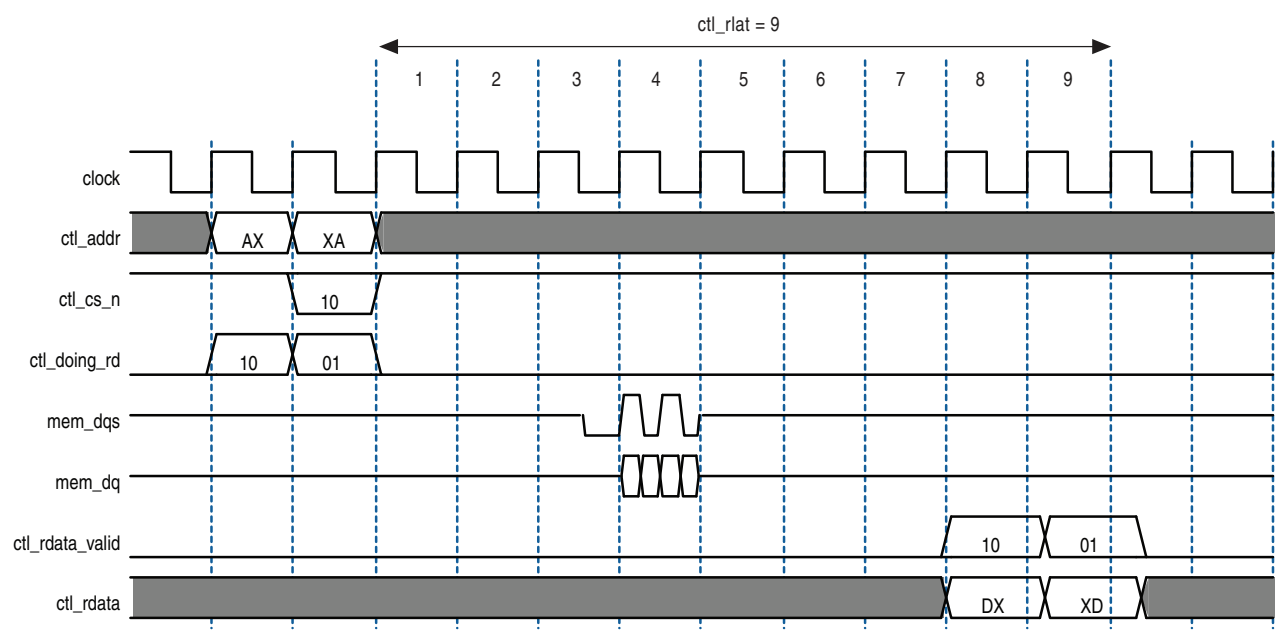




Figure 3-13 and Figure 3-14 show word-aligned writes and reads. In the following read and write examples the data is written to and read from the same address. In each example, `ctl_rdata` and `ctl_wdata` are aligned with controller clock (`ctl_clk`) cycles. All the data in the bit vector is valid at once. For comparison, refer Figure 3-15 and Figure 3-16 that show the word-unaligned writes and reads.


 The `ctl_doing_rd` is represented as a half-rate signal when passed into the PHY. Therefore, the lower half of this bit vector represents one memory clock cycle and the upper half the next memory clock cycle. Figure 3-16 on page 3-21 shows separated word-unaligned reads as an example of two `ctl_doing_rd` bits are different. Therefore, for each x16 device, at least two `ctl_doing_rd` bits need to be driven, and two `ctl_rdata_valid` bits need to be interpreted.


The AFI has the following conventions:


- With the AFI, high and low signals are combined in one signal, so for a single chip select (`ctl_cs_n`) interface, `ctl_cs_n[1:0]`, where location 0 appears on the memory bus on one `mem_clk` cycle and location 1 on the next `mem_clk` cycle.

 This convention is maintained for all signals so for an 8 bit memory interface, the write data (`ctl_wdata`) signal is `ctl_wdata[31:0]`, where the first data on the DQ pins is `ctl_wdata[7:0]`, then `ctl_wdata[15:8]`, then `ctl_wdata[23:16]`, then `ctl_wdata[31:24]`.

- Word-aligned and word-unaligned reads and writes have the following definitions:
  - Word-aligned for the single chip select is active (low) in location 1 (`_1`). `ctl_cs_n[1:0] = 01` when a write occurs. This alignment is the easiest alignment to design with.
  - Word-unaligned is the opposite, so `ctl_cs_n[1:0] = 10` when a read or write occurs and the other control and data signals are distributed across consecutive `ctl_clk` cycles.

 The Altera high-performance controllers use word-aligned data only.

 The timing analysis script does not support word-unaligned reads and writes.

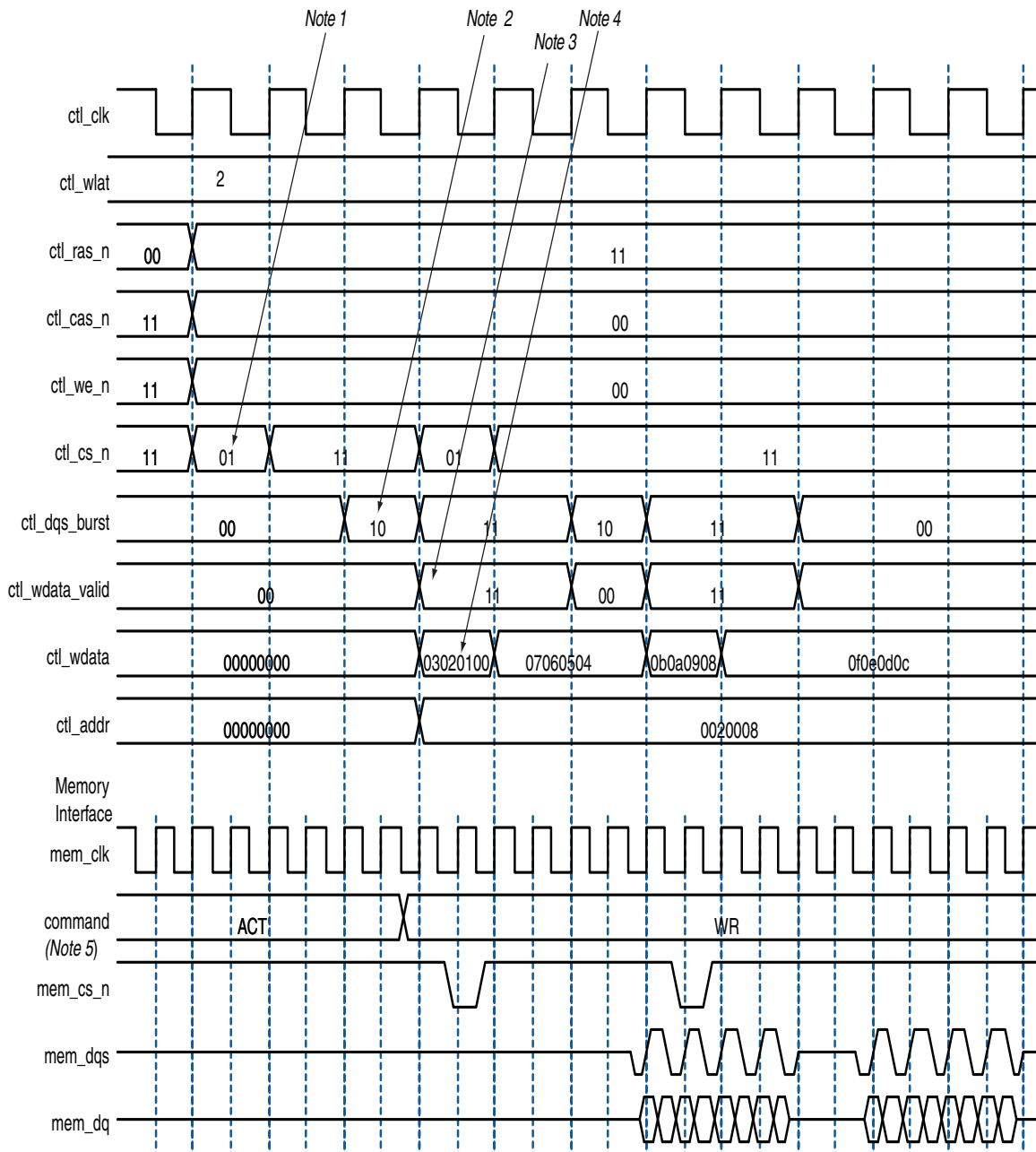
 Word-unaligned reads and writes are only supported by RTL on Stratix III and Stratix IV devices.

- Spaced reads and writes have the following definitions:
  - Spaced writes—write commands separated by a gap of one controller clock (`ctl_clk`) cycle
  - Spaced reads—read commands separated by a gap of one controller clock (`ctl_clk`) cycle

Figure 3-13 through Figure 3-16 assume the following general points:

- The burst length is four. A DDR2 SDRAM is used—the interface timing is identical for DDR3 devices.
- An 8-bit interface with one chip select.
- The data for one controller clock (`ctl_clk`) cycle represents data for two memory clock (`mem_clk`) cycles (half-rate interface).

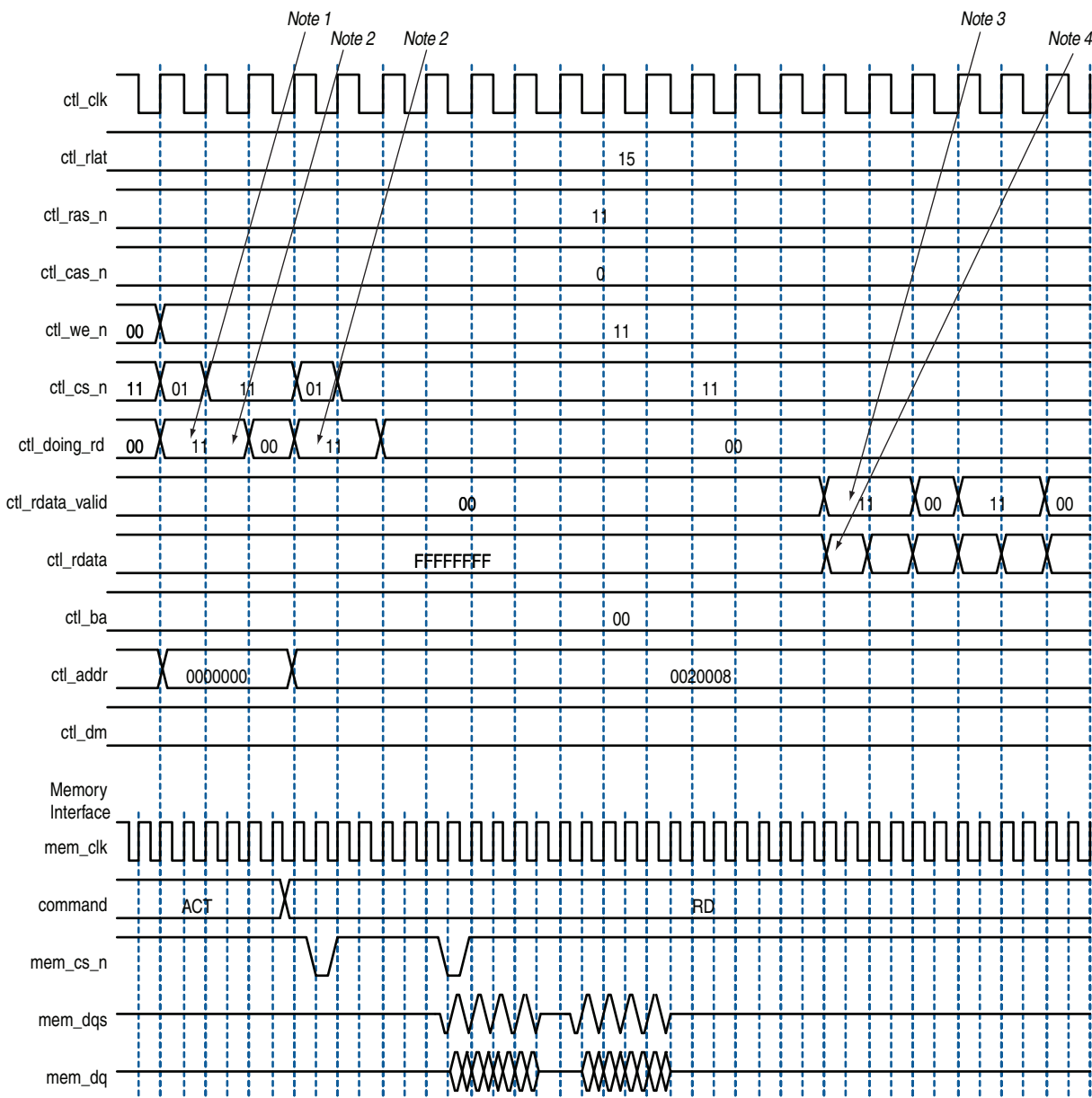
Figure 3-13. Word-Aligned Writes

**Notes to Figure 3-13:**

- (1) To show the even alignment of `ctl_cs_n`, expand the signal (this convention applies for all other signals).
- (2) The `ctl_dqs_burst` must go high one memory clock cycle before `ctl_wdata_valid`. Compare with the word-unaligned case.
- (3) The `ctl_wdata_valid` is asserted two `ctl_wlat` controller clock (`ctl_clk`) cycles after chip select (`ctl_cs_n`) is asserted. The `ctl_wlat` indicates the required write latency in the system. The value is determined during calibration and is dependant upon the relative delays in the address and command path and the write datapath in both the PHY and the external DDR SDRAM subsystem. The controller must drive `ctl_cs_n` and then wait `ctl_wlat` (two in this example) `ctl_clks` before driving `ctl_wdata_valid`.
- (4) Observe the ordering of write data (`ctl_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 3-14. Word-Aligned Reads

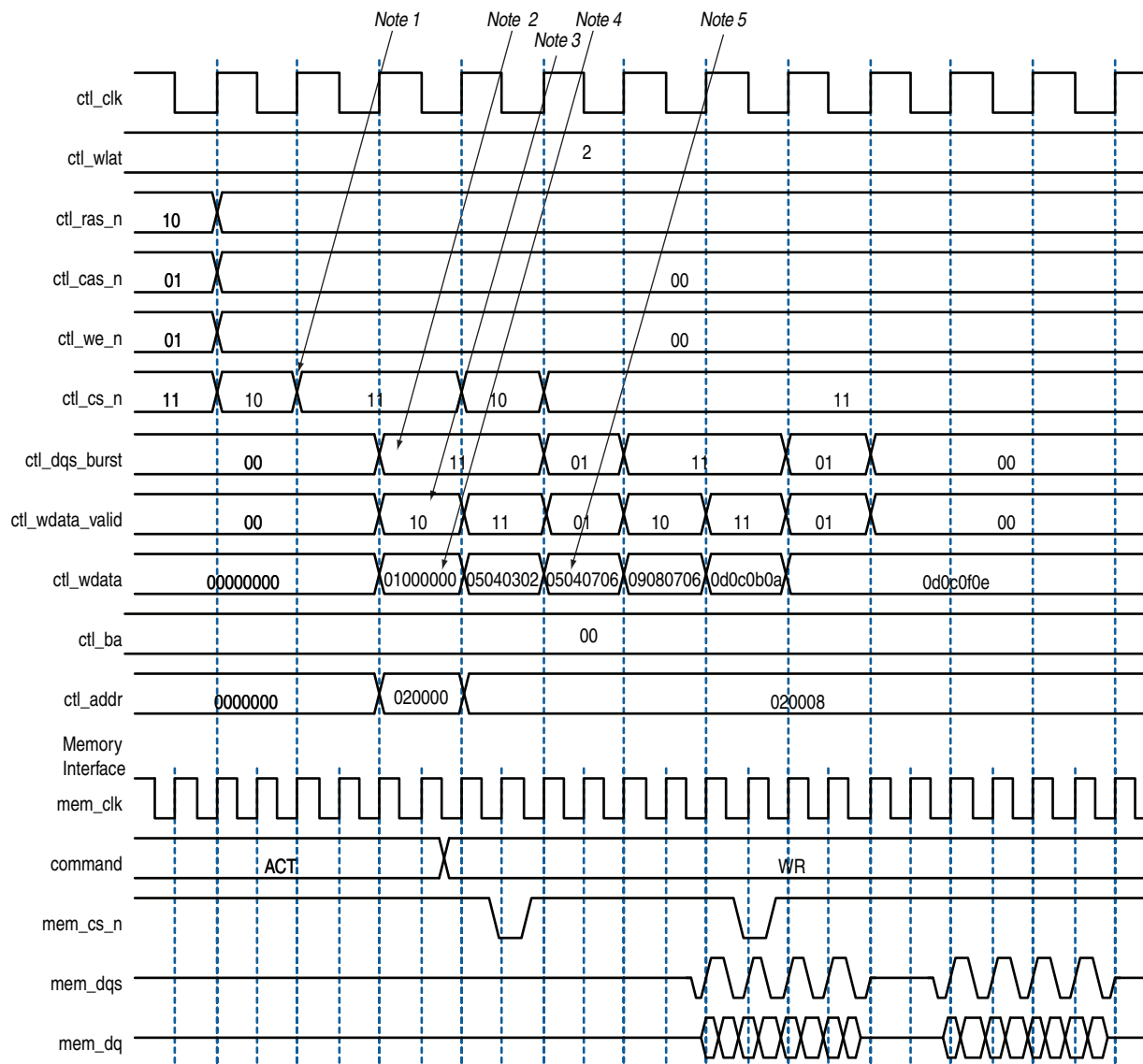


Notes to Figure 3-14:

- (1) For AFI, `ctl_doing_rd` is required to be asserted one memory clock cycle before chip select (`ctl_cs_n`) is asserted. In the half-rate `ctl_clk` domain, this requirement manifests as the controller driving 11 (as opposed to the 01) on `ctl_doing_rd`.
- (2) AFI requires that `ctl_doing_rd` is driven for the duration of the read. In this example, it is driven to 11 for two half-rate `ctl_clks`, which equates to driving to 1, for the four memory clock cycles of this four-beat burst.
- (3) The `ctl_rdata_valid` returns 15 (`ctl_rlat`) controller clock (`ctl_clk`) cycles after `ctl_doing_rd` is asserted. Returned is when the `ctl_rdata_valid` signal is observed at the output of a register within the controller. A controller can use the `ctl_rlat` value to determine when to register to returned data, but this is unnecessary as the `ctl_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.

Figure 3-15 and Figure 3-16 show spaced word-unaligned writes and reads.

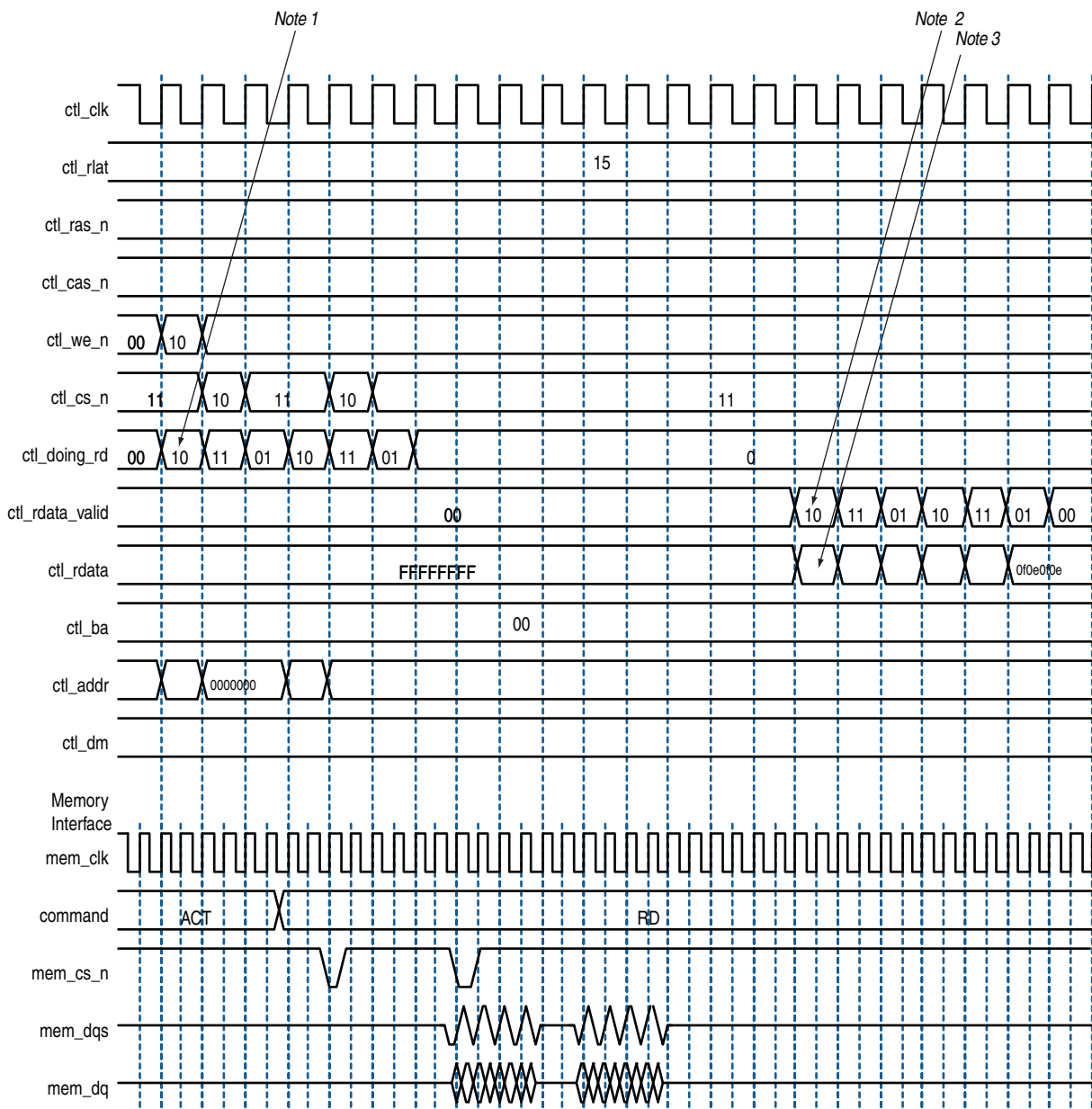
**Figure 3-15.** Word-Unaligned Writes



**Notes to Figure 3-15:**

- (1) Alternative word-unaligned chip select (**ctl\_cs\_n**).
- (2) As with word-aligned writes, **ctl\_dqs\_burst** is asserted one memory clock cycle before **ctl\_wdata\_valid**. You can see **ctl\_dqs\_burst** is 11 in the same cycle where **ctl\_wdata\_valid** is 10. The LSB of these two becomes the first value the signal takes in the **mem\_clk** domain. You can see that **ctl\_dqs\_burst** has the necessary one **mem\_clk** cycle lead on **ctl\_wdata\_valid**.
- (3) The latency between **ctl\_cs\_n** being asserted and **ctl\_wdata\_valid** going high is effectively **ctl\_wlat** (in this example, two) controller clock (**ctl\_clk**) cycles. This can be thought of in terms of relative memory clock (**mem\_clk**) cycles, in which case the latency is four **mem\_clk** cycles.
- (4) Only the upper half is valid (as the **ctl\_wdata\_valid** signal demonstrates, there is one **ctl\_wdata\_valid** bit to two 8-bit words). The write data bits go out on the bus in order, least significant byte first. So for a continuous burst of write data on the DQ pins, the most significant half of write data is used, which goes out on the bus last and is therefore contiguous with the following data. The converse is true for the end of the burst. Write data is spread across three controller clock (**ctl\_clk**) cycles, but still only four memory clock (**mem\_clk**) cycles. However, in relative memory clock cycles the latency is equivalent in the word-aligned and word-unaligned cases.
- (5) The 0504 here is residual from the previous clock cycle. In the same way that only the upper half of the write data is used for the first beat of the write, only the lower half of the write data is used in the last beat of the write. These upper bits can be driven to any value in this alignment.

Figure 3-16. Word-Unaligned Reads



Notes to Figure 3-16:


- (1) Similar to word-aligned reads, **ctl\_doing\_rd** is asserted one memory clock cycle before chip select (**ctl\_cs\_n**) is asserted, which for a word-unaligned read is in the previous controller clock (**ctl\_clk**) cycle. In this example the **ctl\_doing\_rd** signal is now spread over three controller clock (**ctl\_clk**) cycles, the high bits in the sequence '10', '11', '01', '10', '11', '01' providing the required four memory clock cycles of assertion for **ctl\_doing\_rd** for the two 4-beat reads in the full-rate memory clock domain, '011110', '011110'.
- (2) The return pattern of **ctl\_rdata\_valid** is a delayed version of **ctl\_doing\_rd**. Advertised read latency (**ctl\_rlat**) is the number of controller clock (**ctl\_clk**) cycles delay inserted between **ctl\_doing\_rd** and **ctl\_rdata\_valid**.
- (3) The read data (**ctl\_rdata**) is spread over three controller clock cycles and in the pointed to vector only the upper half of the **ctl\_rdata** bit vector is valid (denoted by **ctl\_rdata\_valid**).

## Half-Rate Read and Write Data Mapping

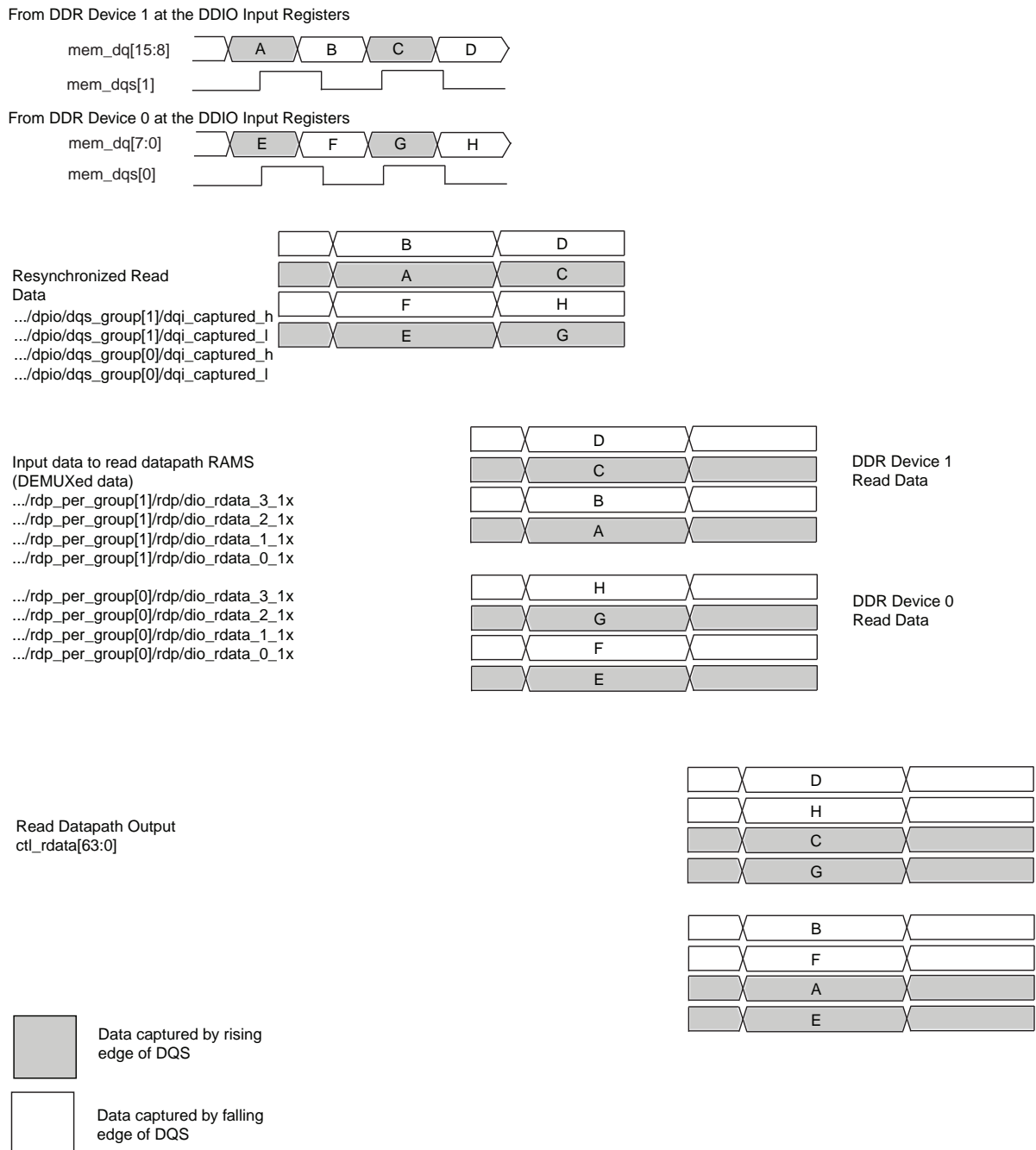
The following sections discuss read and write data mapping for half-rate designs.

## Read Data Mapping

In this example (Figure 3-17), the memory interface consists of two 8-bit wide memory devices, resulting in a memory interface 16-bits wide.

 Read and write data mapping for a full-rate controller is the same as a half-rate controller. However, a full-rate controller's local interface data width is half that of the local interface data width of a half-rate controller.

**Figure 3-17.** Read Data Ordering



1. Data B and F (each of 8-bits wide) is captured during the first falling edge of DQS, and data A and E (each of 8-bits wide) is captured during the first rising edge of DQS.
2. Now the resynchronization registers resynchronize the captured data to the resynchronization clock (`resync_clk_2x`) as `rdata_resynched_2x` (Figure 3-17) of width 32 bits. During the first cycle you get BAFE, and during the second cycle you get DCHG.


 The data changes with each clock cycle of the resynchronization clock.

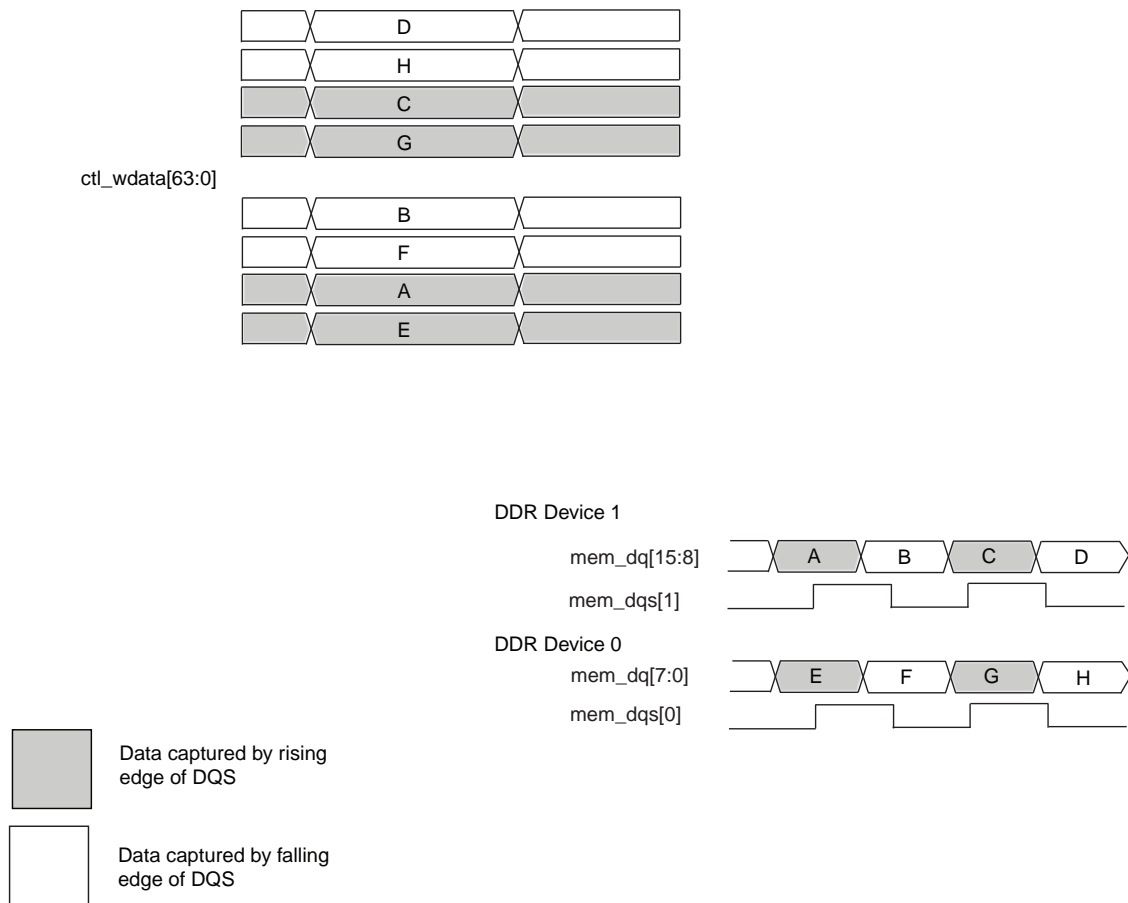
3. Because the FIFO is 32-bits wide on the write side and 64-bits wide on the read side, the two sets (DCHG and BAFE) of 32-bit data forms one 64-bit wide data on the read side. The data (`ram_rdata_1x`) from the FIFO is arranged as DCBAHGFE and is one 64-bit wide data.
4. Now the 64-bit wide data (`ram_rdata_1x`) is presented at the local interface in the form of DHCGBFAE. The user-data interface to the read ports of the ALTMEMPHY megafunction can be thought of as being split into four words, each representing an edge of DQS. Figure 3-17 shows that the LSB is the first in time seen on a DQ pin.

## Write Data Mapping

The write data at the controller, which is  $4n$ -bit wide, is converted to  $n$  bits at the memory interface. Figure 3-18 shows how `ctl_mem_wdata` is mapped into the `mem_dq` by using a 64-bit `ctl_mem_wdata` as an example.

This conversion is the reverse process of the steps required for the read conversion, see “Read Data Mapping” on page 3-22.

 Read and write data mapping for a full-rate controller is the same as a half-rate controller. However, a full-rate controller’s local interface data width is half that of the local interface data width of a half-rate controller.

**Figure 3-18.** Data Mapping And Write Datapath

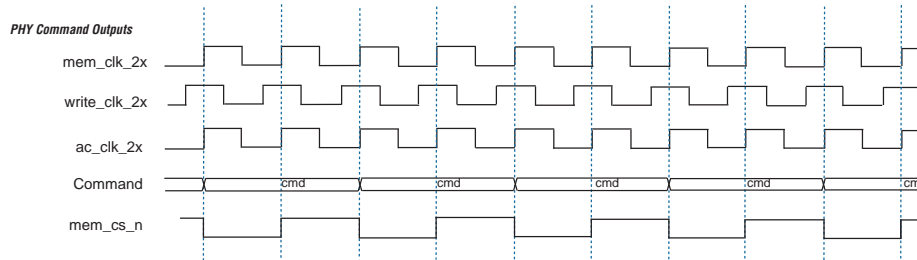
## Half-Rate Address and Command Clocking

Figure 3-19 through Figure 3-22 show the assertion of the command signals with respect to different `ac_clk_2x` phase settings when shared with the `write_clk_2x` or `mem_clk_2x` clock in ALTMEMPHY interfaces that target for Arria GX, Arria II GX, Stratix II/Stratix II GX, Cyclone III and HardCopy II devices.

In Stratix III and Stratix IV devices, the ALTMEMPHY uses a dedicated PLL output for address and command, whose phase shift must be between 180° and 359°.

Figure 3-21 also applies when you choose the dedicated address and command clock to have 180°. Figure 3-22 applies when you choose the dedicated address and command clock to have 270°.

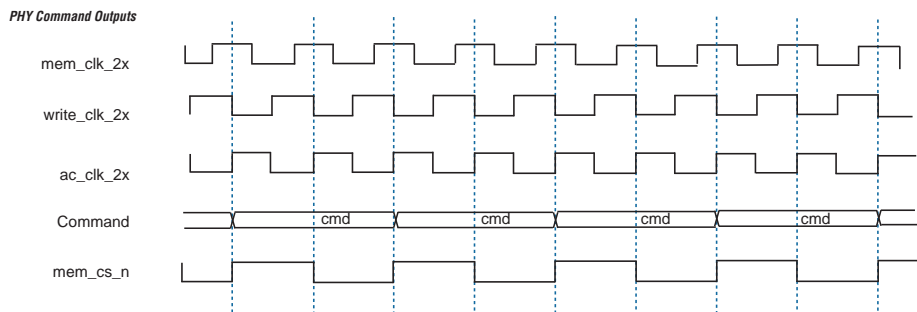
**Figure 3-19.** Address and Command Clock (0° Phase) (Note 1)



**Note to Figure 3-19:**

(1) For the 0° phase address and command clock, the ac\_clk\_2x signal is aligned with mem\_clk\_2x signal.

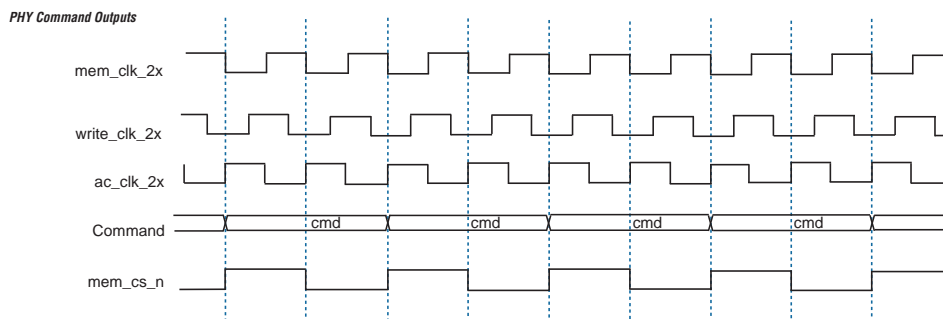
**Figure 3-20.** Address and Command Clock (90° Phase) (Note 1)



**Note to Figure 3-20:**

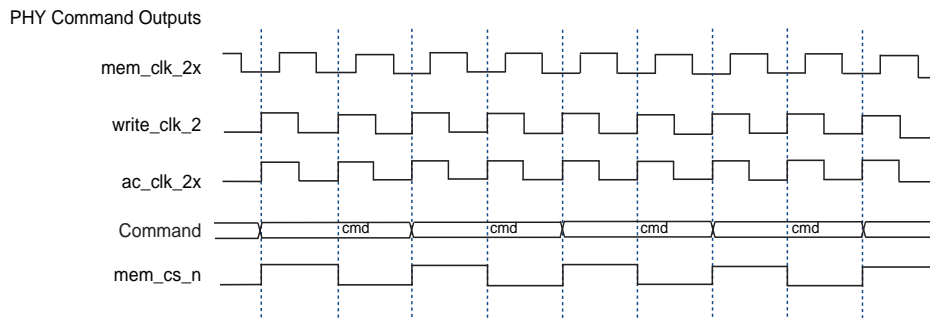
(1) For the 90° phase address and command clock, the ac\_clk\_2x signal is the inverted version of the write\_clk\_2x signal.

**Figure 3-21.** Address and Command Clock (180° Phase) (Note 1)



**Note to Figure 3-21:**

(1) For the 180° phase address and command clock, the ac\_clk\_2x signal is the inverted version of the mem\_clk\_2x signal.

**Figure 3-22.** Address and Command Clock (270° Phase) *(Note 1)***Note to Figure 3-22:**

(1) For the 270° phase address and command clock, the `ac_clk_2x` signal is aligned with `write_clk_2x` signal.

Refer to the *External Memory Interfaces* chapter in the respective device family handbook for help in selecting I/O pins for the address and command signals.

## Ports

This section describes the ALMEMPHY megafunction ports for AFI variants and for QDRII+/QDR SRAM; for non-AFI information, refer to [Appendix A, Non-AFI](#).

[Table 3-1](#) through [Table 3-8](#) show the ports.

Ports with the prefix `mem_` connect the PHY with the memory device; ports with the prefix `ctl_` connect the PHY with the controller.

The port lists include the following signal groups:

- I/O interface to the SDRAM devices
- Clocks and resets
- External DLL signals
- User-mode calibration OCT control
- Write data interface
- Read data interface
- Address and command interface
- Calibration control and status interface
- Debug interface

For more information about the simulation waveforms of a DDR2 SDRAM high-performance controller, refer to the figures in the Interface and Signals section in the *DDR and DDR2 SDRAM High-Performance Controller User Guide*.



**Table 3-1.** Interface to the SDRAM Devices (Note 1)

Signal Name	Type	Width (2)	Description
mem_addr	output	MEM_IF_ROWADDR_WIDTH	The memory row and column address bus.
mem_ba	output	MEM_IF_BANKADDR_WIDTH	The memory bank address bus.
mem_cas_n	output	1	The memory column address strobe.
mem_cke	output	MEM_IF_CS_WIDTH	The memory clock enable.
mem_clk	bidir	MEM_IF_CLK_PAIR_COUNT	The memory clock, positive edge clock. (3)
mem_clk_n	bidir	MEM_IF_CLK_PAIR_COUNT	The memory clock, negative edge clock.
mem_cs_n	output	MEM_IF_CS_WIDTH	The memory chip select signal.
mem_dm	output	MEM_IF_DM_WIDTH	The optional memory DM bus.
mem_dq	bidir	MEM_IF_DWIDTH	The memory bidirectional data bus.
mem_dqs	bidir	MEM_IF_DWIDTH/ MEM_IF_DQ_PER_DQS	The memory bidirectional data strobe bus.
mem_dqsn	bidir	MEM_IF_DWIDTH/ MEM_IF_DQ_PER_DQS	The memory bidirectional data strobe bus.
mem_odt	output	MEM_IF_CS_WIDTH	The memory on-die termination control signal.
mem_ras_n	output	1	The memory row address strobe.
mem_reset_n	output	1	The memory reset signal.
mem_we_n	output	1	The memory write enable signal.
<b>Notes to Table 3-1:</b>			
(1) Connected to I/O pads.			
(2) Refer to Table 3-9 for parameter deescription.			
(3) Output is for memory device, and input path is fed back to ALTMEMPHY megafunction for VT tracking.			

**Table 3-2.** AFI Signals (Part 1 of 4)

Signal Name	Type	Width (1)	Description
<b>Clocks and Resets</b>			
pll_ref_clk	input	1	The reference clock input to the PHY PLL.
global_reset_n (2)	input	1	Active-low global reset for PLL and all logic in the PHY. A level set reset signal, which causes a complete reset of the whole system. The PLL may maintain some state information.
soft_reset_n (2)	input	1	Edge detect reset input intended for SOPC Builder use or to be controlled by other system reset logic. Causes a complete reset of PHY, but not the PLL used in the PHY.

**Table 3-2.** AFI Signals (Part 2 of 4)

Signal Name	Type	Width (1)	Description
reset_request_n (2)	output	1	Directly connected to the locked output of the PLL and is intended for optional use either by automated tools such as SOPC Builder or could be manually ANDed with any other system-level signals and combined with any edge detect logic as required and then fed back to the global_reset_n input.  Reset request output that indicates when the PLL outputs are not locked. Use this as a reset request input to any system-level reset controller you may have. This signal is always low while the PLL is locking (but not locked), and so any reset logic using it is advised to detect a reset request on a falling-edge rather than by level detection.
ctl_clk	output	1	Half-rate clock supplied to controller and system logic. The same signal as the non-AFI phy_clk.
ctl_reset_n	output	1	Reset output on ctl_clk clock domain.
<b>Other Signals</b>			
aux_half_rate_clk	output	1	In half-rate designs, a copy of the phy_clk_1x signal that you can use in other parts of your design, same as phy_clk port.
aux_full_rate_clk	output	1	In full-rate designs, a copy of the mem_clk_2x signal that you can use in other parts of your design.
aux_scan_clk	output	1	Low frequency scan clock supplied primarily to clock any user logic that interfaces to the PLL and DLL reconfiguration interfaces.
aux_scan_clk_reset_n	output	1	This reset output asynchronously asserts (drives low) when global_reset_n is asserted and de-assert (drives high) synchronous to aux_scan_clk when global_reset_n is de-asserted. It allows you to reset any external circuitry clocked by aux_scan_clk.
<b>Write Data Interface</b>			
ctl_dqs_burst	input	MEM_IF_DQS_WIDTH × DWIDTH_RATIO / 2	When asserted, mem_dqs is driven. The ctl_dqs_burst signal must be asserted before ctl_wdata_valid and must be driven for the correct duration to generate a correctly timed mem_dqs signal.
ctl_wdata_valid	input	MEM_IF_DQS_WIDTH × DWIDTH_RATIO / 2	Write data valid. Generates ctl_wdata and ctl_dm output enables.
ctl_wdata	input	MEM_IF_DWIDTH × DWIDTH_RATIO	Write data input from the controller to the PHY to generate mem_dq.
ctl_dm	input	MEM_IF_DM_WIDTH × DWIDTH_RATIO	DM input from the controller to the PHY.
ctl_wlat	output	5	Required write latency between address/command and write data that is issued to ALTMEMPHY controller local interface.
<b>Read Data Interface</b>			

**Table 3-2.** AFI Signals (Part 3 of 4)

Signal Name	Type	Width (1)	Description
ctl_doing_rd	input	$MEM\_IF\_DQS\_WIDTH \times DWIDTH\_RATIO / 2$	Doing read input. Indicates that the DDR3 SDRAM controller is currently performing a read operation. The controller generates <code>ctl_doing_rd</code> to the <code>ALTMEMPHY</code> megafunction. The <code>ctl_doing_rd</code> signal is asserted for one <code>phy_clk</code> cycle for every read command it issues. If there are two read commands, <code>ctl_doing_rd</code> is asserted for two <code>phy_clk</code> cycles. The <code>ctl_doing_rd</code> signal also enables the capture registers and generates the <code>ctl_mem_rdata_valid</code> signal. The <code>ctl_doing_rd</code> signal should be issued at the same time the read command is sent to the <code>ALTMEMPHY</code> megafunction (Figure 7-1 on page 7-3).
ctl_rdata	output	$DWIDTH\_RATIO \times MEM\_IF\_DWIDTH$	Read data from the PHY to the controller.
ctl_rdata_valid	output	$DWIDTH\_RATIO/2$	Read data valid indicating valid read data on <code>ctl_rdata</code> . This signal is two-bits wide (as only half-rate or $DWIDTH\_RATIO = 4$ is supported) to allow controllers to issue reads and writes that are aligned to either the half-cycle of the halfrate clock.
ctl_rlat	output	<code>READ_LAT_WIDTH</code>	Contains the number of clock cycles between the assertion of <code>ctl_doing_rd</code> and the return of valid read data ( <code>ctl_rdata</code> ). This is unused by the Altera high-performance controllers do not use <code>ctl_rlat</code> .
<b>Address and Command Interface</b>			
ctl_addr	input	$MEM\_IF\_ROWADDR\_WIDTH \times DWIDTH\_RATIO / 2$	Row address from the controller.
ctl_ba	input	$MEM\_IF\_BANKADDR\_WIDTH \times DWIDTH\_RATIO / 2$	Bank address from the controller.
ctl_cke	input	$MEM\_IF\_CS\_WIDTH \times DWIDTH\_RATIO / 2$	Clock enable from the controller.
ctl_cs_n	input	$MEM\_IF\_CS\_WIDTH \times DWIDTH\_RATIO / 2$	Chip select from the controller.
ctl_odt	input	$MEM\_IF\_CS\_WIDTH \times DWIDTH\_RATIO / 2$	On-die-termination control from the controller.
ctl_ras_n	input	$DWIDTH\_RATIO / 2$	Row address strobe signal from the controller.
ctl_we_n	input	$DWIDTH\_RATIO / 2$	Write enable.
ctl_cas_n	input	$DWIDTH\_RATIO / 2$	Column address strobe signal from the controller.
ctl_rst_n	input	$DWIDTH\_RATIO / 2$	Reset from the controller.
<b>Calibration Control and Status Interface</b>			
ctl_mem_clk_disable	input	<code>MEM_IF_CLK_PAIR_COUNT</code>	When asserted, <code>mem_clk</code> and <code>mem_clk_n</code> are disabled. Unsupported for Cyclone III devices.

**Table 3-2.** AFI Signals (Part 4 of 4)

Signal Name	Type	Width (1)	Description
ctl_cal_success	output	1	A 1 indicates that calibration was successful.
ctl_cal_fail	output	1	A 1 indicates that calibration has failed.
ctl_cal_req	input	1	When asserted, a new calibration sequence is started. Currently not supported.
ctl_cal_byte_lane_sel_n	input	MEM_IF_DQS_WIDTH × MEM_CS_WIDTH	Indicates which DQS groups should be calibrated. Not supported.

**Note to Table 3-1:**

- (1) Refer to Table 3-9 for parameter descriptions.
- (2) See Figure 4-3 on page 4-14 for the reset mechanism for this signal.

**Table 3-3.** Other Interface Signals

Signal Name	Type	Width	Description
<b>External DLL Signals</b>			
dqs_delay_ctrl_export	output	DQS_DELAY_CTL_WIDTH	Allows sharing DLL in this ALTMEMPHY instance with another ALTMEMPHY instance. Connect the dqs_delay_ctrl_export port on the ALTMEMPHY instance with a DLL to the dqs_delay_ctrl_import port on the other ALTMEMPHY instance.
dqs_delay_ctrl_import	input	DQS_DELAY_CTL_WIDTH	Allows the use of DLL in another ALTMEMPHY instance in this ALTMEMPHY instance. Connect the dqs_delay_ctrl_export port on the ALTMEMPHY instance with a DLL to the dqs_delay_ctrl_import port on the other ALTMEMPHY instance.
dqs_offset_delay_ctrl	input	DQS_DELAY_CTL_WIDTH	Connects to the DQS delay logic when dll_import_export is set to IMPORT. Only connect if you are using a DLL offset, which can otherwise be tied to zero. If you are using a DLL offset, connect this input to the offset_ctrl_out output of the dll_offset_ctrl block.
dll_reference_clk	output	1	Reference clock to feed to an externally instantiated DLL. This clock is typically from one of the PHY PLL outputs.
<b>User-Mode Calibration OCT Control Signals</b>			
oct_ctl_rs_value	input	14	OCT RS value port for use with ALT_OCT megafunction if you want to use OCT with user-mode calibration.
oct_ctl_rt_value	input	14	OCT RT value port for use with ALT_OCT megafunction if you want to use OCT with user-mode calibration.
<b>Debug Interface Signals</b> (Note 1), (Note 2)			
dbg_clk	input	1	Debug interface clock.
dbg_reset_n	input	1	Debug interface reset.
dbg_addr	input	DBG_A_WIDTH	Address input.
dbg_wr	input	1	Write request.
dbg_rd	input	1	Read request.
dbg_cs	input	1	Chip select.

**Table 3-3.** Other Interface Signals

Signal Name	Type	Width	Description
dbg_wr_data	input	32	Debug interface write data.
dbg_rd_data	output	32	Debug interface read data.
dbg_waitrequest	output	1	Wait signal.
<b>PLL Reconfiguration Signals—Stratix III and Stratix IV Devices</b>			
pll_reconfig_enable	Input	1	This signal enables the PLL reconfiguration I/O, and is used if the user requires some custom PLL phase reconfiguration. It should otherwise be tied low.
pll_phasecountersselect	Input	4	When <code>pll_reconfig_enable</code> is asserted, this input is directly connected to the PLL's <code>phasecountersselect</code> input. Otherwise this input has no effect.
pll_phaseupdown	Input	1	When <code>pll_reconfig_enable</code> is asserted, this input is directly connected to the PLL's <code>phaseupdown</code> input. Otherwise this input has no effect.
pll_phasestep	Input	1	When <code>pll_reconfig_enable</code> is asserted, this input is directly connected to the PLL's <code>phasestep</code> input. Otherwise this input has no effect.
pll_phase_done	Output	1	Directly connected to the PLL's <code>phase_done</code> output.
<b>PLL Reconfiguration Signals—Stratix II Devices</b>			
pll_reconfig_enable	Input	1	Allows access to the PLL reconfiguration block. This signal should be held low in normal operation. While the PHY is held in reset (with <code>soft_reset_n</code> ), and <code>reset_request_n</code> is 1, it is safe to reconfigure the PLL. To reconfigure the PLL, set this signal to 1 and use the other <code>pll_reconfig</code> signals to access the PLL. When finished reconfiguring set this signal to 0, and then set the <code>soft_reset_n</code> signal to 1 to bring the PHY out of reset. For this signal to work, the <code>PLL_RECONFIG_PORTS_EN</code> GUI parameter must be set to <code>TRUE</code> .
pll_reconfig_write_param	Input	9	Refer to the <i>ALTPLL_RECONFIG User Guide</i> , for more information.
pll_reconfig_read_param	Input	9	Refer to the <i>ALTPLL_RECONFIG User Guide</i> , for more information.
pll_reconfig	Input	1	Refer to the <i>ALTPLL_RECONFIG User Guide</i> , for more information.
pll_reconfig_counter_type	Input	4	Refer to the <i>ALTPLL_RECONFIG User Guide</i> , for more information.
pll_reconfig_counter_param	Input	3	Refer to the <i>ALTPLL_RECONFIG User Guide</i> , for more information.
pll_reconfig_data_in	Input	9	Refer to the <i>ALTPLL_RECONFIG User Guide</i> for more information.
pll_reconfig_busy	Output	1	Refer to the <i>ALTPLL_RECONFIG User Guide</i> , for more information.
pll_reconfig_data_out	Output	9	Refer to the <i>ALTPLL_RECONFIG User Guide</i> , for more information.
pll_reconfig_clk	Output	1	Synchronous clock to use for any logic accessing the <code>pll_reconfig</code> interface. The same as <code>aux_scan_clk</code> .

**Table 3-3.** Other Interface Signals

Signal Name	Type	Width	Description
pll_reconfig_reset	Output	1	Resynchronised reset to use for any logic accessing the pll_reconfig interface.

**Notes to Table 3-3:**

- (1) The debug interface uses the simple Avalon-MM interface protocol.
- (2) These ports exist in the Quartus II software, even though the debug interface is for Altera's use only.

**Table 3-4.** I/O Interface to QDRII+/QDRII SRAM (Note 1)

Signal Name	Type	Width	Description
mem_addr	output	MEM_IF_ROWADDR_WIDTH	Memory address bus.
mem_clk	output	MEM_IF_CLK_PAIR_COUNT	Memory clock, positive edge clock (K).
mem_clk_n	output	MEM_IF_CLK_PAIR_COUNT	Memory clock, positive edge clock (K#) 180° offset from mem_clk.
mem_d	output	MEM_IF_DWIDTH	Memory data bus. D input to QDRII SRAM device.
mem_dm	output	MEM_IF_DM_WIDTH	Memory write select. BWS# input to QDRII SRAM device.
mem_dq	input	MEM_IF_DWIDTH	Memory data bus. Q output from QDRII SRAM device.
mem_dqs	input	MEM_IF_DWIDTH / MEM_IF_DQ_PER_DQS	Memory read clock high bits (CQ).
mem_dqsn	input	MEM_IF_DWIDTH / MEM_IF_DQ_PER_DQS	Memory read clock low bits (CQn).
mem_doff_n	output	1	Memory DLL disable control.
mem_rps_n	output	MEM_IF_CS_WIDTH	Memory read enable signal.
mem_wps_n	output	MEM_IF_CS_WIDTH	Memory write enable signal.

**Note to Table 3-4:**

- (1) Connected to WYSIWYGS/pad atoms.

**Table 3-5.** Clock and Reset Signals for QDRII+/QDRII SRAM (Part 1 of 2)

Signal Name	Type	Width	Description
global_reset_n (1)	input	1	The asynchronous reset input to the controller. All other reset signals are derived from resynchronized versions of this. This signal holds the complete ALTMEMPHY megafunction, including the PLL, in reset while low.
soft_reset_n (1)	input	1	The asynchronous reset input to reset controller, for SOPC Builder use, or to be controlled by other system reset logic. This signal causes a complete reset of the PHY, but not the PLL in the PHY. In Arria GX, Arria II GX, Stratix II, and Stratix II GX devices, this signal also resets the PLL reconfiguration block on a falling-edge detection.

**Table 3-5.** Clock and Reset Signals for QDRII+/QDRII SRAM (Part 2 of 2)

Signal Name	Type	Width	Description
phy_clk	output	1	The ALTMEMPHY megafunction half-rate clock provided to the user. All user inputs and outputs to the ALTMEMPHY megafunction are synchronous to this clock in half-rate designs. However, this clock is not used in full-rate designs.
pll_ref_clk	input	1	The reference clock input to PLL.
reset_phy_clk_n (1)	output	1	Asynchronous reset, that is de-asserted synchronously with respect to the associated phy_clock clock domain. Use this to reset any additional user logic on that clock domain.
reset_request_n (1)	output	1	Directly connected to the locked output of the PLL and is intended for optional use either by automated tools such as SOPC Builder or could be manually ANDed with any other system-level signals and combined with any edge detect logic as required and then fed back to the global_reset_n input.  Reset request output that indicates when the PLL outputs are not locked. Use this as a reset request input to any system-level reset controller you may have. This signal is always low while the PLL is locking (but not locked), and so any reset logic using it is advised to detect a reset request on a falling-edge rather than by level detection.
aux_half_rate_clk	output	1	A copy of the phy_clk_1x signal that you can use in other parts of your design, same as phy_clk port.
aux_full_rate_clk	output	1	A copy of the mem_clk_2x signal that you can use in other parts of your design.

**Note to Table 3-5:**

- (1) Refer to Figure 4-3 on page 4-14 for the reset mechanism in Stratix III and Stratix IV devices or to Figure 5-3 for the reset mechanism in Arria GX, Arria II GX, Cyclone III, Stratix II and Stratix II GX devices.

The ports listed in Table 3-6 only exist when you target Stratix III and Stratix IV devices. You can leave them unconnected if you are not using user-mode calibrated OCT.

**Table 3-6.** User-Mode Calibrated OCT Control Signals for QDRII+/QDRII SRAM (Note 1), (2)

Signal Name	Type	Width	Description
oct_ctl_rs_value	input	14	Specifies serial termination value. Connects to the serieterminationcontrol bus of the ALT_OCT megafunction. This port exists when you target Stratix IV and Stratix III devices only.
oct_ctl_rt_value	input	14	Specifies parallel termination value. Connects to the parallelterminationcontrol bus of the ALT_OCT megafunction. This port exists when you target Stratix IV and Stratix III devices only.

**Notes to Table 3-6:**

- (1) These ports are available if you want to use user-mode OCT calibration. Otherwise, they can be left unconnected.  
(2) For more information on OCT, see the ALT\_OCT Megafunction User Guide.

**Table 3-7.** Datapath Interface for QDRII+/QDRII SRAM (Note 1)

Signal Name	Type	Width	Description
ctl_mem_addr_h	input	MEM_IF_ ROWADDR_WIDTH	Write address from the controller to the external memory.
ctl_mem_addr_l	input	MEM_IF_ ROWADDR_WIDTH	Read address from the controller to the external memory.
ctl_mem_be	input	LOCAL_IF_ DWIDTH × DWIDTH_ RATIO	Optional byte enable signals for the write data to the external memory. The PHY converts the byte enables into memory DM signals. If DM pins are not required (MEM_DM_PINS set to <b>FALSE</b> ), the DM logic is not generated and the DM pins are not instantiated.
ctl_mem_wdata	input	MEM_IF_ DWIDTH * DWIDTH_ RATIO	The write data bus, which has valid data in the same clock cycles that control_wdata_valid is asserted.
ctl_mem_rdata_valid	output	1	Indicates when the ctl_mem_rdata is valid.
ctl_mem_wps_n	input	MEM_IF_CS_ WIDTH	Write enable signal from the controller to the memory (QDRII SRAM). When this signal is asserted, a write request is issued to the address presented on the ctl_mem_addr_h port.
ctl_mem_rps_n	input	MEM_IF_CS_ WIDTH	Read enable signal from the controller to the memory (QDRII SRAM). When this signal is asserted, a read request is issued to the address presented on the ctl_mem_addr_l port.
ctl_mem_rdata	output	LOCAL_IF_ DWIDTH	Captured, resynchronized, and de-multiplexed read data from the PHY to the controller.
ctl_mem_wdata_valid	input	1	Generates the DQ output enable.

**Note to Table 3-7:**

(1) Address and command and wdata/rdata.

**Table 3-8.** Calibration Status Signals for QDRII+/QDRII SRAM (Note 1)

Signal Name	Type	Width	Description
ctl_usr_mode_rdy	output	1	Active high signal specifying the PHY has finished its calibration and is ready to accept user read or write requests. This signal does not indicate that calibration was successful, so you must check whether resynchronization_successful is high also.
resynchronisation_successful	output	1	Active high signal that shall be set to indicate that calibration of the read data resynchronization clock phase was successful.
ctl_rlat	output	5	Indicates the read latency of the interface in PHY clock cycles.

**Note to Table 3-8:**

(1) Calibration control or passed through from the user interface.

Table 3-9 shows the parameters that Table 3-1 through Table 3-8 refer to.



**Table 3-9.** Parameters

Parameter Name	Description
DBG_A_WIDTH	—
DQS_DELAY_CTL_WIDTH	—
DWIDTH_RATIO	The data width ratio from the local interface to the memory interface. DWIDTH_RATIO of 2 means full rate, while DWIDTH_RATIO of 4 means half rate.
LOCAL_IF_DWIDTH	The width of the local data bus must be quadrupled for half-rate and doubled for full-rate.
MEM_IF_DWIDTH	The data width at the memory interface. MEM_IF_DWIDTH can have values that are multiples of MEM_IF_DQ_PER_DQS.
MEM_IF_DQS_WIDTH	The number of DQS pins in the interface.
MEM_IF_ROWADDR_WIDTH	The row address width of the memory device.
MEM_IF_BANKADDR_WIDTH	The bank address with the memory device (not used in QDRII+/QDRII SRAM variations).
MEM_IF_CS_WIDTH	The number of chip select pins in the interface. The sequencer only calibrates one chip select pin.
MEM_IF_DM_WIDTH	The number of mem_dm pins on the memory interface.
MEM_IF_DQ_PER_DQS	The number of mem_dq[ ] pins per mem_dqs pin.
MEM_IF_CLK_PAIR_COUNT	The number of mem_clk/mem_clk_n pairs in the interface.
READ_LAT_WIDTH	The bus width for the ctl_rlat signal, used in QDRII+/QDRII SRAM PHY, that determines the read latency of your system.



### Introduction

This chapter describes the ALTMEMPHY megafunction support for Stratix IV, Stratix III, HardCopy IV, and HardCopy III devices.



In this chapter, any mention of Stratix IV and Stratix III devices also includes HardCopy IV and HardCopy III devices.

These devices support the broadest variations of the ALTMEMPHY megafunction including those for half-rate DDR3/DDR2/DDR SDRAM and QDRII+/QDRII SRAM. In addition, the ALTMEMPHY variations supports full-rate DDR2/DDR SDRAM interfaces.



If you are not targeting Stratix IV or Stratix III, refer to [“Support for Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Devices”](#) on page 5–1 or [“ALTMEMPHY Support for Cyclone III Devices”](#) on page 6–1.

### DDR2/DDR SDRAM

Stratix IV and Stratix III devices support half-rate or full-rate DDR2/DDR SDRAM.

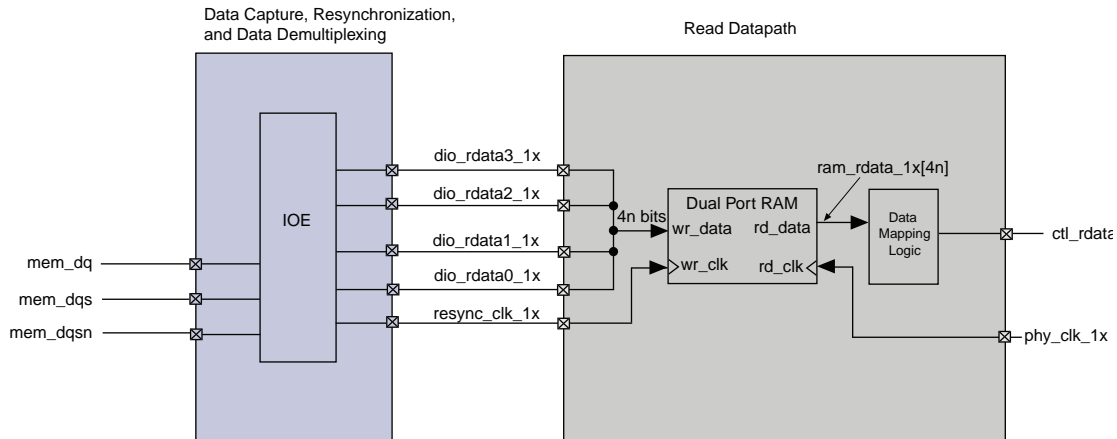
### Half-Rate Support

The following section discusses half-rate support for DDR2/DDR SDRAM in Stratix IV and Stratix III devices.

#### Read Datapath

The Stratix IV and Stratix III read datapath ([Figure 4–1](#)) consists of two main blocks:

- Data capture, resynchronization, and demultiplexing
- Read datapath logic (read datapath)

**Figure 4-1.** DDR2/DDR SDRAM Data Capture and Read Data Mapping in Stratix IV and Stratix III Devices**Note to Figure 4-1:**

(1) This figure shows a half-rate variation. For a full-rate controller, `dio_radata2_1x` and `dio_radata3_1x` are unconnected.

**Data Capture, Resynchronization, and Demultiplexing**

In Stratix IV and Stratix III devices, the smart interface module in the IOE performs the following tasks:

- Captures the data
- Resynchronizes the captured data from the DQS domain to the resynchronization clock (`resync_clk_1x`) domain
- Converts the resynchronized data into half-rate data, which is performed by feeding the resynchronized data into the HDR conversion block within the IOE, which is clocked by the half-rate version of the resynchronization clock. The `resync_clk_1x` signal is generated from the I/O clock divider module based on the `resync_clk_2x` signal from the PLL.



For more information about IOE registers, refer to the *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* and the *External Memory Interfaces in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

**Data Resynchronization**

The read datapath block performs the following two tasks:

1. Transfers the captured read data (`rdata[n]_1x`) from the half-rate resynchronization clock (`resync_clk_1x`) domain to the half-rate system clock (`phy_clk_1x`) domain using DPRAM. Resynchronized data from the FIFO is shown as `ram_data_1x`.
2. Reorders the resynchronized data (`ram_rdata_1x`) into `ctl_mem_rdata`.

### Postamble Protection

A dedicated postamble register controls the gating of the shifted DQS signal that clocks the DQ input registers at the end of a read operation. This ensures that any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches. The postamble path is also calibrated to determine the correct clock cycle, clock phase shift, and delay chain settings. You can see the process in simulation if you choose Full calibration (long simulation time) mode in the MegaWizard Plug-In.



For more information about the postamble protection circuitry, refer to the *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* and the *External Memory Interfaces in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

### Clock and Reset Management

The clocking and reset block is responsible for clock generation, reset management, and phase shifting of clocks. It also has control of clock network types that route the clocks.

The ability of the ALTMEMPHY megafunction to work out the optimum phase during calibration and to track voltage and temperature variation relies on phase shifting the clocks relative to each other.



Certain clocks need to be phase shifted during the ALTMEMPHY megafunction operation.

Clock management circuitry is implemented by using:

- PLL
- DLL

#### PLL

The ALTMEMPHY MegaWizard Plug-In automatically generates an ALTPLL megafunction instance. The ALTPLL megafunction is responsible for generating the different clock frequencies and relevant phases used within the ALTMEMPHY megafunction.

The device families available have different PLL capabilities. The minimum PHY requirement is to have 16 phases of the highest frequency clock. The PLL uses **With No Compensation** operation mode to minimize jitter. Changing the PLL compensation mode may result in inaccurate timing results.

You must choose a PLL and PLL input clock pin that are located on the same side of the device as the memory interface to ensure minimal jitter. Cascaded PLLs are not recommended as jitter can accumulate, causing the memory output clock to violate the memory device jitter specification. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset (by driving the `gloabl_reset_n` signal low) and relock the PLL to ensure that the phase relationship between all PLL outputs are properly set.

For more information about the VCO frequency range and the available phase shifts, refer to the *Clock Networks and PLLs in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* or the *Clock Networks and PLLs in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.

For Stratix IV and Stratix III devices, the PLL reconfiguration is done using the phase-shift inputs on the PLL instead of using the PLL reconfiguration megafunction.

Table 4-1 shows the Stratix IV and Stratix III PLL clock outputs.

**Table 4-1.** DDR2 SDRAM Clocking in Stratix IV and Stratix III Devices (Part 1 of 3)

Design Rate	Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate	phy_clk_1x and aux_half_rate_clk	C0	30	Half-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. It is set to <b>30°</b> to ensure proper half-rate to full-rate transfer for write data and DQS. This clock also feeds into a divider circuit to provide the PLL scan_clk signal for reconfiguration.
	aux_full_rate_clk	C2	60	Full-Rate	Global	The aux_clk. The 60°-offset maintains edge alignment with the offset on phy_clk_1x.
Full-rate	aux_half_rate_clk	C0	0	Half-Rate	Global	The aux_clk.
	phy_clk_1x and aux_full_rate_clk	C2	0	Full-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. This clock also feeds into a divider circuit to provide the PLL scan_clk signal for reconfiguration.

**Table 4-1.** DDR2 SDRAM Clocking in Stratix IV and Stratix III Devices (Part 2 of 3)

Design Rate	Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate and full-rate	mem_clk_2x	C1	0	Full-Rate	Special	Generates mem_clk provides the reference clock for the DLL. A dedicated routing resource exists from the PLL to the DLL, which you select with the regional routing resource for the mem_clk using the following attribute in the HDL: <pre>(-name global_signal dual_regional _clock; -to dll~DFFIN -name global_signal off).</pre> If you use an external DLL, apply this attribute similarly to the external DLL.
Half-rate and full-rate	write_clk_2x	C3	-90	Full-Rate	Dual regional	This clock is for clocking the data out of the double data rate input/output (DDIO) pins in advance of the DQS strobe (or equivalent). As a result, its phase leads that of the mem_clk_2x clock by 90°.
Half-rate and full-rate	resync_clk_2x	C4	Calibrated	Full-Rate	Dual regional	This clock feeds the I/O clock divider that then clocks the resynchronization registers after the capture registers. Its phase is adjusted in the calibration process. You can use an inverted version of this clock for postamble clocking.

**Table 4-1.** DDR2 SDRAM Clocking in Stratix IV and Stratix III Devices (Part 3 of 3)


Design Rate	Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate and full-rate	measure_clk_1x (2)	C5	Calibrated	Half-Rate	Dual regional	This clock is for VT tracking. This free-running clock measures relative phase shifts between the internal clock(s) and those being fed back through a mimic path. As a result, the ALTMEMPHY megafunction can track VT effects on the FPGA and compensate for the effects.
Half-rate and full-rate	ac_clk_1x	C6	Set in the GUI	Half-Rate	Dual regional	Address and command clock.

**Notes to Table 4-1:**

- (1) In full-rate designs a \_1x clock may run at full-rate clock rate.  
(2) This clock should be of the same clock network clock as the resync\_clk\_2x clock.

**DLL**

DLL settings are set depending on the memory clock frequency of operation.

 For more information on the DLL, refer to the *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* and the *External Memory Interfaces in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.


**Reset Management**


The reset management block for DDR2/DDR SDRAM in Stratix IV and Stratix III devices is similar to the reset management block for DDR3 SDRAM. For more information, refer to [Figure 4-3 on page 4-14](#).

**Write Datapath**

The memory controller interface outputs 4 *n*-bit wide data (ctl\_wdata) at phy\_clk\_1x frequency. The write data is clocked by the system clock phy\_clk\_1x at half data rate and reordered into HDR of width 4 *n*-bits represented in [Figure 4-4](#) by wdp\_wdata3\_1x, wdp\_wdata2\_1x, wdp\_wdata1\_1x, and wdp\_wdata0\_1x.

All of the write datapath registers in the Stratix IV and Stratix III devices are clocked by the half-rate clock, phy\_clk\_1x.

 For full-rate controllers, phy\_clk\_1x runs at full rate and there are only two bits of wdata.

 For more information about the Stratix III I/O structure, refer to the *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* and the *External Memory Interfaces in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.



## Address and Command Datapath

The address and command clock is one of the PLL dedicated clock outputs whose phase can be adjusted to meet the setup and hold requirements of the memory clock. The Stratix III address/command clock, `ac_clk_1x`, is half-rate. The command/address pins use the DDIO output circuitry to launch commands from either the rising or falling edges of the clock (set in the GUI). The chip select (`cs_n`) pins and ODT are only enabled for one memory clock cycle and can be launched from either the rising or falling edge of `ac_clk_1x` signal, while 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 `ac_clk_1x` signal.

## Full-Rate Support

The following section discusses full-rate support.

### Read Datapath

The full-rate datapath is similar to the half-rate datapath, except that the resynchronization FIFO converts from the full-rate resynchronization clock domain (`resync_clk_2x`) to the full-rate PHY clock domain, instead of converting it to the half-rate PHY clock domain as in half-rate designs.

### Postamble Protection

Full-rate postamble protection is the same as the half-rate support. For more information, refer to [“Postamble Protection” on page 4-3](#).

### Clock and Reset Management

Clock and reset management is similar to half-rate support (see [Table 4-1 on page 4-4](#)). The PLL is configured exactly in the same way as for half-rate support. The `mem_clk_2x` output acts as the PHY full-rate clock. Also, instead of going through the I/O clock divider, the `resync_clk_2x` output is now directly connected to the resynchronization registers. The rest of the PLL outputs are connected in the same way as for half-rate support.

You must choose a PLL and PLL input clock pin that are located on the same side of the device as the memory interface to ensure minimal jitter. Cascaded PLLs are not recommended as jitter can accumulate, causing the memory output clock to violate the memory device jitter specification. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset (by driving the `global_reset_n` signal low) and relock the PLL to ensure that the phase relationship between all PLL outputs are properly set. The PLL restrictions in half-rate designs also applies to full-rate designs.

### Write Datapath

The write datapath is similar to the half-rate PHY. The IOE block is identical to the half-rate PHY. The latency of the write datapath in the full-rate PHY is less than in the half-rate PHY because the full-rate PHY does not have half-rate to full-rate conversion logic.

## Address and Command Datapath

The address and command datapath is the same as that of the half-rate address and command datapath, except that there is no full-rate to half-rate conversion in the IOE. The address and command signals are full-rate here.

## DDR3 SDRAM

The ALTMEMPHY megafunction supports DDR3 SDRAM DIMMs with leveling and DDR3 SDRAM components without leveling:

- ALTMEMPHY with leveling is for unbuffered DIMMs (including SODIMM and MicroDIMM) or DDR3 SDRAM components up to 80-bit total data bus width with a layout like a DIMM:
  - Supports a fully-calibrated DDR3 SDRAM PHY for DDR3 SDRAM single-rank unbuffered DIMM with ×4 and ×8 devices with 300-MHz to 533-MHz frequency targets.
  - Deskew circuitry is enabled automatically for interfaces higher than 400.000 MHz
  - Supports single chip select only—you cannot create dual-rank or multiple DIMM DDR3 SDRAM interfaces
  - Support for RDIMM and MiniDIMM formats is still under evaluation
- ALTMEMPHY supports DDR3 SDRAM components without leveling for Stratix III and Stratix IV devices using T-topology for clock, address, and command bus:
  - Supports multiple chip selects
- The DDR3 SDRAM PHY with leveling  $f_{MAX}$  is 533 MHz; without leveling  $f_{MAX}$  is 400 MHz.
- No support for DM pins for ×4 DDR3 SDRAM DIMMs or components, so select **No** for **Drive DM pins from FPGA** when using ×4 devices
- The ALTMEMPHY megafunction supports half-rate DDR3 SDRAM interfaces only.

To select leveling or without leveling, change the **Memory Format** option in the preset editor. Select **Discrete Device** for a component without leveling; select **Unbuffered DIMM** for a DIMM with leveling functionality. **Registered DIMM** is not supported.

For DDR3 SDRAM components on the board in a DIMM configuration, you must select **Unbuffered DIMM**.



For Altera-recommended layout guidelines for DDR3 SDRAM memory interfaces using devices, refer to *AN 520: DDR3 Memory Interface Termination and Layout Guidelines*.

## Read Datapath

The DDR3 SDRAM controller asserts `ctl_doing_rd` to indicate that a read command is requested. The `ctl_doing_rd` signal is then used for the following purposes:

- Control of the postamble circuit
- Generation of `ctl_rdata_valid` from one bit to two bits
- Dynamic OCT control timing

The DDR3 SDRAM ALTMEMPHY then asserts the `ctl_rdata_valid` signal to indicate that the data on the read data bus is valid. The `ctl_rdata_valid` signal is two-bits wide to allow controllers to issue reads and writes that are aligned to either the half-cycle of the half-rate clock.

When calibration is over, the read latency of the PHY (the `ctl_rlat` signal) is sent back to the controller to indicate how long it takes in `ctl_clk` clock cycles from assertion of the `ctl_doing_read` signal to the valid read data returning on the `ctl_rdata` bus. The `ctl_rlat` signal is only valid when calibration has successfully completed and never changes values during normal user mode operation.

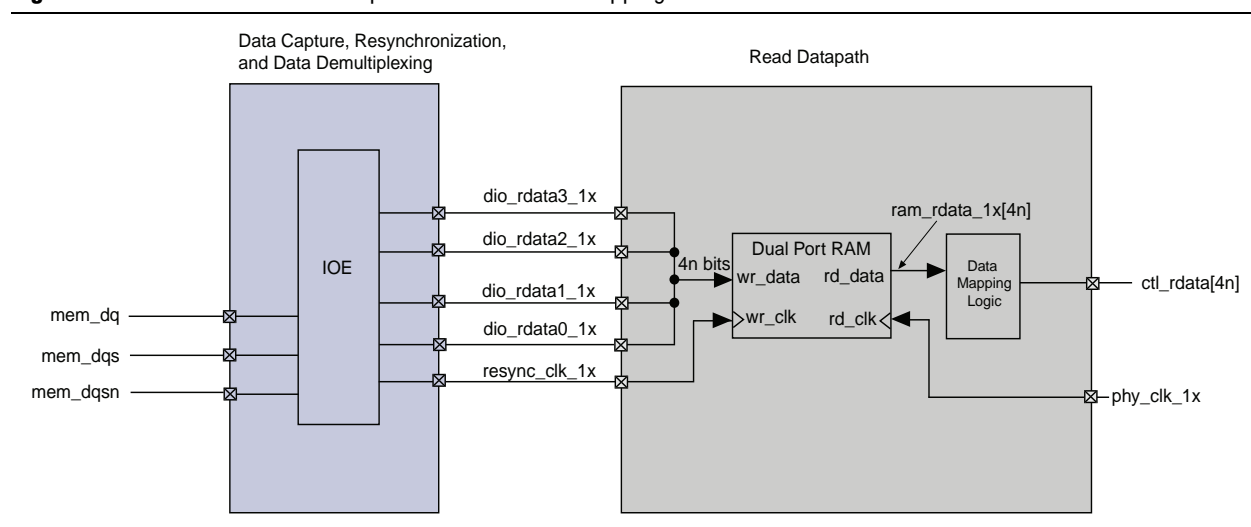
The read datapath for DDR3 SDRAM consists of two main blocks:

- Read data capture, resynchronization, and demultiplexing (in the `dp_io_siii` module)
- Read data alignment logic (in the `read_dp` module) to transfer data from the `resync_clk_2x` (half-rate resynchronization) clock domain to the `phy_clk` clock domain.

As the DQS/DQSn signal is not continuous, the PHY also has postamble protection logic to ensure that any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches.

Figure 4-2 shows the order of the functions performed by the read datapath and the frequency at which the read data is handled.

**Figure 4-2.** DDR3 SDRAM Data Capture and Read Data Mapping in Stratix IV and Stratix III Devices



### Data Capture, Resynchronization, and Demultiplexing

The IOE in Stratix III and Stratix IV devices performs the following tasks during read operation:

- Captures the data

- Resynchronizes the captured data from the DQS domain to the resynchronization clock (`resync_clk_1x`) domain
- Converts the resynchronized data into HDR data

This operation is performed by feeding the resynchronized data into the HDR conversion block within the IOE, which is clocked by the half-rate resynchronization clock (`resync_clk_1x`). The `resync_clk_1x` signal is generated from the I/O clock divider module, based on the `resync_clk_2x` signal from the PLL.

### Read Data Storage Logic

The read block performs the following two tasks:

- Transfers the captured read data (`rdata[n]_1x`) from the half-rate resynchronization clock (`resync_clk_1x`) domain to the half-rate system clock (`phy_clk_1x`) domain using DPRAM. Resynchronized data from the DPRAM is shown as `ram_data_1x`.
- Reorders the resynchronized data (`ram_rdata_1x`) into `ctl_mem_rdata`, to be presented in the user clock domain in the same clock cycle.

### Postamble Protection

A dedicated postamble register controls the gating of the shifted DQS signal that clocks the DQ input registers at the end of a read operation. This ensures that any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches.

The postamble path is also calibrated to determine the correct clock cycle, clock phase shift, and delay chain settings. You can see the process in simulation if you choose **Full calibration** (long simulation time) mode in the MegaWizard Plug-In.

## Clock and Reset Management

The clocking and reset block is responsible for clock generation, reset management, and phase shifting of clocks. It also has control of clock network types that route the clocks, which is handled in the `<variation_name>_alt_mem_phy_clk_reset` module in the `<variation_name>_alt_mem_phy.v.vhd` file.

### Clock Management

The ability of the ALTMEMPHY megafunction to work out the optimum phase during calibration, and to track voltage and temperature variation relies on phase shifting the clocks relative to each other.



Certain clocks require phase shifting during the ALTMEMPHY megafunction operation.

Clock management circuitry is implemented by using:

- PLL
- DLL

## PLL

The ALTMEMPHY MegaWizard Plug-In automatically generates an ALTPLL megafunction instance. The ALTPLL megafunction is responsible for generating the different clock frequencies and relevant phases used within the ALTMEMPHY megafunction.

The available device families have different PLL capabilities. The minimum PHY requirement is to have 16 phases of the highest frequency clock. The PLL uses **With No Compensation** to minimize jitter. Changing the PLL compensation to a different operation mode may result in inaccurate timing results.

The input clock to the PLL does not have any other fan-out to the PHY, so you do not have to use a global clock resource for the path between the clock input pin to the PLL. You must use the PLL located in the same device quadrant or side as the memory interface and the corresponding clock input pin for that PLL, to ensure optimal performance and accurate timing results from the Quartus II software.

You must choose a PLL and PLL input clock pin that are located on the same side of the device as the memory interface to ensure minimal jitter. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset (by driving the `global_reset_n` signal low) and relock the PLL to ensure that the phase relationship between all PLL outputs are properly set.



If the design cascades PLLs, the source (upstream) PLL should have a low-bandwidth setting; the destination (downstream) PLL should have a high-bandwidth setting. Adjacent PLLs cascading is recommended to reduce clock jitter. Cross-device cascading PLLs are only allowed in Stratix III devices with the following conditions:

- Upstream PLL:  $0.59 \text{ MHz} \leq \text{upstream PLL bandwidth} < 1 \text{ MHz}$ . The upstream PLL should use the **With No Compensation** operation mode.
- Downstream PLL:  $\text{downstream PLL bandwidth} > 2 \text{ MHz}$ .



For more information about the VCO frequency range and the available phase shifts, refer to the *Clock Networks and PLLs* chapter in the respective device family handbook.

Table 4-2 shows the PLL outputs and their usage for Stratix III and Stratix IV devices.

**Table 4-2.** DDR3 SDRAM Clocking Stratix IV and Stratix III Devices (Part 1 of 2)

<b>Clock Name (1)</b>	<b>Postscale Counter</b>	<b>Phase (Degrees)</b>	<b>Clock Rate</b>	<b>Clock Network Type</b>	<b>Notes</b>
phy_clk_1x and aux_half_rate_clk	C0	30° (with leveling) -40° (without leveling)	Half-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. With phy_clk_1x the sequencer generates another sc_clk_dp clock with this clock that programs the scan chains of the I/O elements. For more information on changing the clock network type, refer to “Instantiating Multiple ALTMEMPHY Instances” on page 2-31.
mem_clk_2x	C1	0	Full-Rate	Special	Generates mem_clk provides the reference clock for the DLL. A dedicated routing resource exists from the PLL to the DLL, which you select with the regional routing resource for the mem_clk using the following attribute in the HDL: (-name global_signal dual_regional_clock; -to dll~DFFIN -name global_signal off). If you use an external DLL, apply this attribute similarly to the external DLL.
aux_full_rate_clk	C2	0° (with leveling) 60° (without leveling)	Full-Rate	None	A copy of mem_clk_2x that you can use in other parts of your design.
write_clk_2x	C3	0° (with leveling) -90° (without leveling)	Full-Rate	Regional	This clock feeds the write leveling delay chains that generate the DQ, DM, DQS, and mem_clk signals.
resync_clk_2x	C4	Calibrated	Full-Rate	Regional	This clock feeds the I/O clock divider that then reads the data out of the DDIO pins. Its phase is adjusted in the calibration process. The design uses an inverted version of this clock for postamble clocking.
measure_clk_1x	C5	Calibrated	Half-Rate	Regional (2)	This clock is for VT tracking. This free-running clock measures relative phase shifts between the internal clock(s) and those being fed back through a mimic path. As a result, you can track VT effects on the FPGA and compensate for the effects.

**Table 4-2.** DDR3 SDRAM Clocking Stratix IV and Stratix III Devices (Part 2 of 2)

<b>Clock Name (1)</b>	<b>Postscale Counter</b>	<b>Phase (Degrees)</b>	<b>Clock Rate</b>	<b>Clock Network Type</b>	<b>Notes</b>
ac_clk_1x	C6	Set in the GUI	Half-Rate	Regional	Address and command clock.

**Notes to Table 4-2:**

- (1) In full-rate designs a `_1x` clock may run at full-rate clock rate.
- (2) This clock should be of the same clock network clock as the `resync_clk_1x` clock.

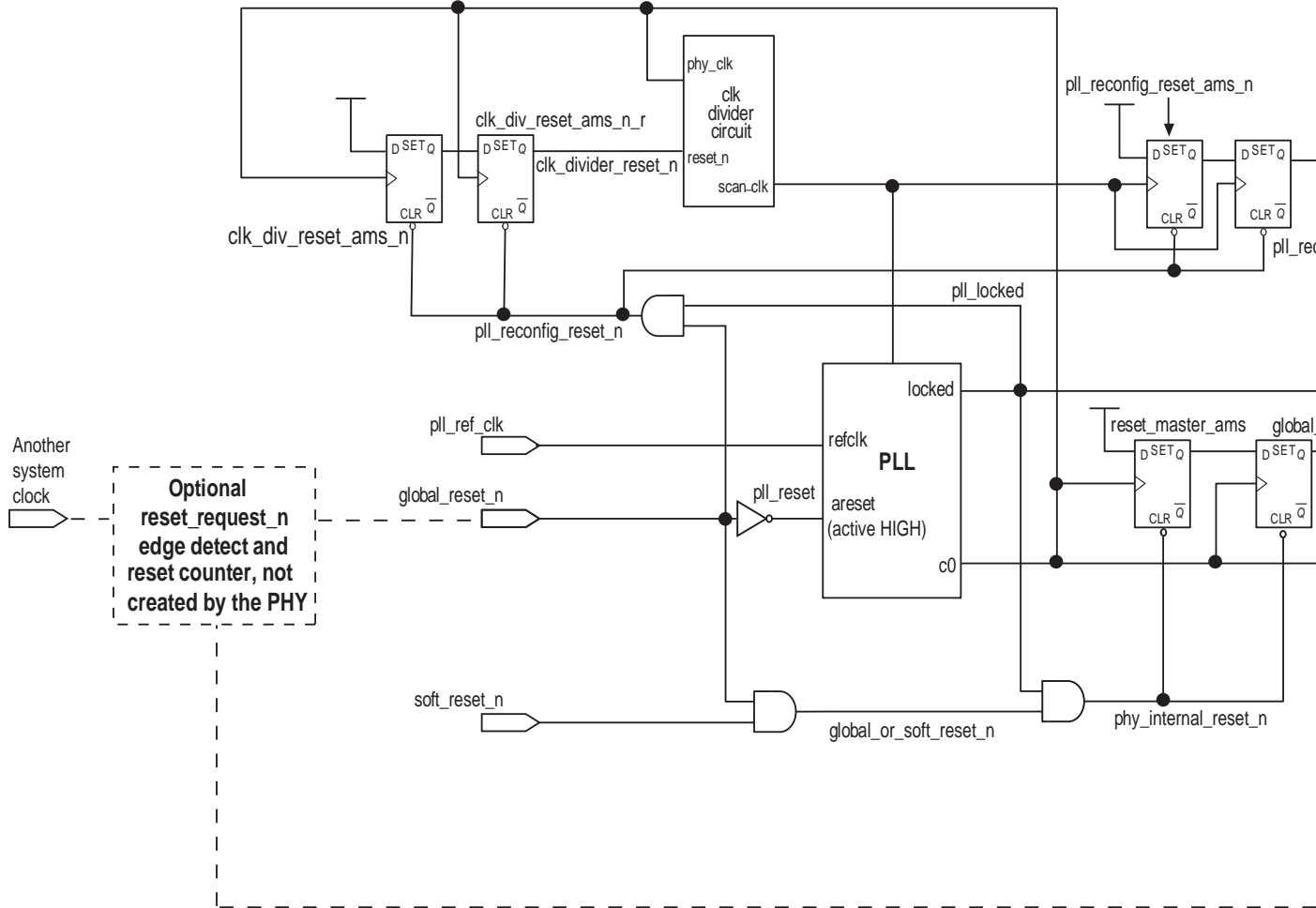
For Stratix III and Stratix IV devices, the phase-shift inputs on the PLL perform the PLL reconfiguration. The PLL reconfiguration megafunction is not required.

### Reset Management

Figure 4-3 shows the main features of the reset management block for the DDR3 SDRAM PHY. You can use the `pll_ref_clk` input to feed the optional `reset_request_n` edge detect and reset counter module. However, this requires the `pll_ref_clk` signal to use a global clock network resource.

There is a unique reset metastability protection circuit for the clock divider circuit because the `phy_clk` domain reset metastability protection registers have fan-in from the `soft_reset_n` input so these registers cannot be used.

**Figure 4-3.** ALTMEMPHY Reset Management Block for Stratix IV and Stratix III Devices



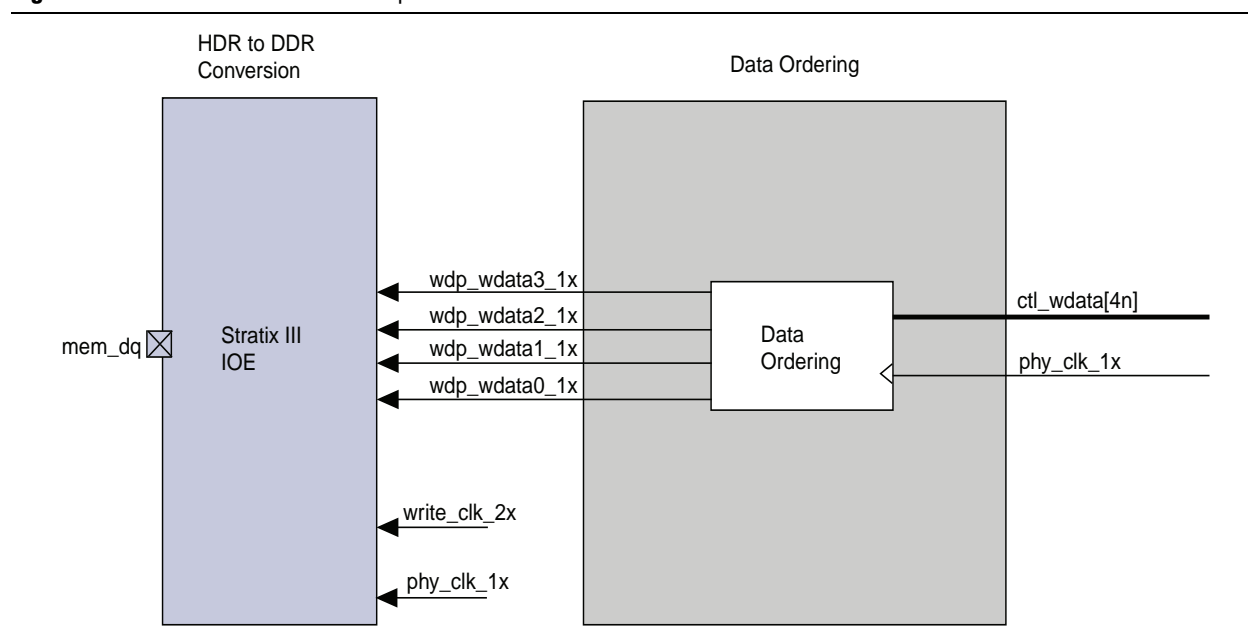


## Write Datapath

The memory controller interface outputs four  $n$ -bit wide data (`ctl_wdata`) at `phy_clk_1x` frequency. The write data is clocked by the system clock `phy_clk_1x` at half-data rate (HDR) and reordered into HDR of width four with  $n$ -bits each, represented in Figure 4-4 by `wdp_wdata3_1x`, `wdp_wdata2_1x`, `wdp_wdata1_1x`, and `wdp_wdata0_1x`.

Figure 4-4 shows the reordered or the reordered-and-delayed HDR data is then converted to DDR data within the IOE element using both the half-rate and full-rate clocks.

**Figure 4-4.** DDR3 SDRAM Write Datapath in Stratix IV and Stratix III Devices



The write datapath DDIO registers are clocked by the `phy_clk_1x` clock. The `write_clk_2x` signal then clocks the alignment registers.


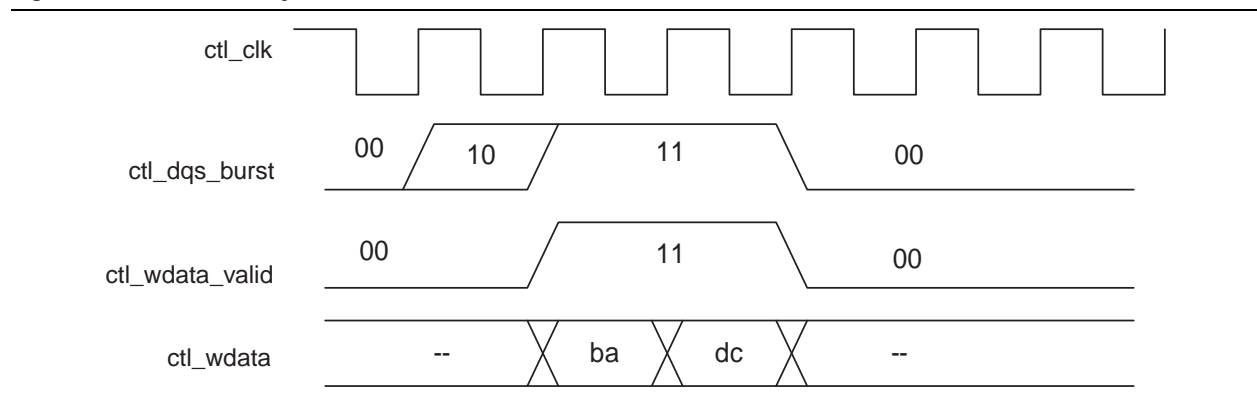
 For more information about the I/O structure, refer to the *External Memory Interface* chapter in the respective device family handbook.

Figure 4-5 shows how the write data, `ctl_wdata` signals should be aligned from the controller during a (half rate, normally aligned) write operation. The PHY then issues the write data as ABCD where a is the first data to be written to the memory. (ABCD represent two beats of data each.) The `ctl_wdata_valid` signal in Figure 4-5 shows the output enable for the DQ and DM pins.

**Figure 4-5.** Write Data Alignment from the DDR3 SDRAM Controller

## Address and Command Datapath

The address and command clock is one of the PLL dedicated clock outputs whose phase can be adjusted to meet the setup and hold requirements of the memory clock. The Stratix III address and command clock, *ac\_clk\_1x*, is half rate. 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 (*mem\_cs\_n*), clock enable (*mem\_cke*), and *mem\_odt* pins are enabled on one memory clock cycle basis and can be launched from either the rising or falling edge of the *ac\_clk\_1x* signal, while 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 *ac\_clk\_1x* signal. It is the responsibility of the controller to maintain the relative timing of the signals.


The DDR3 SDRAM PHY generates a write latency output *ctl\_wlat* that indicates the number of *ctl\_clk* cycles between the write command being issued, *ctl\_cs\_n* asserted, and *ctl\_dqs\_burst* being asserted. This *ctl\_wlat* signal is only valid when calibration has been successfully completed by the ALTMEMPHY sequencer and does not change at any point during normal user mode operation. The value on *ctl\_wlat* includes the effect of the following as determined during calibration:

- CAS write latency (CWL)
- Additive latency
- Datapath latencies and relative phases
- Board and memory module layout
- Address and command path latency and 1T register setting which is dynamically set up to take into account any leveling effects

## QDRII+/QDRII SRAM

The ALTMEMPHY megafunction currently supports half-rate QDRII+/QDRII SRAM interfaces with burst length of four only. In addition, Stratix IV and Stratix III FPGAs support QDRII SRAM devices with 1.5-cycle read latency and QDRII+ SRAM devices with 2.5-cycle read latency. Stratix III devices do not support QDRII+ SRAM devices with 2.0-cycle read latency.

The ALTMEMPHY sequencer for QDRII+/QDRII SRAM selects the correct clock phase for the clock to the HDR registers, as described in detail in the Calibration section in “Specifications” on page 3-1. As the QDRII+/QDRII SRAM read data clock is continuous, there is no postamble circuitry required in the ALTMEMPHY megafunction. Furthermore, because the data bus is unidirectional, the ALTMEMPHY megafunction for QDRII+/QDRII SRAM interfaces does not support dynamic OCT.

 In the QDRII+/QDRII SRAM ALTMEMPHY megafunction, the read data bus is called mem\_dq ports, similar to the DDR3, DDR2, and DDR SDRAM ALTMEMPHY megafunction. The write data bus in the QDRII+/QDRII SRAM ALTMEMPHY megafunction is called mem\_d ports.

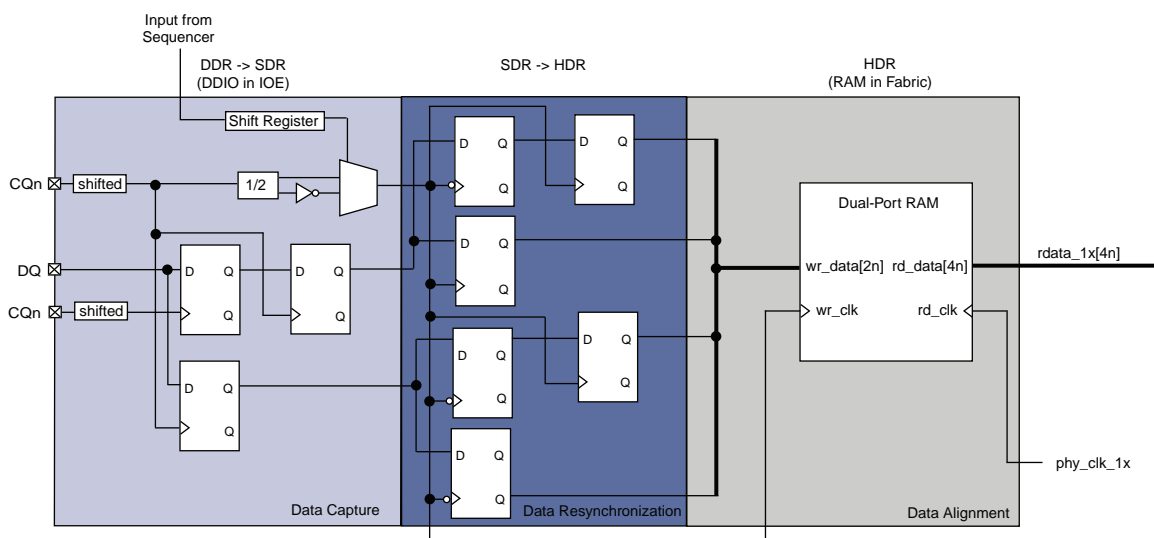
## Read Datapath

The read datapath in the QDRII+/QDRII SRAM ALTMEMPHY megafunction is responsible for capturing data sent by the memory device and subsequently aligning the data back to the system clock domain, through the following operations:

- Data capture, resynchronization, and demultiplexing in the IOE
- Data alignment in the device RAM block

Figure 4-6 shows the order of the functions performed by the read datapath, along with the frequency at which the read data is handled.

**Figure 4-6.** QDRII+/QDRII SRAM Read Datapath



### Data Capture, Resynchronization, and Demultiplexing

Data capture and resynchronization is the process of capturing the read data (DQ) with the rising edge of the shifted CQ and shifted CQn signals and transferring the data to the HDR registers in the IOE. The resynchronization clock is the delayed CQ signal that has been divided by 2, whose phase shift is determined during the calibration stage. This signal clocks the HDR registers where the demultiplexed data goes to a DPRAM.

## Data Alignment

Data alignment is the process controlled by the sequencer to ensure the correct captured read data is present in the same half-rate clock cycle at the output of the read data DPRAM.

## Clock and Reset Management

The clocking and reset block is responsible for clock generation, reset management, and phase shifting of clocks. It also has control of clock network types that route the clocks.

### Clock Management

Clock management circuitry is implemented by using:

- PLL
- DLL

#### PLL

The ALTMEMPHY MegaWizard Plug-In automatically generates an ALTPLL megafunction instance. The ALTPLL megafunction is responsible for generating the different clock frequencies and relevant phases used within the ALTMEMPHY megafunction.

The device families available have different PLL capabilities. The minimum PHY requirement is to have 16 phases of the highest frequency clock. The PLL uses **With no compensation** operation mode to minimize jitter. Changing the PLL compensation mode may result in inaccurate timing results.

You must choose a PLL and PLL input clock pin that are located on the same side of the device as the memory interface to ensure minimal jitter. Cascaded PLLs are not recommended as jitter can accumulate, causing the memory output clock to violate the memory device jitter specification. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, must to perform a manual PLL reset (by driving the `global_reset_n` signal low) and relock the PLL to ensure that the phase relationship between all PLL outputs are properly set.



For more information about the VCO frequency range and the available phase shifts, refer to the *Clock Networks and PLLs* chapter in the respective device family handbook.

Table 4-3 shows the PLL outputs and their usage for Stratix III and Stratix IV devices.

**Table 4-3.** QDRII+/QDRII SRAM Clocking in Stratix IV and Stratix III Devices

Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
phy_clk_1x	C0	30°	Half-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. It is set to 30° to ensure proper half-rate to full-rate transfer for write data and DQS.
mem_clk_2x	C1	0°	Full-Rate	Regional	For clocking the mem_clk generation block.
write_clk_2x	C3	-90°	Full-Rate	Regional	This clock is for clocking the data out from the DDIO pins in advance of the DQS strobe. As a result, its phase leads that of the mem_clk by 90°.
resync_clk_2x	C4	0°	Full-Rate	Regional	Unused in QDRII+/QDRII SRAM PHY.
measure_clk_1x	C5	0°	Half-Rate	Regional	Unused in QDRII+/QDRII SRAM PHY.
ac_clk_1x	C6	Set in the GUI	Half-Rate	Regional	Address and command clock.

**Note to Table 4-3:**

(1) The \_1x notation represents a frequency that is half of the memory clock frequency; the \_2x notation represents the memory clock frequency.

### DLL

The DLL settings are set depending on the memory clock frequency of operation.

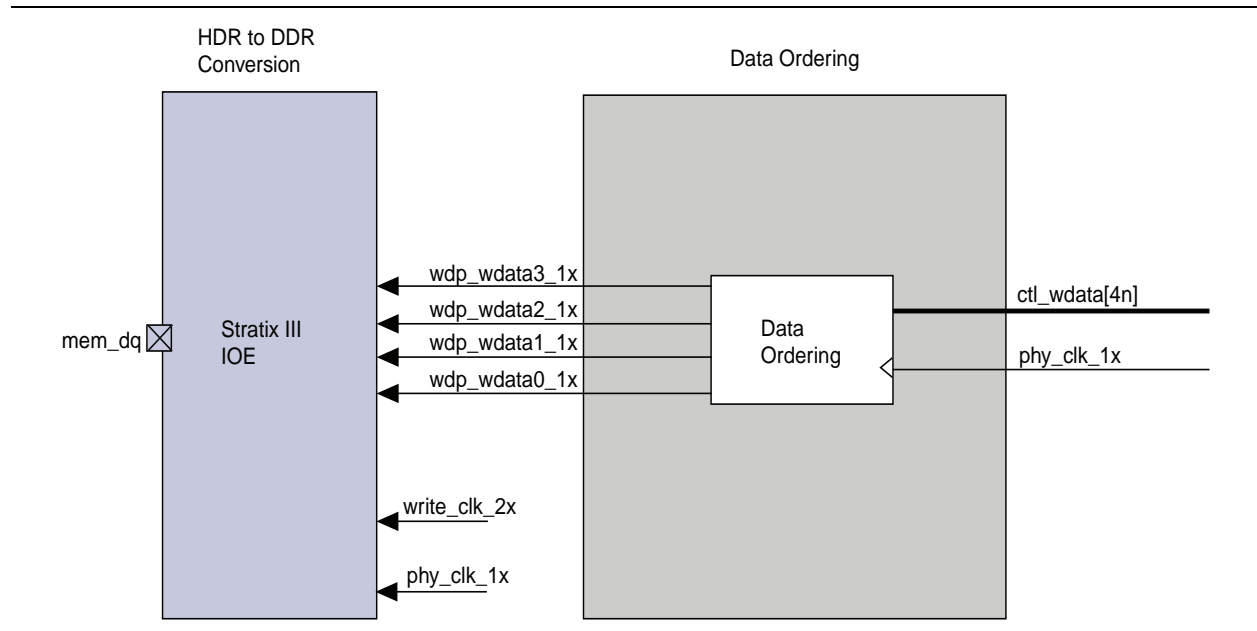
### Reset Management

Figure 4-3 on page 4-14 shows the reset management block for the QDRII+/QDRII SRAM PHY.

## Write Datapath

The memory controller interface outputs four  $n$ -bit wide data (ctl\_wdata) at phy\_clk\_1x frequency. The write data is clocked by the system clock phy\_clk\_1x at half-data rate (HDR) and reordered into HDR of width four  $n$ -bits represented in Figure 4-7 by wdp\_wdata3\_1x, wdp\_wdata2\_1x, wdp\_wdata1\_1x, and wdp\_wdata0\_1x.

Figure 4-7 shows the reordered or the reordered-and-delayed HDR data is then converted to DDR data within the IOE element using both the half-rate and full-rate clocks.

**Figure 4-7.** QDRII+/QDRII SRAM Write Datapath in Stratix IV and Stratix III Devices

## Address and Command Datapath

The address and command clock is one of the PLL dedicated clock outputs whose phase can be adjusted to meet the setup and hold requirements of the memory clock. The Stratix III address and command clock, `ac_clk_1x`, is half rate. The command and address pins use the DDIO output circuitry to launch commands from either the rising or falling edges of the clock.

## x36 Emulation for QDRII+/QDRII SRAM Interfaces

A few packages in the Stratix III and Stratix IV device families do not offer any  $\times 32/\times 36$  DQ/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 F484-pin packages for all Stratix III devices
- All I/O banks in F780-pin and F1152-pin packages for all Stratix III and Stratix IV devices
- Side I/O banks in F1517-pin and F1760-pin packages for all Stratix III devices
- All I/O banks in F1517-pin for EP4SGX180, EP4SGX230, EP4SGX290, EP4SGX360, and EP4SGX530 devices
- Side I/O banks in F1517-, F1760-, and F1932-pin packages for all Stratix IV devices

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





The maximum frequency supported in  $\times 36$  QDRII+/QDRII SRAM interfaces using  $\times 36$  emulation is lower than the maximum frequency when using a native  $\times 36$  DQ/DQS group.

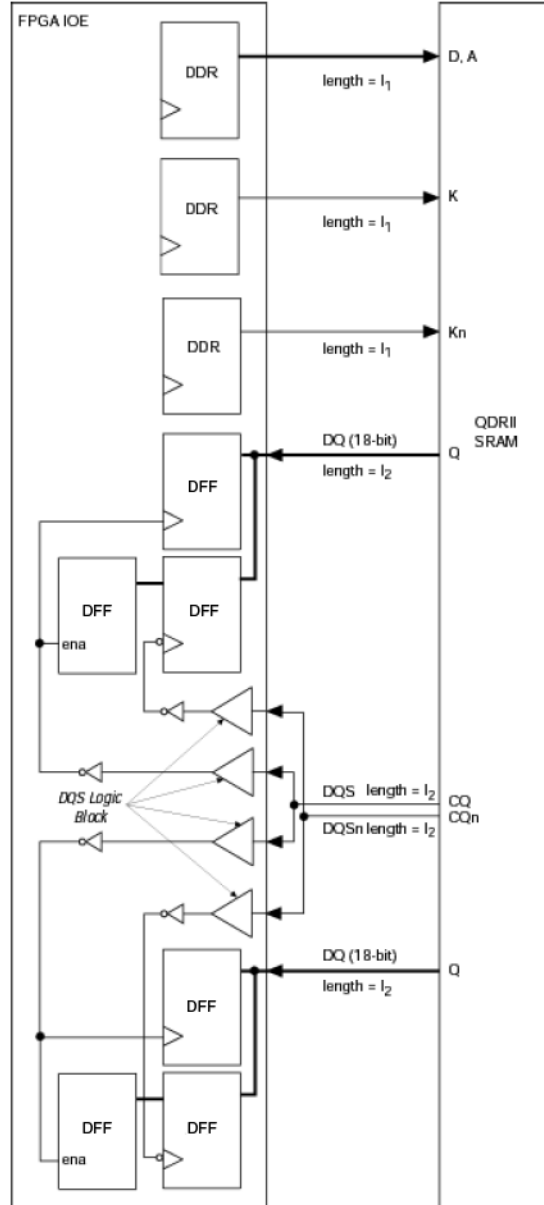
 Refer to the *DC & Switching Characteristics* chapters of the specific device handbook, for the maximum frequency supported.

 The F484-pin package cannot support  $\times 32/\times 36$  DQ/DQS groups as it does not even support  $\times 16/\times 18$  DQ/DQS groups.

To emulate a  $\times 32/\times 36$  DQ/DQS group, combine two  $\times 16/\times 18$  DQ/DQS groups together. For  $\times 36$  QDRII+/QDRII 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 QDRII+/QDRII SRAM device traces are then split on the board to connect to two pairs of CQ/CQn pins in the FPGA. This connection is the only connection 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 [Figure 4-8](#)). You cannot use the FPGA OCT features with  $\times 36$  emulation. You need to use external termination for the QDRII+/QDRII SRAM interface pins.

 Other QDRII+/QDRII SRAM interface rules for Stratix IV devices also apply for this implementation.

 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 length from the FPGA pin to the Y or T junction on the board. The total trace length from the memory device to each pin on the FPGA should match the Q trace length ( $I_2$ ).

**Figure 4–8.** Board Trace Connection for Emulated x36 QDRII+/QDRII SRAM Interface


### Timing Impact on x36 Emulation

With x36 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 ALTMEMPHY Preset Editor that you are using x36 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 ALTMEMPHY wizard.

 For more information on how to determine the CQ/CQn arrival time skew, see “Determining the CQ/CQn Arrival Time Skew” on page 4-26. For more information on the ALTMEMPHY Preset Editor settings for  $\times 36$  emulation, see “Creating an Emulated  $\times 36$  QDRII+/QDRII SRAM ALTMEMPHY Variation” on page 2-37.


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/DQ group, as indicated in the Stratix III and Stratix IV device family handbook.

### Rules to Combine Groups


Table 4-4 and Table 4-5 show the possible combinations to use two  $\times 16/\times 18$  DQ/DQS groups to form a  $\times 32/\times 36$  group on Stratix III and Stratix IV devices lacking a native  $\times 32/\times 36$  DQ/DQS groups. The tables show the combinations as I/O bank combinations, as the two  $\times 16/\times 18$  groups come from two different banks. You can use any of these combinations as read or write 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 II software.

 I/O bank 1C in both Stratix III and Stratix IV devices has dual-function configuration pins. Some of the DQ/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  $\times 36$  QDRII+/QDRII SRAM interface on one side of the device, by changing the **Memory Interface Data Group** default assignment, from the default 18 to 9.

 The ALTMEMPHY megafunction does not support  $\times 36$  mode emulation hybrid interface, where the  $\times 36$  group consists of a  $\times 18$  group from the top/bottom I/O bank and a  $\times 18$  group from the side I/O banks.

**Table 4-4.** Possible Group Combinations in Stratix III Devices

Package	Device Density	I/O Bank Combinations
780-pin FineLine BGA	EP3SE50, EP3SL50, EP3SL70, EP3SE80, EP3SE110, EP3SL110, EP3SL150, EP3SL200, EP3SE260	1A and 2A, 5A and 6A , 3A and 4A, 7A and 8A
1,152-pin FineLine BGA	EP3SE80 EP3SE110 EP3SL110 EP3SL150 EP3SL200 EP3SE260 EP3SL340	1A and 1C, 2A and 2C, 3A and 3B, 4A and 4B, 5A and 5C, 6A and 6C, 7A and 7B, 8A and 8B.
1,517-pin FineLine BGA	EP3SL200 EP3SE260 EP3SL340	1A and 1B, 2A and 2B or 1B and 1C, 2B and 2C, (2) 5A and, 5B, 6A and 6B or 5B and 5C, 6B and 6C, (2)
1760-pin FineLine BGA (1)	EP3SL340	1A and 1B, 2A and 2B or 1B and 1C, 2B and 2C, (2) 5A and, 5B, 6A and 6B or 5B and 5C, 6B and 6C, (2)

**Notes to Table 4-4:**

- (1) This device supports  $\times 36$  DQ/DQS groups on the top and bottom I/O banks natively.
- (2) You can combine the  $\times 16/\times 18$  DQ/DQS groups from I/O banks 1A and 1C, 2A and 2C, 5A and 5C, 6A and 6C, however this process is discouraged because of the size of the package. Similarly, crossing a bank number (for example combining groups from I/O banks 6C and 5C) is not supported in this package.

**Table 4-5.** Possible Group Combinations in Stratix IV Devices (Part 1 of 2)

Package	Device Density	I/O Bank Combinations
780-pin FineLine BGA	EP4SGX70 EP4SGX110 EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360	3A and 4A, 7A and 8A (bottom and top I/O banks) (1)
	EP4SE110 EP4SE230 EP4SE290 EP4SE360	1A and 2A, 5A and 6A (left and right I/O banks) 3A and 4A, 7A and 8A (bottom and top I/O banks) (1)

**Table 4-5.** Possible Group Combinations in Stratix IV Devices (Part 2 of 2)

Package	Device Density	I/O Bank Combinations
1,152-pin FineLine BGA	EP4SGX70 EP4SGX110	3A and 4A, 7A and 8A (bottom and top I/O banks) (1)
	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530	1A and 1C, 6A and 6C (left and right I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 7A and 7B, 8A, 8B (top I/O banks)
	EP4SE290 EP4SE360 EP4SE530 EP4SE680	1A and 1C, 2A and 2C (left I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 5A and 5C, 6A and 6C (right I/O banks) 7A and 7B, 8A and 8B (top I/O banks)
1,517-pin FineLine BGA	EP4SGX180 EP4SGX230 EP4SGX290 EP4SGX360 EP4SGX530 EP4S40G2 (2) EP4S100G2 (2) EP4S40G5 (2) EP4S100G5 (2)	1A and 1C, 2A and 2C (left I/O banks) 3A and 3B, 4A and 4B (bottom I/O banks) 5A and 5C, 6A and 6C (right I/O banks) 7A and 7B, 8A and 8B (top I/O banks)
	EP4SE290 (3) EP4SE360 (3)	1A and 1C, 2A and 2C (left I/O banks) 5A and 5C, 6A and 6C (right banks)
	EP4SE530 (3) EP4SE680 (3)	1A and 1B, 2A and 2B or 1B and 1C, 2B and 2C (left I/O banks) (4) 5A and 5B, 6A and 6B or 5B and 5C, 6B and 6C (right I/O banks)
1,760-pin FineLine BGA	EP4SGX290 EP4SGX360 EP4SGX530	1A & 1C, 2A & 2C (left I/O banks) 3A & 3B, 4A & 4B (bottom I/O banks) 5A & 5C, 6A & 6C (right I/O banks) 7A & 7B, 8A & 8B (top I/O banks)
	EP4SE530 (3) EP4SE680 (3)	1A & 1B, 2A & 2B or 1B & 1C, 2B & 2C (left I/O banks) (4) 5A & 5B, 6A & 6B or 5B & 5C, 6B & 6C (right I/O banks) (4)
1,932-pin FineLine BGA	EP4SGX290 (3) EP4SGX360 (3) EP4SGX530 (3)	1A and 1C, 2A and 2C (left I/O banks) 5A and 5C, 6A and 6C (right I/O banks)

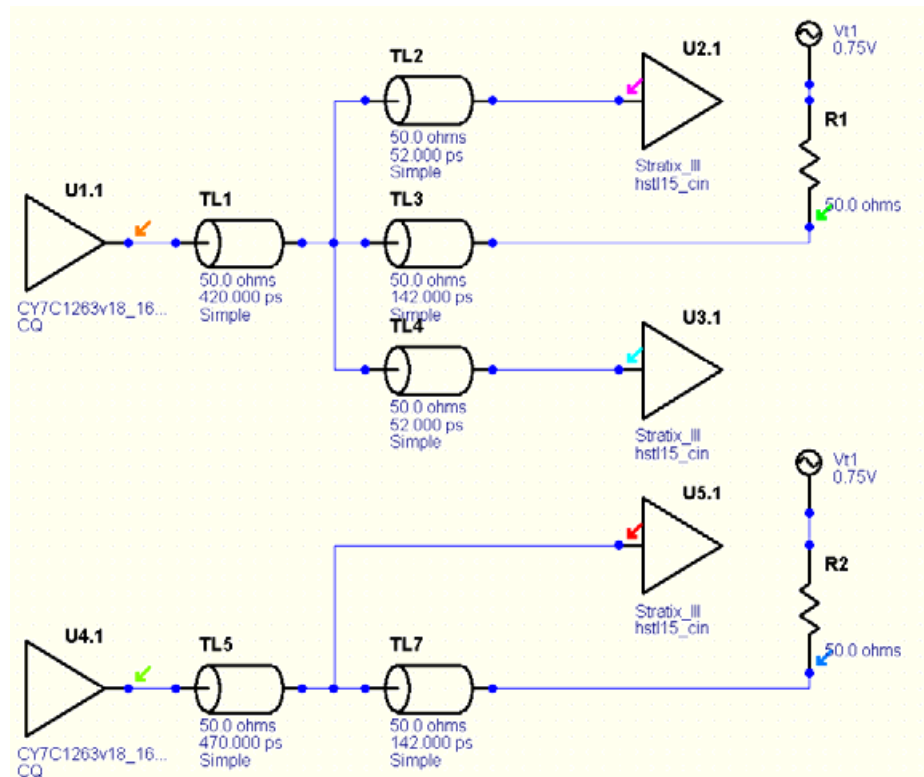
**Notes to Table 4-5:**

- (1) Each side of the device in these packages has four remaining  $\times 8/\times 9$  groups. You can combine them for the write side (only). If you want to keep the  $\times 36$  QDRII+/QDRII SRAM interface on one side of the device, change the Memory Interface Data Group default assignment from the default 18 to 9.
- (2) The Quartus II software does not support external memory interface using DQS/DQ groups from the left and right I/O banks in these devices.
- (3) This device supports  $\times 36$  DQ/DQS groups on the top and bottom I/O banks natively.
- (4) You can combine the  $\times 16/\times 18$  DQ/DQS groups from I/O banks 1A and 1C, 2A and 2C, 5A and 5C, 6A and 6C, however, this practise is discouraged because of the size of the package. Similarly, crossing a bank number (for example combining groups from I/O banks 6C and 5C) is not supported in this package.

### Determining the CQ/CQn Arrival Time Skew

Before compiling a design in Quartus II, 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 QDRII+/QDRII SRAM interface in the Quartus II software. Figure 4-9 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.

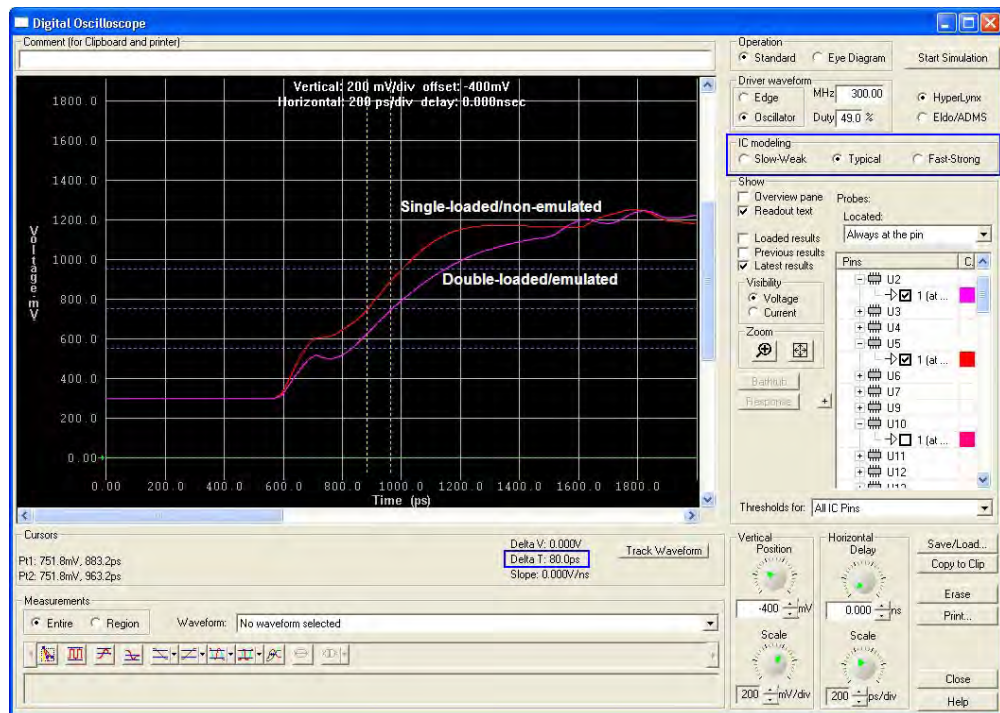
**Figure 4-9.** Board Simulation Topology Example



Run the simulation and look at the signal at the FPGA pin. Figure 4-10 shows an example of the simulation results from Figure 4-9. 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  $V_{REF}$  level (0.75 V in this case). Record the skew and re-run the simulation in the other two cases (slow-weak and fast-strong). To pick the largest and smallest skew to be included in Quartus II 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 4-10. Board Simulation Results



## Dynamic OCT Support

This section discusses dynamic OCT support for DDR, DDR2, and DDR3 (without leveling only) SDRAM in Stratix IV and Stratix III devices.

 QDR II SRAM does not need OCT as the signals are not bidirectional.

 Do not confuse OCT with ODT, which is a similar termination scheme but is implemented at the memory chip side.

If you turn on **Enable dynamic parallel on-chip termination** in the MegaWizard interface, any DDR, DDR2, or DDR3 SDRAM controller that uses ALTMEMPHY must allow a certain number of idle cycles, when switching from reads to writes (and writes to reads), to allow time for OCT switching.

Stratix III and Stratix IV devices support dynamic OCT. Dynamic OCT enables series termination ( $R_S$ ) and parallel termination ( $R_T$ ) to be dynamically turned on or off during the data transfer.

 For more information, refer to the [Termination Solutions in Stratix IV FPGAs](#) webpage.

The ALTMEMPHY controls the timing of the switching of  $R_T$  and  $R_S$ , implementing a park at  $R_S$  policy. This policy ensures that  $R_T$  is turned off during periods of inactivity on the DQ/DQS bus, which in turn ensures that overall power consumption is minimized.

The OCT hardware requires one full-rate memory clock cycle to switch from  $R_S$  to  $R_T$ , and some number of additional cycles are required to accommodate the DQS preamble or postamble sequence. These extra cycles impose a restriction on how quickly any DDR SDRAM controllers that use ALTMEMPHY with OCT enabled may switch from read to write accesses and write to read access (Figure 4-11).

**Figure 4-11.** OCT Switching and Read Timing

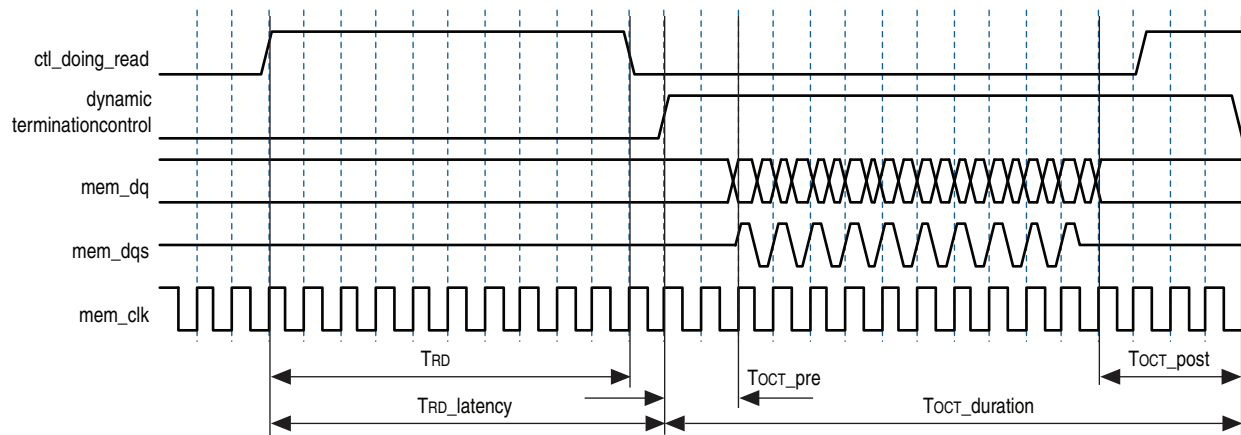


Figure 4-11 shows an incoming read from a controller and the resultant DQ/DQS signals and accompanying OCT signal. The controller asserts `ctl_doing_rd` for the duration of the read ( $T_{RD}$ ) and the ALTMEMPHY sequencer uses this signal with the calculated latencies in the system to produce the OCT signal, `dynamicterminationcontrol`.

The OCT signal is extended ( $Toct\_duration$ ) so that it meets the required timing constraints before and after the read occurs. The OCT control is driven the cycle before the DQS preamble, which is two cycles before the read data appears ( $Toct\_pre$ ). It is held for 3.5 cycles after the last data beat, to allow for postamble glitch settling with worst-case trace delay ( $Toct\_post$ ).


Consecutive reads or consecutive writes are unaffected by the  $Toct\_pre$  and  $Toct\_post$  timing requirements, as the OCT signal is continually driven either high or low. However, read to write and write to read transitions are affected as described.

## Write-to-Read Timing Requirement


The controller must ensure that for write-to-read transitions there is at least a 2 memory clock ( $Toct\_pre$ ) delay between issuing the write and read commands. This delay allows sufficient time for the OCT signal to switch before the read data comes back. DDR memories already have a requirement to add a 1 clock delay between writes and reads to allow for DQS preamble, so the OCT circuitry only adds one additional cycle delay.

## Read-to-Write Timing Requirement

The controller must ensure that for read-to-write transitions there is at least a 3.5 memory clock ( $T_{\text{OCT\_post}}$ ) delay between issuing the read and write commands. This delay allows sufficient time after the read completes for the OCT signal to switch off before the write data is driven onto DQ.

 In a configuration without OCT, there is still a requirement to add at least a single memory clock delay to allow for bus turnaround. So the additional overhead to accommodate the OCT requirements is up to 2.5 memory clock cycles.

The Altera high-performance controllers meet these read-to-write and write-to-read timing requirements.


 For more information, refer to either the *External Memory Interfaces in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook* or the *External Memory Interfaces in Stratix IV Devices* chapter in volume 1 of the *Stratix IV Device Handbook*.






## Introduction

The following sections describe the ALTMEMPHY megafunction support for Arria® GX, Arria II GX, Stratix® II, and Stratix II GX devices. Arria II GX devices support full- and half-rate DDR2 and DDR SDRAM interfaces and half-rate DDR3 SDRAM interfaces using ALTMEMPHY megafunction.

 HardCopy® II ASIC device support is similar to that of the Stratix II FPGA family. However, some design considerations are specific to the HardCopy II device family as previously noted in [“MegaWizard Plug-In Manager Page Descriptions”](#) on page 2–2.

 For more information about using the ALTMEMPHY Megafunction Plug-In Manager with HardCopy II ASICs, refer to [AN 463: Using ALTMEMPHY Megafunction with HardCopy II Structured ASICs](#).

The ALTMEMPHY megafunction does not natively support a memory interface that spans on multiple sides of the device in these device families, because the memory interface pins that are connected to the DLL are only available on the top and bottom of the device. In silicon, you can route the DLL control settings from one side of the device to another side of the device via local routing. To perform local routing, you must register the DLL control settings and to minimize the arrival skew of the DLL control settings at the other side of the device. However, this method has not been characterized and its performance is unknown. Therefore, this implementation is discouraged.

## DDR2/DDR SDRAM

Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX devices support both full-rate and half-rate DDR2/DDR SDRAM PHYs.

 Arria II GX devices offer enhanced IOE DDR capabilities by including the DDR registers, synchronization, and postamble registers directly in the IOE.

## Half-Rate Support

The following section discusses half-rate support for DDR2/DDR SDRAM for Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX devices.

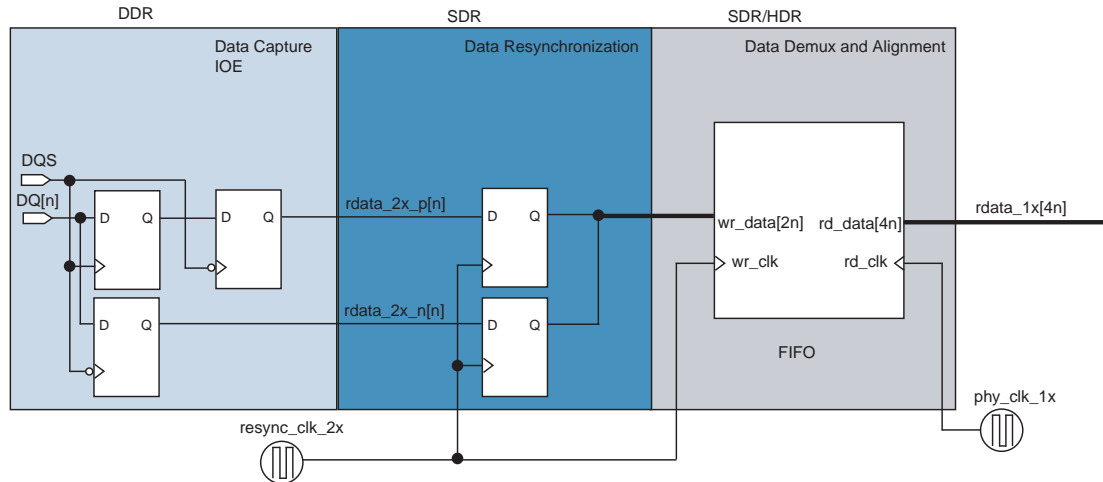
### Read Datapath

The read datapath logic is responsible for capturing data sent by the memory device and subsequently aligning the data back to the system clock domain. The following functions are performed by the read datapath:

1. Data capture and resynchronization
2. Data demultiplexing
3. Data alignment

Figure 5-1 shows the order of the functions performed by the read datapath, along with the frequency at which the read data is handled.

**Figure 5-1.** DDR2/DDR SDRAM Read Datapath in Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Devices (Note 1)



**Note to Figure 5-1:**

(1) In Arria II GX devices the resynchronization register is implemented in IOE.

### Data Capture and Resynchronization

Data capture and resynchronization is the process of capturing the read data (DQ) with the DQS strobe and re-synchronizing the captured data to an internal free-running full-rate clock supplied by the enhanced phase-locked loop (PLL).

The resynchronization clock is an intermediate clock whose phase shift is determined during the calibration stage.

Timing constraints ensure that the data resynchronization registers are placed close to the DQ pins to achieve maximum performance. Timing constraints also further limit skew across the DQ pins. The captured data ( $rdata\_2x\_p$  and  $rdata\_2x\_n$ ) is synchronized to the resynchronization clock ( $resync\_clk\_2x$ ), see Figure 5-1.

### Data Demultiplexing

Data demultiplexing is the process of SDR data into HDR data. Data demultiplexing is required to bring the frequency of the resynchronized data down to the frequency of the system clock, so that data from the external memory device can ultimately be brought into the FPGA DDR2/DDR SDRAM controller clock domain. Before data capture, the data is DDR and  $n$ -bit wide. After data capture, the data is SDR and  $2n$ -bit wide. After data demuxing, the data is HDR of width  $4n$ -bits wide. The system clock frequency is half the frequency of the memory clock.

Demultiplexing is achieved using a dual-port memory with a  $2n$ -bit wide write-port operating on the resynchronization clock (SDR) and a  $4n$ -bit wide read-port operating on the PHY clock (HDR). The basic principle of operation is that data is written to the memory at the SDR rate and read from the memory at the HDR rate while incrementing the read- and write-address pointers. As the SDR and HDR clocks are generated, the read and write pointers are continuously incremented by the same PLL, and the  $4n$ -bit wide read data follows the  $2n$ -bit wide write data with a constant latency.

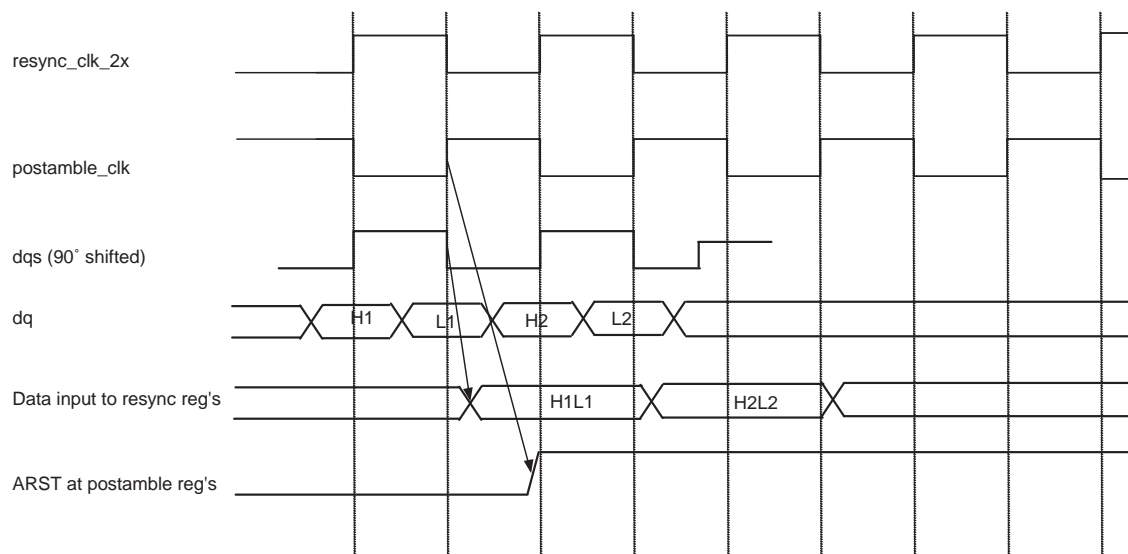
### Read Data Alignment

Data alignment is the process controlled by the sequencer to ensure the correct captured read data is present in the same half-rate clock cycle at the output of the read data DPRAM. Data alignment is implemented using either M4K or M512K memory blocks. The bottom of Figure 5-2 shows the concatenation of the read data into valid HDR data.

### Postamble Protection


The ALTMEMPHY megafunction provides the DQS postamble logic. The postamble clock is derived from the resynchronization clock and is the negative edge of the resynchronization clock. The ALTMEMPHY megafunction calibrates the resynchronization clock such that it is in the center of the data-valid window. The clock that controls the postamble logic, the postamble clock, is the negative edge of the resynchronization clock. No additional clocks are required. Figure 5-2 shows the relationship between the postamble clock and the resynchronization clock.

**Figure 5-2.** Relationship Between Postamble Clock and Resynchronization Clock (Note 1)



**Note to Figure 5-2:**

(1) `resync_clk_2x` is delayed further to allow for the I/O element (IOE) to core transition time.

 For more information about the postamble circuitry, refer to the *External Memory Interfaces* chapter in the *Stratix II Device Handbook*.

## Clock and Reset Management

The clocking and reset block is responsible for clock generation, reset management, and phase shifting of clocks. It also has control of clock network types that route the clocks.

### Clock Management

The ability of the ALTMEMPHY megafunction to work out the optimum resynchronization clock phase during calibration, and to track the system voltage and temperature (VT) variations. Clock management is done by phase-shifting the clocks relative to each other.

Clock management circuitry is implemented by using the following device resources:

- PLL
- PLL reconfiguration
- DLL

### PLL

The ALTMEMPHY MegaWizard® Plug-In automatically generates an ALTPLL megafunction instance. The ALTPLL megafunction is responsible for generating the different clock frequencies and relevant phases used within the ALTMEMPHY megafunction.

The minimum PHY requirement is to have 16 phases of the highest frequency clock. The PLL uses **With no compensation** operation mode to minimize jitter.

You must choose a PLL and PLL input clock pin that are located on the same side of the memory interface to ensure minimal jitter. Cascaded PLLs are not recommended for DDR2/DDR SDRAM interfaces as jitter can accumulate with the use of cascaded PLLs causing the memory output clock to violate the memory device jitter specification. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset and relock the PLL to ensure that the phase relationship between all PLL outputs are properly set.



If the design cascades PLLs, the source (upstream) PLL should have a low-bandwidth setting; the destination (downstream) PLL should have a high-bandwidth setting. Adjacent PLLs cascading is recommended to reduce clock jitters.



For more information about the VCO frequency range and the available phase shifts, refer to the *PLLs in Stratix II and Stratix II GX Devices* chapter in the respective device family handbook.

Table 5-1 shows the clock outputs that Arria GX, HardCopy II, Stratix II and Stratix II GX devices use; Table 5-2 shows the clock outputs that Arria II GX devices use.

**Table 5-1.** DDR2/DDR SDRAM Clocking in Arria GX, HardCopy II, Stratix II, and Stratix II GX Devices (Part 1 of 2)

Design Rate	Clock Name	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate	phy_clk_1x and aux_half_rate_clk	C0	0°	Half-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. This clock also feeds into a divider circuit to provide the PLL scan_clk signal (for reconfiguration) that must be lower than 100 MHz.
	mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	This clock is for clocking DQS and as a reference clock for the memory devices.
Full rate	aux_half_rate_clk	C0	0°	Half-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. This clock also feeds into a divider circuit to provide the PLL scan_clk signal (for reconfiguration) that must be lower than 100 MHz.
	phy_clk_1x (1) and mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	This clock is for clocking DQS and as a reference clock for the memory devices.
Half-rate and full rate	write_clk_2x	C2	-90°	Full-Rate	Global	This clock is for clocking the data out of the DDR I/O (DDIO) pins in advance of the DQS strobe (or equivalent). As a result, its phase leads that of the mem_clk_2x by 90°.

**Table 5-1.** DDR2/DDR SDRAM Clocking in Arria GX, HardCopy II, Stratix II, and Stratix II GX Devices (Part 2 of 2)

Design Rate	Clock Name	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate and full rate	mem_clk_ext_2x	C3	> 0°	Full-Rate	Dedicated	This clock is only used if the memory clock generation uses dedicated output pins. Applicable only in HardCopy II or Stratix II prototyping for HardCopy II designs.
Half-rate and full rate	resync_clk_2x	C4	Calibrated	Full-Rate	Regional	Clocks the resynchronization registers after the capture registers. Its phase is adjusted to the center of the data valid window across all the DQS-clocked DDIO groups.
Half-rate and full rate	measure_clk_2x	C5	Calibrated	Full-Rate	Regional (2)	This clock is for VT tracking. This free-running clock measures relative phase shifts between the internal clock(s) and those being fed back through a mimic path. As a result, the ALTMEMPHY megafunction can track VT effects on the FPGA and compensate for the effects.
Half-rate and full rate	ac_clk_2x	—	0, 90°, 180°, 270°	Full-Rate	Global	The ac_clk_2x clock is derived from either mem_clk_2x (when you choose 0° or 180° phase shift) or write_clk_2x (when you choose 90° or 270° phase shift). See “Half-Rate Address and Command Clocking” on page 3-24 for illustrations of the address and command clock relationship with the mem_clk_2x or write_clk_2x signals.

**Notes to Table 5-1:**

- (1) In full-rate designs a \_1x clock may run at full-rate clock rate.
- (2) This clock should be of the same clock network clock as the resync\_clk\_2x clock.

**Table 5-2.** DDR2/DDR SDRAM Clocking in Arria II GX Devices (Part 1 of 2)

Design Rate	Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate	phy_clk_1x and aux_half_rate_clk	C0	0°	Half-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. This clock also feeds into a divider circuit to provide the PLL scan_clk signal (for reconfiguration) that must be lower than 100 MHz.
	mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	This clock is for clocking DQS and as a reference clock for the memory devices.
Full rate	aux_half_rate_clk	C0	0°	Half-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. This clock also feeds into a divider circuit to provide the PLL scan_clk signal (for reconfiguration) that must be lower than 100 MHz.
	phy_clk_1x (1) and mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	This clock is for clocking DQS and as a reference clock for the memory devices.
Half-rate and full rate	mem_clk_1x	C2	0°	Half-Rate	Global	This clock is for clocking DQS and as a reference clock for the memory devices.
Half-rate and full rate	write_clk_2x	C3	-90°	Full-Rate	Global	This clock is for clocking the data out of the DDR I/O (DDIO) pins in advance of the DQS strobe (or equivalent). As a result, its phase leads that of the mem_clk_2x by 90°.

**Table 5-2.** DDR2/DDR SDRAM Clocking in Arria II GX Devices (Part 2 of 2)

Design Rate	Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate and full rate	ac_clk_2x	C3	90°	Full-Rate	Global	Address and command clock. The ac_clk_2x clock is derived from either mem_clk_2x (when you choose 0° or 180° phase shift) or write_clk_2x (when you choose 90° or 270° phase shift). See “Half-Rate Address and Command Clocking” on page 3-24 for illustrations of the address and command clock relationship with the mem_clk_2x or write_clk_2x signals.
Half-rate and full rate	cs_n_clk_2x	C3	90°	Full-Rate	Global	Memory chip-select clock. The cs_n_clk_2x clock is derived from ac_clk_2x.
Half-rate and full rate	resync_clk_2x	C4	Calibrated	Full-Rate	Global	Clocks the resynchronization registers after the capture registers. Its phase is adjusted to the center of the data valid window across all the DQS-clocked DDIO groups.
Half-rate and full rate	measure_clk_2x	C5	Calibrated	Full-Rate	Global	This clock is for VT tracking. This free-running clock measures relative phase shifts between the internal clock(s) and those being fed back through a mimic path. As a result, the ALTMEMPHY megafunction can track VT effects on the FPGA and compensate for the effects.

**Note to Table 5-2:**

(1) In full-rate designs a \_1x clock may run at full-rate clock rate.



### PLL Reconfiguration

The ALTMEMPHY MegaWizard Plug-In automatically generates the PLL reconfiguration block by instantiating an ALTPLL\_RECONFIG variation for Stratix II and Stratix II GX devices to match the generated ALTPLL megafunction instance. The ALTPLL\_RECONFIG megafunction varies the resynchronization clock phase and the measure clock phase.



The ALTMEMPHY MegaWizard Plug-In does not instantiate an ALTPLL\_RECONFIG megafunction for Arria II GX devices, as this device uses the dedicated phase stepping I/O on the PLL.

### DLL

A DLL instance is included in the generated ALTMEMPHY variation. When using the DQS to capture the DQ read data, the DLL center-aligns the DQS strobe to the DQ data. The DLL settings depend on the interface clock frequency.



For more information, refer to the *External Memory Interfaces* chapter in the device handbook for your target device family.

### Reset Management

The reset management block is responsible for the following:

- Provides appropriately timed resets to the ALTMEMPHY megafunction datapaths and functional modules
- Performs the reset sequencing required for different clock domains
- Provides reset management of PLL and PLL reconfiguration functions
- Manages any circuit-specific reset sequencing

Each reset is an asynchronous assert and synchronous de-assert on the appropriate clock domain. The reset management design uses a standard two-register synchronizer to avoid metastability. A unique reset metastability protection circuit for the clock divider circuit is required because the `phy_clk` domain reset metastability protection flipflops have fan-in from the `soft_reset_n` input, and so these registers cannot be used.

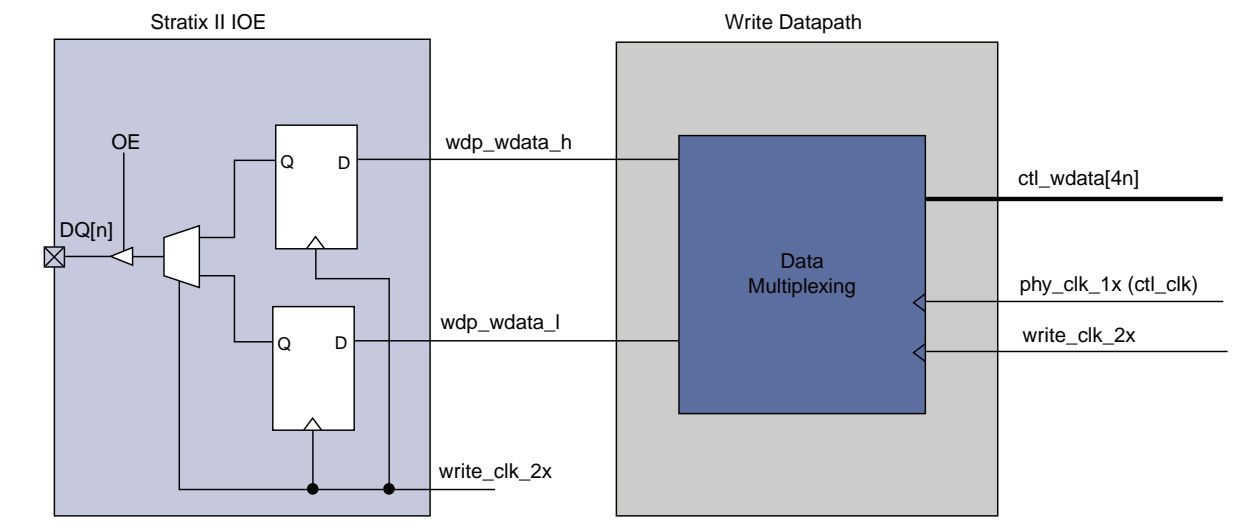
Figure 5-3 shows the ALTMEMPHY reset management block for Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX devices. The `pll_ref_clk` signal goes directly to the PLL, eliminating the need for global clock network routing. If you are using the `pll_ref_clk` signal to feed other parts of your design, you must use a global clock network for the signal. If `pll_reconfig_soft_reset_en` is held low, the PLL reconfig is not reset during a soft reset, which allows designs targeting HardCopy II devices to hold the PHY in reset while still accessing the PLL reconfig block. However, designs targeting Arria GX, Arria II GX, or Stratix II devices are expected to tie the `pll_reconfig_soft_en` shell to VCC to enable PLL reconfig soft resets.



The memory controller interface outputs  $4n$ -bit wide data ( $ctl\_wdata[4n]$ ) at half-rate frequency. Figure 5-4 shows that the HDR write data ( $ctl\_wdata[4n]$ ) is clocked by the half-rate clock  $phy\_clk\_1x$  and is converted into SDR which is represented by  $wdp\_wdata\_h$  and  $wdp\_wdata\_l$  and clocked by the full-rate clock  $write\_clk\_2x$ .

The DQ IOEs convert  $2n$ -SDR bits to  $n$ -DDR bits.

**Figure 5-4.** DDR2/DDR SDRAM Write Datapath in Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Devices



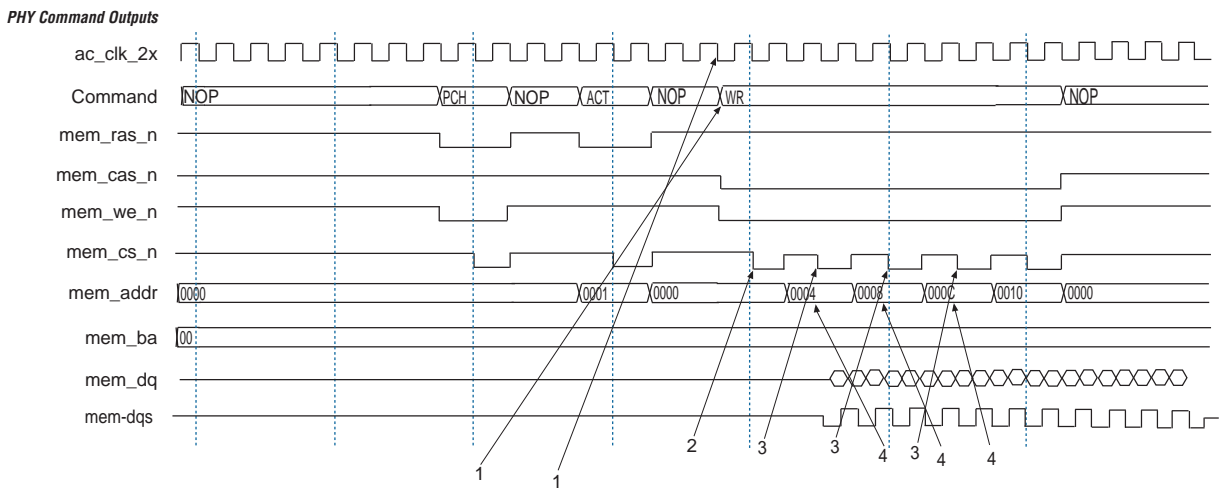
### Address and Command Datapath

The address and command datapath is responsible for taking the address and command outputs from the controller and converting them from half-rate clock to full-rate clock. Two types of addressing are possible:

- 1T (full rate)—The duration of the address and command is a single memory clock cycle ( $mem\_clk\_2x$ , Figure 5-5). This applies to all address and command signals in full-rate designs or  $mem\_cs\_n$ ,  $mem\_cke$ , and  $mem\_odt$  signals in half-rate designs.
- 2T (half rate)—The duration of the address and command is two memory clock cycles. For half-rate designs, the ALTMEMPHY megafunction supports only a burst size of four, which means the burst size on the local interface is always set to 1. The size of the data is  $4n$ -bits wide on the local side and is  $n$ -bits wide on the memory side. To transfer all the  $4n$ -bits at the double data rate, two memory-clock cycles are required. The new address and command can be issued to memory every two clock cycles. This scheme applies to all address and command signals, except for  $mem\_cs\_n$ ,  $mem\_cke$ , and  $mem\_odt$  signals in half-rate mode.

Refer to Table 5-1 in “PLL” on page 5-4 to see the frequency relationship of  $mem\_clk\_2x$  with the rest of the clocks.


Figure 5-5 shows a 1T chip select signal ( $mem\_cs\_n$ ), which is active low, and disables the command in the memory device. All commands are masked when the chip-select signal is inactive. The  $mem\_cs\_n$  signal is considered part of the command code.


**Figure 5-5.** Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX Address and Command Datapath

The command interface is made up of the signals `mem_ras_n`, `mem_cas_n`, `mem_we_n`, `mem_cs_n`, `mem_cke`, and `mem_odt`.

The waveform in [Figure 5-5](#) shows a NOP command followed by five back-to-back write commands.

1. The commands are asserted either on the rising edge of `ac_clk_2x`. The `ac_clk_2x` is derived from either `mem_clk_2x` ( $0^\circ$ ), `write_clk_2x` ( $270^\circ$ ), or the inverted variations of those two clocks (for  $180^\circ$  and  $90^\circ$  phase shifts). This depends on the setting of the address and command clock in the ALTMEMPHY MegaWizard Plug-In. See [“Half-Rate Address and Command Clocking”](#) on [page 3-24](#) for illustrations of this clock in relation to the `mem_clk_2x` or `write_clk_2x` signals.
2. All address and command signals (except for `mem_cs_n`, `mem_cke`, and `mem_odt` signals) remain asserted on the bus for two clock cycles, allowing sufficient time for the signals to settle.
3. The `mem_cs_n`, `mem_cke`, and `mem_odt` signals are asserted during the second cycle of the address/command phase.
4. By asserting the chip-select signal in alternative cycles, back-to-back read or write commands can be issued.
5. The address is incremented every other `ac_clk_2x` cycle.

 The `ac_clk_2x` clock is derived from either `mem_clk_2x` (when you choose  $0^\circ$  or  $180^\circ$  phase shift) or `write_clk_2x` (when you choose  $90^\circ$  or  $270^\circ$  phase shift).

 The address and command clock can be  $0^\circ$ ,  $90^\circ$ ,  $180^\circ$ , or  $270^\circ$  from the system clock (refer to [“Half-Rate Address and Command Clocking”](#) on [page 3-24](#)).

## Full-Rate Support

The following section discusses full-rate support for Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX devices.

### Read Datapath

The full-rate datapath is similar to the half-rate datapath. The full-rate datapath also consists of a RAM with the same width as the data input (just like that of the half-rate), but the width on the data output of the RAM is half that of the half-rate PHY. The function of the RAM is to transfer the read data from the resynchronization clock domain to the system clock domain.

### Postamble Protection

The postamble protection is the same as the half-rate support.

### Clock and Reset Management

For full-rate clock and reset management refer to . The PLL is configured exactly in the same way as in half-rate designs. The PLL information and restriction from half-rate designs also applies.



The `phy_clk_1x` clock is now full-rate, despite the “1x” naming convention.

You must choose a PLL and PLL input clock pin that are located on the same side of the memory interface to ensure minimal jitter. Cascaded PLLs are not recommended for DDR2/DDR SDRAM interfaces as jitter can accumulate with the use of cascaded PLLs causing the memory output clock to violate the memory device jitter specification. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset and relock the PLL to ensure that the phase relationship between all PLL outputs are properly set. The PLL restrictions in half-rate designs also applies to full-rate designs.

### Write Datapath

The write datapath is similar to the half-rate PHY. The IOE block is identical to the half-rate PHY. The latency of the write datapath in the full-rate PHY is less than in the half-rate PHY because the full-rate PHY does not have the half-rate-to-full-rate conversion logic.

### Address and Command Datapath

The address and command datapath for full-rate designs is similar to half-rate designs, except that the address and command signals are all asserted for one memory clock cycle only (1T signaling).

## DDR3 SDRAM

For Arria II GX devices, the ALTMEMPHY megafunction supports a fully-calibrated DDR3 SDRAM PHY without leveling for  $\times 8$  or  $\times 4$  devices with a 300-MHz frequency target.

ALTMEMPHY only supports DDR3 SDRAM components interfacing for Arria II GX devices using T-topology for clock, address, and command bus. DDR3 SDRAM DIMMs (fly-by clock, address, and command topology) are not supported, as Arria II GX devices do not support read and write leveling. The ALTMEMPHY variation supports half-rate DDR3 SDRAM interfaces only.



Arria GX, Stratix II and Stratix II GX devices do not support DDR3 SDRAM.

## Read Datapath

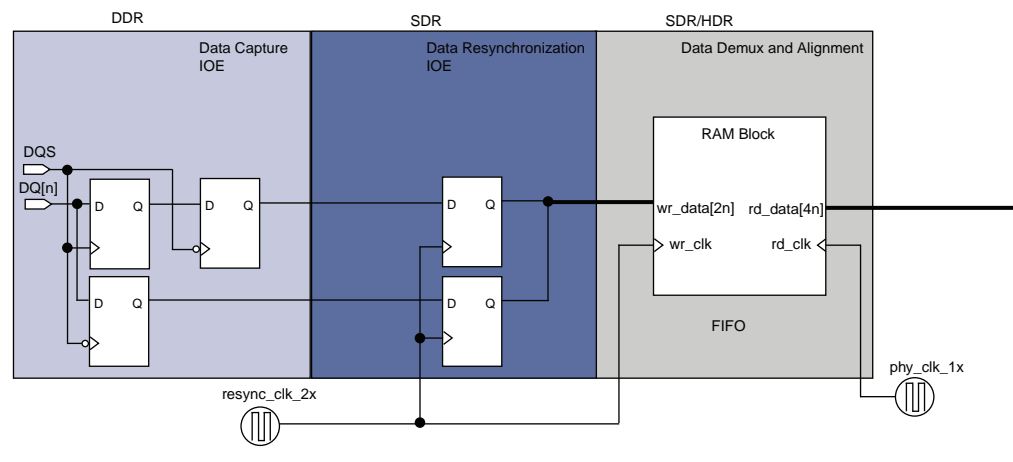
The read datapath logic captures data sent by the memory device and subsequently aligns the data back to the system clock domain. The read datapath for DDR3 SDRAM consists of the following three main blocks:

- Data capture
- Data resynchronization
- Data demultiplexing and alignment

As the DQS/DQS<sub>n</sub> signal is not continuous, the PHY also has postamble protection logic to ensure that any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches.

Figure 5-6 shows the order of the functions performed by the read datapath and the frequency at which the read data is handled.

**Figure 5-6.** DDR3 SDRAM Read Datapath in Arria II GX Devices



## Data Capture and Resynchronization

The data capture and resynchronization registers for Arria II GX devices are implemented in the I/O element (IOE) to achieve maximum performance. Data capture and resynchronization is the process of capturing the read data (DQ) with the DQS/DQS<sub>n</sub> strobcs and resynchronizing the captured data to an internal free-running full-rate clock supplied by the enhanced PLL. The resynchronization clock is an intermediate clock whose phase shift is determined during the calibration stage. The captured data (`rdata_p_captured` and `rdata_n_captured`) is synchronized to the resynchronization clock (`resync_clk_2x`), see Figure 5-6. For Arria II GX devices, the ALTMEMPHY instances an ALTDQ\_DQS megafunction that instantiates the required IOEs for all the DQ and DQS pins.

## Data Demultiplexing

Data demultiplexing is the process of changing the SDR data into HDR data. Data demultiplexing is required to bring the frequency of the resynchronized data down to the frequency of the system clock, so that data from the external memory device can ultimately be brought into the FPGA controller clock domain. Before data capture, the data is DDR and  $n$ -bit wide. After data capture, the data is SDR and  $2n$ -bit wide. After data demuxing, the data is HDR of width  $4n$ -bits wide. The system clock frequency is half the frequency of the memory clock. Demultiplexing is achieved using a dual-port memory with a  $2n$ -bit wide write-port operating on the resynchronization clock (SDR) and a  $4n$ -bit wide read-port operating on the PHY clock (HDR). The basic principle of operation is that data is written to the memory at the SDR rate and read from the memory at the HDR rate while incrementing the read- and write-address pointers. As the SDR and HDR clocks are generated, the read and write pointers are continuously incremented by the same PLL, and the  $4n$ -bit wide read data follows the  $2n$ -bit wide write data with a constant latency

## Read Data Alignment

Data alignment is the process controlled by the sequencer to ensure the correct captured read data is present in the same half-rate clock cycle at the output of the read data DPRAM. Data alignment is implemented using memory blocks in the core of devices.

## Postamble Protection

A dedicated postamble register controls the gating of the shifted DQS signal that clocks the DQ input registers at the end of a read operation. Any glitches on the DQS input signals at the end of the read postamble time do not cause erroneous data to be captured as a result of postamble glitches. The postamble path is also calibrated to determine the correct clock cycle, clock phase shift, and delay chain settings.

## Clock and Reset Management

The clocking and reset block is responsible for clock generation, reset management, and phase shifting of clocks and control of clock network types that route the clocks. These tasks are handled in the `<variation_name>_alt_mem_phy_clk_reset` module in the `<variation_name>_alt_mem_phy.v` or `.vhd` file.

### Clock Management

The ability of the ALTMEMPHY megafunction to work out the optimum phase during calibration and to track voltage and temperature variation relies on phase shifting the clocks relative to each other.



Certain clocks require phase shifting during the ALTMEMPHY megafunction operation.

Clock management circuitry is implemented by using the following components:

- PLL
- DLL

## PLL

The ALTMEMPHY MegaWizard Plug-In automatically generates an ALTPLL megafunction instance. The ALTPLL megafunction is responsible for generating the different clock frequencies and relevant phases used within the ALTMEMPHY megafunction. The available device families have different PLL capabilities. The minimum PHY requirement is to have 16 phases of the highest frequency clock. The PLL uses **With no compensation** operation mode to minimize jitter.

The input clock to the PLL does not have any other fan-out to the PHY, so you do not have to use a global clock resource for the path between the clock input pin to the PLL. You must use the PLL located on the same device quadrant or side of the memory interface and the corresponding clock input pin for that PLL, to ensure optimal performance and accurate timing results from the Quartus II software. You must choose a PLL and PLL input clock pin that are located on the same side of the memory interface to ensure minimal jitter. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset (by driving the `global_reset_n` signal low) and relock the PLL to ensure that the phase relationship between all PLL outputs is properly set.



If PLLs are cascaded in your designs, you must use the adjacent PLL (direct connection) method to reduce clock jitter.



For more information about the VCO frequency range and the available phase shifts, refer to the *Clock Networks and PLLs in Arria II GX Devices* chapter in volume 1 of the *Arria II GX Device Handbook*.

Table 5-3 shows the clock outputs that Arria II GX devices use.

**Table 5-3.** DDR3 SDRAM Clocking in Arria II GX Devices (Part 1 of 3)

Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
phy_clk_1x and aux_half_rate_clk	C0	0°	Half-Rate	Global	The only clock that is made available on the user interface of the ALTMEMPHY megafunction. This clock also feeds into a divider circuit to provide the PLL <code>scan_clk</code> signal (for reconfiguration) that must be lower than 100 MHz.
mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	This clock is for clocking DQS and as a reference clock for the memory devices.
mem_clk_1x	C2	0°	Half-Rate	Global	This clock is for clocking DQS and as a reference clock for the memory devices.



**Table 5-3.** DDR3 SDRAM Clocking in Arria II GX Devices (Part 2 of 3)

Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
write_clk_2x	C3	-90°	Full-Rate	Global	This clock is for clocking the data out of the DDR I/O (DDIO) pins in advance of the DQS strobe (or equivalent). As a result, its phase leads that of the mem_clk_2x by 90°.
ac_clk_2x	C3	-90°	Full-Rate	Global	Address and command clock. The ac_clk_2x clock is derived from either mem_clk_2x (when you choose 0° or 180° phase shift) or write_clk_2x (when you choose 90° or 270° phase shift). See “Half-Rate Address and Command Clocking” on page 3-24 for illustrations of the address and command clock relationship with the mem_clk_2x or write_clk_2x signals.
cs_n_clk_2x	C3	-90°	Full-Rate	Global	Memory chip-select clock. The cs_n_clk_2x clock is derived from ac_clk_2x.
resync_clk_2x	C4	Calibrated	Full-Rate	Global	Clocks the resynchronization registers after the capture registers. Its phase is adjusted to the center of the data valid window across all the DQS-clocked DDIO groups.

**Table 5-3.** DDR3 SDRAM Clocking in Arria II GX Devices (Part 3 of 3)

Clock Name (1)	Postscale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
measure_clk_2x	C5	Calibrated	Full-Rate	Global	This clock is for VT tracking. This free-running clock measures relative phase shifts between the internal clock(s) and those being fed back through a mimic path. As a result, the ALTMEMPHY megafunction can track VT effects on the FPGA and compensate for the effects.

**Note to Table 5-3:**

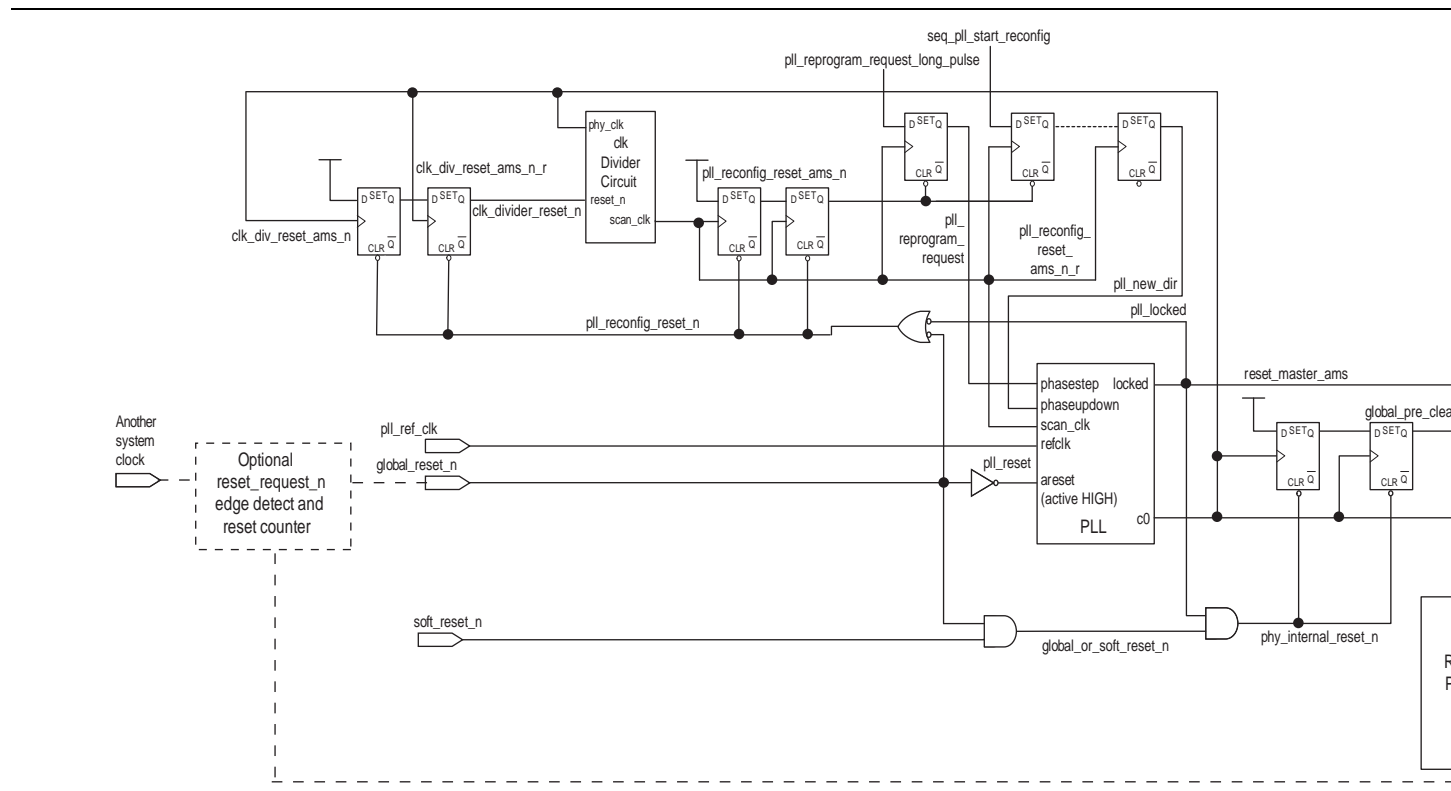
(1) The `_1x` clock represents a frequency that is half of the memory clock frequency; the `_2x` clock represents the memory clock frequency.

For Arria II GX devices, the PLL reconfiguration uses the phase-shift inputs on the PLL. The PLL reconfiguration megafunction is not required.

### Reset Management

Figure 5-7 shows the reset management block for the DDR3 SDRAM PHY. You can use the `pll_ref_clk` input to feed the optional `reset_request_n` edge detect and reset counter module. However, this requires the `pll_ref_clk` signal to use a global clock network resource. There is a unique reset metastability protection circuit for the clock divider circuit because the `phy_clk` domain reset metastability protection registers have fan-in from the `soft_reset_n` input so these registers cannot be used.

**Figure 5-7.** ALTMEMPHY Reset Management Block for Arria II GX Devices



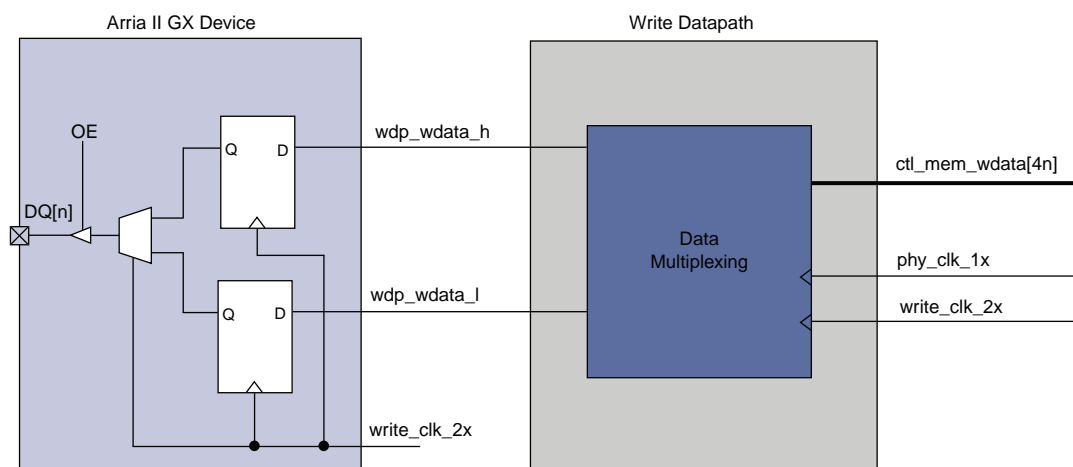
## Write Datapath

The write datapath logic efficiently transfers data from the HDR memory controller to DDR3 SDRAM-based memories. The write datapath logic consists of:

- DQ and DQ output-enable logic
- DQS/DQSn and DQS/DQSn output-enable logic
- DM logic

The memory controller interface outputs  $4n$ -bit wide data (`ctl_wdata[4n]`) at half-rate frequency. Figure 5-4 shows that the HDR write data (`ctl_wdata[4n]`) is clocked by the half-rate clock `phy_clk_1x` (`ctl_clk`) and is converted into SDR, which is represented by `wdp_wdata_h` and `wdp_wdata_l` and clocked by the full-rate clock `write_clk_2x`. The DQ IOEs convert  $2n$  SDR bits to  $n$ -DDR bits.

**Figure 5-8.** DDR3 SDRAM Write Datapath in Arria II GX Devices



## Address and Command Datapath

The address and command datapath is the same as half-rate DDR2 SDRAM address and command datapath ([“Address and Command Datapath”](#) on page 5-13).



## Introduction

The following sections describe the ALTMEMPHY megafunction support for Cyclone® III devices. Cyclone III devices support half-rate and full-rate DDR2/DDR SDRAM interfaces using ALTMEMPHY megafunction.

## DDR2/DDR SDRAM

The following sections discuss half-rate and full-rate DDR2/DDR SDRAM for in Cyclone III devices

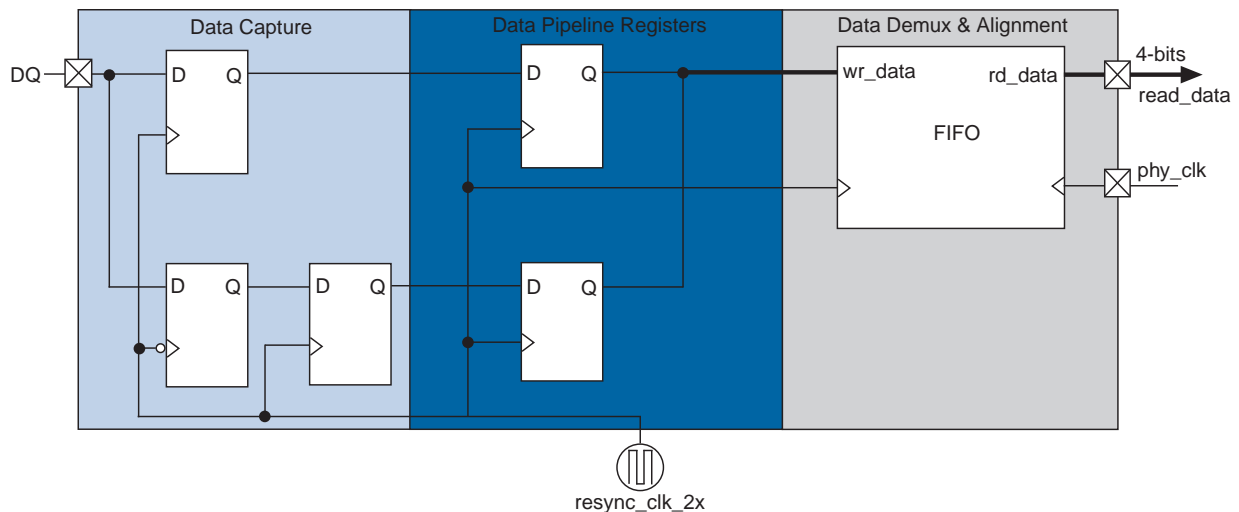
## Half-Rate Support

The following section discusses half-rate support for DDR2/DDR SDRAM interfaces in Cyclone III devices.

### Read Datapath

Figure 6–1 shows the Cyclone III read datapath for a single DQ pin. The diagram shows a half-rate read path where four data bits are produced for each DQ pin. Unlike Stratix® II and Stratix III devices, data capture is entirely done in the core logic because the I/O element (IOE) does not contain DDIO capture registers (nonDQS capture).

**Figure 6–1.** Cyclone III Read Datapath



### Capture and Pipelining

The DDR and DDR2 SDRAM read data is captured using registers in the Cyclone III FPGA core. These capture registers are clocked using the capture clock (`resynch_clk_2x`). The captured read data generates two data bits per DQ pin; one data bit for the read data captured by the rising edge of the capture clock and one data bit for the read data captured by the falling edge of the capture clock.

After the read data has been captured, it may be necessary to insert registers in the read datapath between the capture registers and the read data FIFO to help meet timing. These registers are known as pipeline registers and are clocked off the same clock used by the capture registers, the capture clock (`resynch_clk_2x`).

### Data Demultiplexing

The data demultiplexing for Cyclone III devices is instantiated in the same way as it is with Stratix II devices.

### Postamble Protection

Postamble protection circuitry is not required in the Cyclone III device implementation as DQS mode capture of the DQ data is not supported. The data capture is done using the clock (`resynch_clk_2x`) generated from the ALTPLL megafunction.

### Clock and Reset Management

Clock management circuitry is implemented using the ALTPLL megafunction.

### PLL

The ALTPLL megafunction is instantiated within the ALTMEMPHY megafunction and is responsible for generating all the clocks used by the ALTMEMPHY megafunction and the memory controller.

The minimum PHY requirement is to have 48 phases of the highest frequency clock. The PLL uses Normal mode, unlike other device families. Cyclone III PLL in normal mode emits low jitter already such that you do not require to set the PLL in **With no compensation** operation mode. Changing the PLL compensation mode may result in inaccurate timing results.

You must choose a PLL and PLL input clock pin that are located on the same side of the memory interface to ensure minimal jitter. Cascaded PLLs are not recommended as jitter can accumulate with the use of cascaded PLLs causing the memory output clock to violate the memory device jitter specification. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset and relock the PLL to ensure that the phase relationship between all PLL outputs are properly set.

Table 6-1 lists the clocks generated by the ALTPLL megafunction.

**Table 6-1.** DDR2/DDR SDRAM Clocking in Cyclone III Devices (Part 1 of 2) (Part 1 of 2)

Design Rate	Clock Name	Post-Scale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate	phy_clk_1x and aux_half_rate_clk	C0	0°	Half-Rate	Global	The only half-rate clock that is made available on the user interface of the ALTMEMPHY megafunction to be used by the controller. This clock is not used in full-rate controllers. This clock also feeds into a divider circuit to provide the PLL scan_clk signal for reconfiguration.
	mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	Generates DQS signals and the memory clock and to clock the PHY in full-rate mode.
Full rate	aux_half_rate_clk	C0	0°	Half-Rate	Global	The only half-rate clock that is made available on the user interface of the ALTMEMPHY megafunction to be used by the controller. This clock is not used in full-rate controllers. This clock also feeds into a divider circuit to provide the PLL scan_clk signal for reconfiguration.
	phy_clk_1x and mem_clk_2x and aux_full_rate_clk	C1	0°	Full-Rate	Global	Generates DQS signals and the memory clock and to clock the PHY in full-rate mode.
Half-rate and full rate	write_clk_2x	C2	-90°	Full-Rate	Global	Clocks the data (DQ) when you perform a write to the memory.
Half-rate and full rate	resynch_clk_2x	C3	Calibrated	Full-Rate	Global	A full-rate clock that captures and resynchronizes the captured read data. The capture and resynchronization clock has a variable phase that is controlled via the PLL reconfiguration logic by the control sequencer block.

**Table 6-1.** DDR2/DDR SDRAM Clocking in Cyclone III Devices (Part 2 of 2) (Part 2 of 2)

Design Rate	Clock Name	Post-Scale Counter	Phase (Degrees)	Clock Rate	Clock Network Type	Notes
Half-rate and full rate	measure_clk_2x	C4	Calibrated	Full-Rate	Global	This clock is for VT tracking. This free-running clock measures relative phase shifts between the internal clock(s) and those being fed back through a mimic path. As a result, you can track VT effects on the FPGA and compensate for them.
Half-rate and full rate	ac_clk_2x	—	0°, 90°, 180°, 270°	Full-Rate	Global	This clock is derived from mem_clk_2x when you choose 0° or 180° phase shift) or write_clk_2x (when you choose 90° or 270° phase shift), see <a href="#">“Half-Rate Address and Command Clocking”</a> on page 3-24.

### Reset Management

The reset management for Cyclone III devices is instantiated in the same way as it is with Stratix II devices. Refer to [“Reset Management”](#) on page 5-9 for more information.

### Write Datapath

The write datapath for Cyclone III devices is instantiated in the same way as it is with Stratix II devices. Refer to [“Write Datapath”](#) on page 5-10 for more information.

### Address and Command Datapath

The address and command datapath for Cyclone III devices is similar to the Stratix II device. Refer to [“Address and Command Datapath”](#) on page 5-11 for more information.

## Full-Rate Support

The following section discusses full-rate support for DDR2/DDR SDRAM interfaces in Cyclone III devices.

### Read Datapath

The full-rate read datapath for Cyclone III devices is similar to the half-rate Cyclone III implementation, except that the data is read out of the FIFO with a full-rate clock instead of a half-rate clock. Refer to [“Half-Rate Support”](#) on page 6-1 for more information about Cyclone III support.

### Postamble Protection

There is no postamble protection circuitry as the read DQS signal is ignored in this interface.



## Clock and Reset Management

The clock and reset management for Cyclone III devices is similar to half-rate support for Stratix II devices except that the `phy_clk_1x` clock defined in [Table 6-1](#) is now a full-rate clock and is derived from `mem_clk_2x`. The PLL is configured exactly in the same way as in the half-rate designs.



`phy_clk_1x` is now full-rate, despite the “1x” naming convention.

You must choose a PLL and PLL input clock pin that are located on the same side of the memory interface to ensure minimal jitter. Cascaded PLLs are not recommended as jitter can accumulate with the use of cascaded PLLs causing the memory output clock to violate the memory device jitter specification. Also, ensure that the input clock to the PLL is stable before the PLL locks. If not, you must perform a manual PLL reset and relock the PLL to ensure that the phase relationship between all PLL outputs are properly set. The PLL restrictions in half-rate designs also applies to full-rate designs.

## Write Datapath

The write datapath is similar to the half-rate PHY. The IOE block is identical to the half-rate PHY.

## Address and Command Datapath

The address and command datapath is similar to Stratix II full-rate address and command datapath see [“Address and Command Datapath” on page 5-13](#).



## Introduction

The ALTMEMPHY megafunction can be integrated with your own controller. This section describes the interface requirement and the handshake mechanism for efficient read and write transactions.

## Preliminary Steps

Perform the following steps to generate the ALTMEMPHY megafunction:

1. If you are creating your own DDR3/DDR2/DDR SDRAM controller, generate the Altera® High-Performance Controller targeting your chosen Altera and memory devices, as described in the [DDR3 SDRAM High-Performance Controller User Guide](#) and the [DDR and DDR2 SDRAM High-Performance Controller MegaCore Function User Guide](#). If you are creating a controller or a driver for the QDRII+/QDRII SRAM interface, generate the ALTMEMPHY megafunction as described in [“Getting Started” on page 2–1](#) by selecting the desired parameters.
2. Compile and verify the timing. This step is optional; refer to [“Compiling in the Quartus II Software” on page 2–40](#) and [“Analyzing Timing” on page 2–42](#) for details.
3. If targeting a DDR3/DDR2/DDR SDRAM device, simulate the high-performance controller design so you can determine how to drive the PHY signals using your own controller. If you are targeting a QDRII+/QDRII SRAM interface, continue to the next step.
4. Integrate the top-level ALTMEMPHY design with your controller. If you started with the high-performance controller, the PHY variation name is `<controller_name>_phy.v/.vhd`. Details about integrating your controller with Altera’s ALTMEMPHY megafunction are described in the following sections.
5. Compile and simulate the whole interface to ensure that you are driving the PHY properly and that your commands are recognized by the memory device.

## Design Considerations

This section discusses the important considerations for implementing your own controller with the ALTMEMPHY megafunction. This section describes the design considerations for AFI variants; for non-AFI variants, refer to [Appendix A, Non-AFI](#).



Simulating the high-performance controller is useful if you do not know how to drive the PHY signals.

## Clocks and Resets

The ALTMEMPHY megafunction automatically generates a PLL instance, but you must still provide the reference clock input (`pll_ref_clk`) with a clock of the frequency that you specified in the MegaWizard Plug-In. An active-low global reset input is also provided, which you can deassert asynchronously. The clock and reset management logic synchronizes this reset to the appropriate clock domains inside the ALTMEMPHY megafunction.

A clock output (half the memory clock frequency for a half-rate controller; the same as the memory clock for a full-rate controller) is provided and all inputs and outputs of the ALTMEMPHY megafunction are synchronous to this clock. For AFIs, this signal is called `ctl_clk`.



DDR3 SDRAM ALTMEMPHY variants only support half-rate controllers.

There is also an active-low synchronous reset output signal provided, `ctl_reset_n`. This signal is synchronously de-asserted with respect to the `ctl_clk` or `phy_clk` clock domain and it can reset any additional user logic on that clock domain.

## Calibration Process Requirements

When the global `reset_n` is released the ALTMEMPHY handles the initialization and calibration sequence automatically. The sequencer calibrates memory interfaces by issuing reads to multiple ranks of DDR SDRAM (multiple chip select). Timing margins decrease as the number of ranks increases. It is impractical to supply one dedicated resynchronization clock for each rank of memory, as it consumes PLL resources for the relatively small benefit of improved timing margin. When calibration is complete `ctl_cal_success` goes high if successful; `ctl_cal_fail` goes high if calibration fails. Calibration can be repeated by the controller using the `soft_reset_n` signal, which when asserted puts the sequencer into a reset state and when released the calibration process begins again.



You can ignore the following two warning and critical warning messages:

```
Warning: Timing Analysis for multiple chip select DDR/DDR2/DDR3-SDRAM
configurations is preliminary (memory interface has a chip select width
of 4)
```

```
Critical Warning: Read Capture and Write timing analyses may not be
valid due to violated timing model assumptions
```

## Other Local Interface Requirements

The memory burst length can be two, four, or eight for DDR SDRAM devices; the memory burst length can be four or eight for DDR2 SDRAM devices; DDR3 SDRAM burst lengths can be set at either four or eight—when using the Altera high-performance controller, only burst length four is supported. For a half-rate controller, the memory clock runs twice as fast as the clock provided to the local interface, so data buses on the local interface are four times as wide as the memory data bus. For a full-rate controller, the memory clock runs at the same speed as the clock provided to the local interface, so the data buses on the local interface are two times as wide as the memory data bus.

## High-Performance Controller with AFI

This section describes the DDR, DDR2, or DDR3 SDRAM high-performance controllers with the AFI.

### Address and Command Interfacing

Address and command signals are automatically sized for 1T operation, such that for full-rate designs there is one input bit per pin (for example, one `cs_n` input per chip-select configured); for half-rate designs there are two. If you require a more conservative 2T address and command scheme, use a full-rate design and drive the address/command inputs for two clock cycles, or in a half-rate design drive both address/command bits for a given pin identically.



Although the PHY inherently supports 1T addressing, the high performance controllers support only 2T addressing, so PHY timing analysis is performed assuming 2T address and command signals.

### Handshake Mechanism Between Read Commands and Read Data

When performing a read, a high-performance controller with the AFI asserts `ctl_doing_read` to indicate that a read command is requested and the byte lanes that it expects valid data to return on. ALTMEMPHY uses `ctl_doing_read` for the following actions:

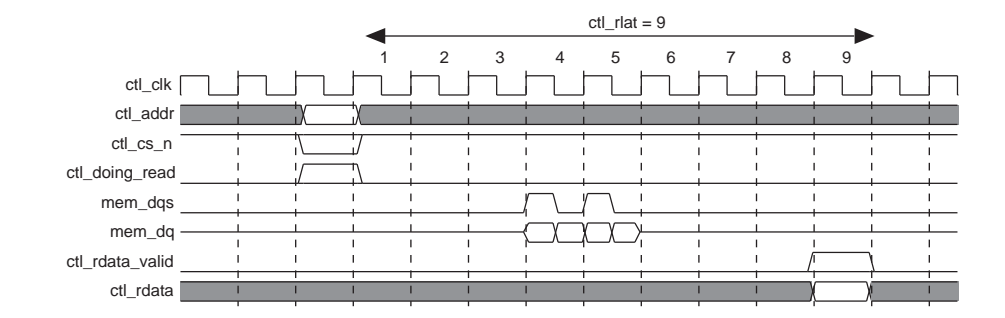
- Control of the postamble circuit
- Generation of `ctl_rdata_valid`
- Dynamic termination ( $R_t$ ) control timing

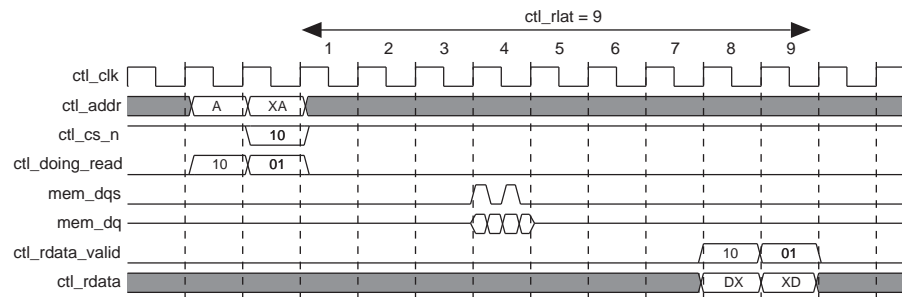
The read latency, `ctl_rlat`, is advertised back to the controller. This signal indicates how long it takes in `ctl_clk` clock cycles from assertion of `ctl_doing_read` to valid read data returning on `ctl_rdata`. The `ctl_rlat` signal is only valid when calibration has successfully completed and never changes values during normal user mode operation.

The ALTMEMPHY provides a signal, `ctl_rdata_valid`, to indicate that the data on read data bus is valid. The width of this signal varies between half-rate and full-rate designs to support the option to indicate that the read data is not word aligned.

Figure 7-1 and Figure 7-2 show these relationships.

**Figure 7-1.** Address and Command and Read-Path Timing—Full-Rate Design



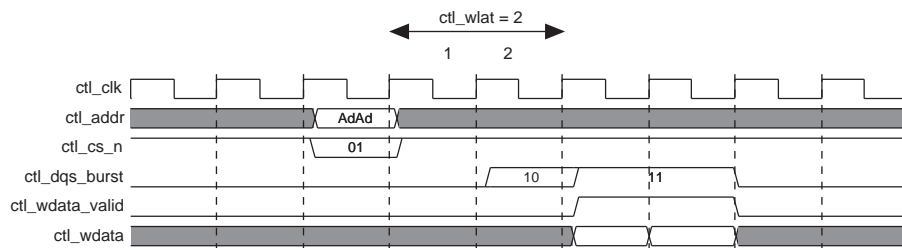
**Figure 7-2.** Second Read Alignment—Half-Rate Design

### Handshake Mechanism Between Write Commands and Write Data

In the AFI, the ALTMEMPHY output `ctl_wlat` gives the number of `ctl_clk` cycles between the write command that is issued `ctl_cs_n` asserted and `ctl_dqs_burst` asserted. The `ctl_wlat` signal takes account of the following actions to provide a single value in `ctl_clk` clock cycles:

- CAS write latency
- Additive latency
- Datapath latencies and relative phases
- Board layout
- Address and command path latency and 1T register setting, which is dynamically setup to take into account any leveling effects

The `ctl_wlat` signal is only valid when the calibration has been successfully completed by the ALTMEMPHY sequencer and does not change at any point during normal user mode operation. Figure 7-3 shows the operation of `ctl_wlat` port.

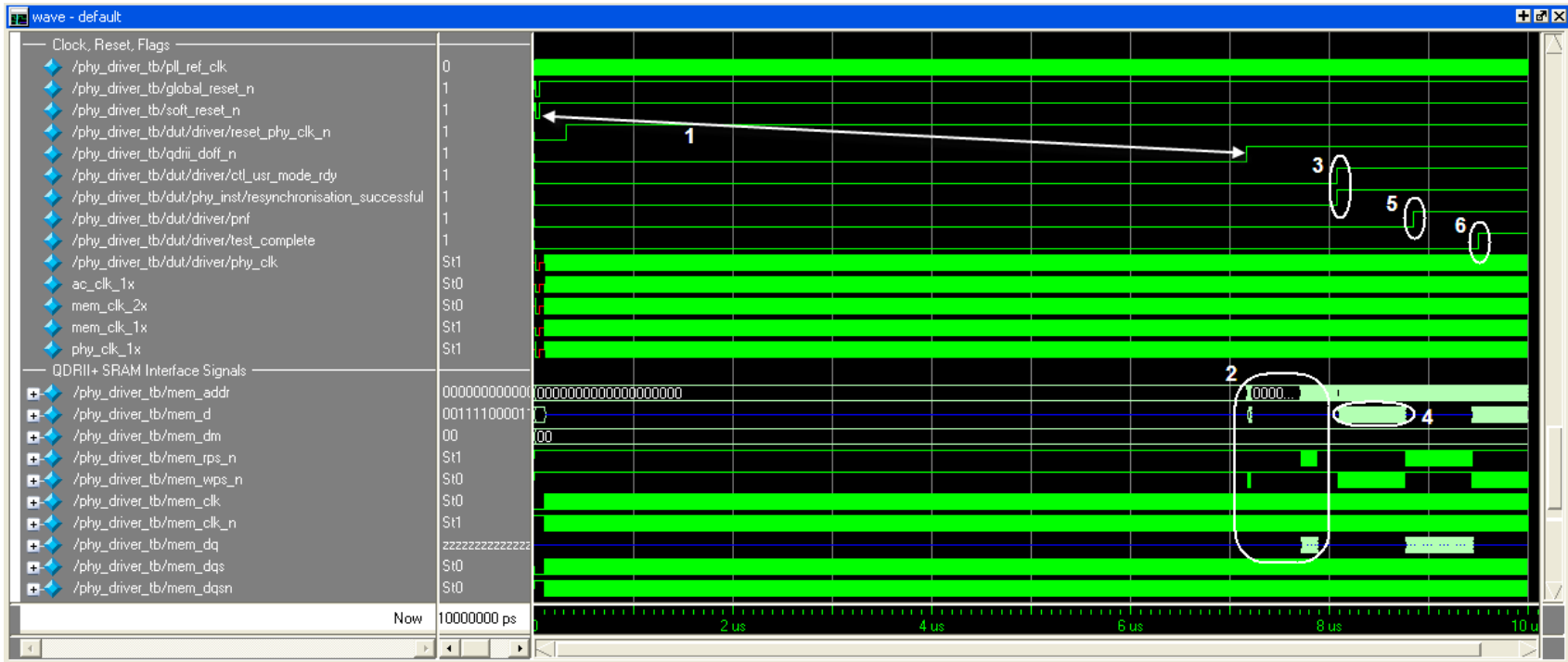
**Figure 7-3.** Timing for `ctl_dqs_burst`, `ctl_wdata_valid`, Address, and Command—Half-Rate Design

For a half-rate design `ctl_cs_n` is 2 bits, not 1. Also the `ctl_dqs_burst` and `ctl_wdata_valid` waveforms indicate a half-rate design. This write results in a burst of 8 at the DDR. Where `ctl_cs_n` is driven 2'b01, the LSB (1) is the first value driven out of `mem_cs_n`, and the MSB (0) follows on the next `mem_clk`. Similarly, for `ctl_dqs_burst`, the LSB is driven out of `mem_dqs` first (0), then a 1 follows on the next clock cycle. This sequence produces the continuous DQS pulse as required. Finally, the `ctl_addr` bus is twice `MEM_IF_ADDR_WIDTH` bits wide and so the address is concatenated to result in an address phase two `mem_clk` cycles wide.

## QDRII+/QDRII SDRAM

Figure 7-4 shows an overview of a QDRII+ SRAM simulation where the driver writes a block of data and compares the read data back to the FPGA.

Figure 7-4. QDRII+ SRAM Interface Simulation Example



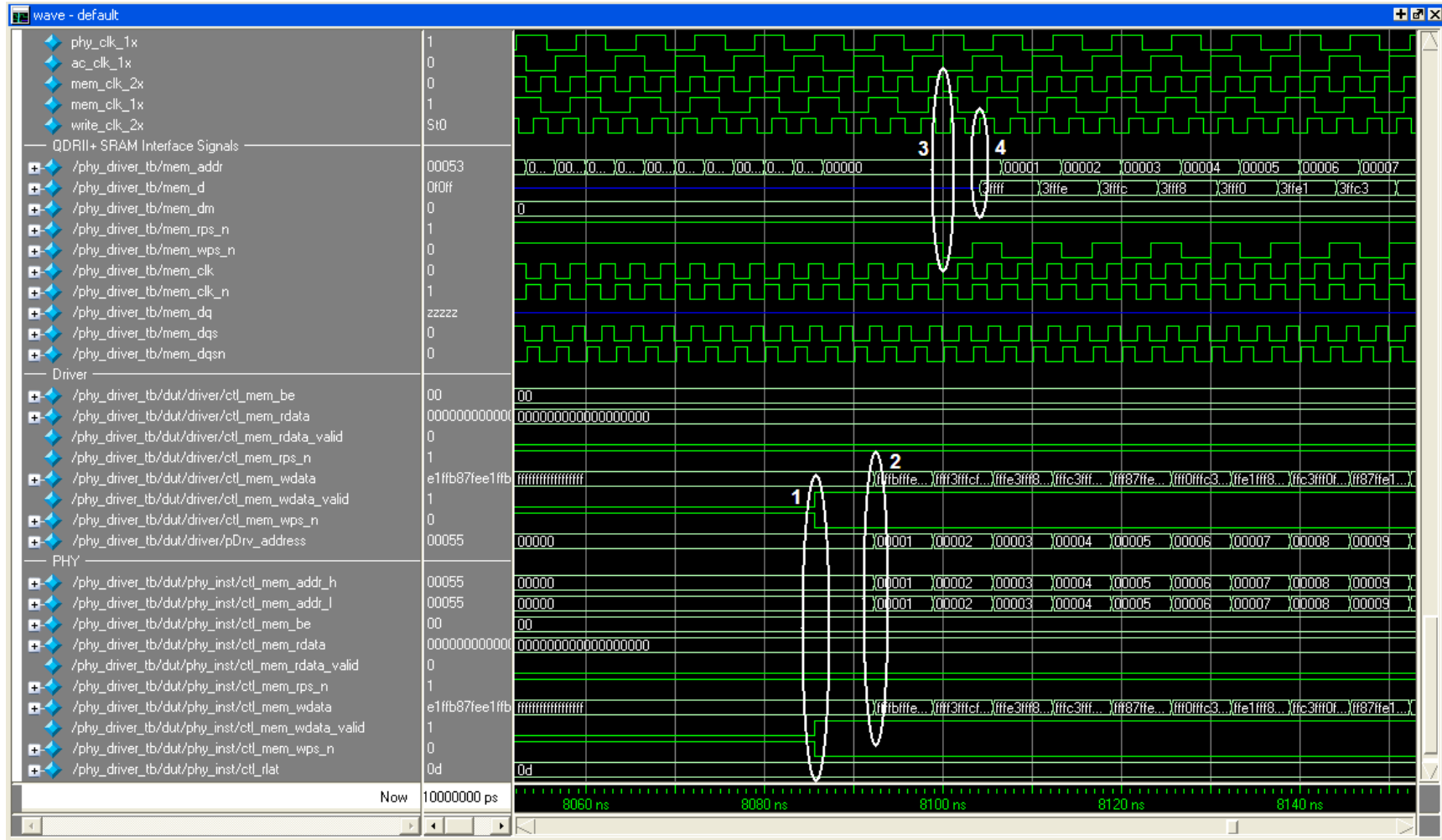
The waveform in [Figure 7-4](#) shows the following operation:

1. The PHY waits for the DLL to lock and then de-asserts the reset signals to start initializing the QDRII+ SRAM DLL. It then asserts the `DOFF` signal to the QDRII+ SRAM device to enable the use of the memory DLL.
2. The sequencer then calibrates the device by writing some training pattern and reads it back.
3. The PHY asserts the `resynchronization_successful` signal when calibration is successful and the `ctl_usr_mode_rdy` signal when all the DQS groups in the interface have been calibrated.
4. The driver in this example then writes a block of data.
5. The driver in this example compares the read data and asserts the `pnf` signal when the read data matches the previously written data.
6. When the test is complete, the driver in this example asserts the `test_complete` signal.



Figure 7-5 shows a QDRII+ SRAM write operation.

Figure 7-5. QDRII+ SRAM Write Example

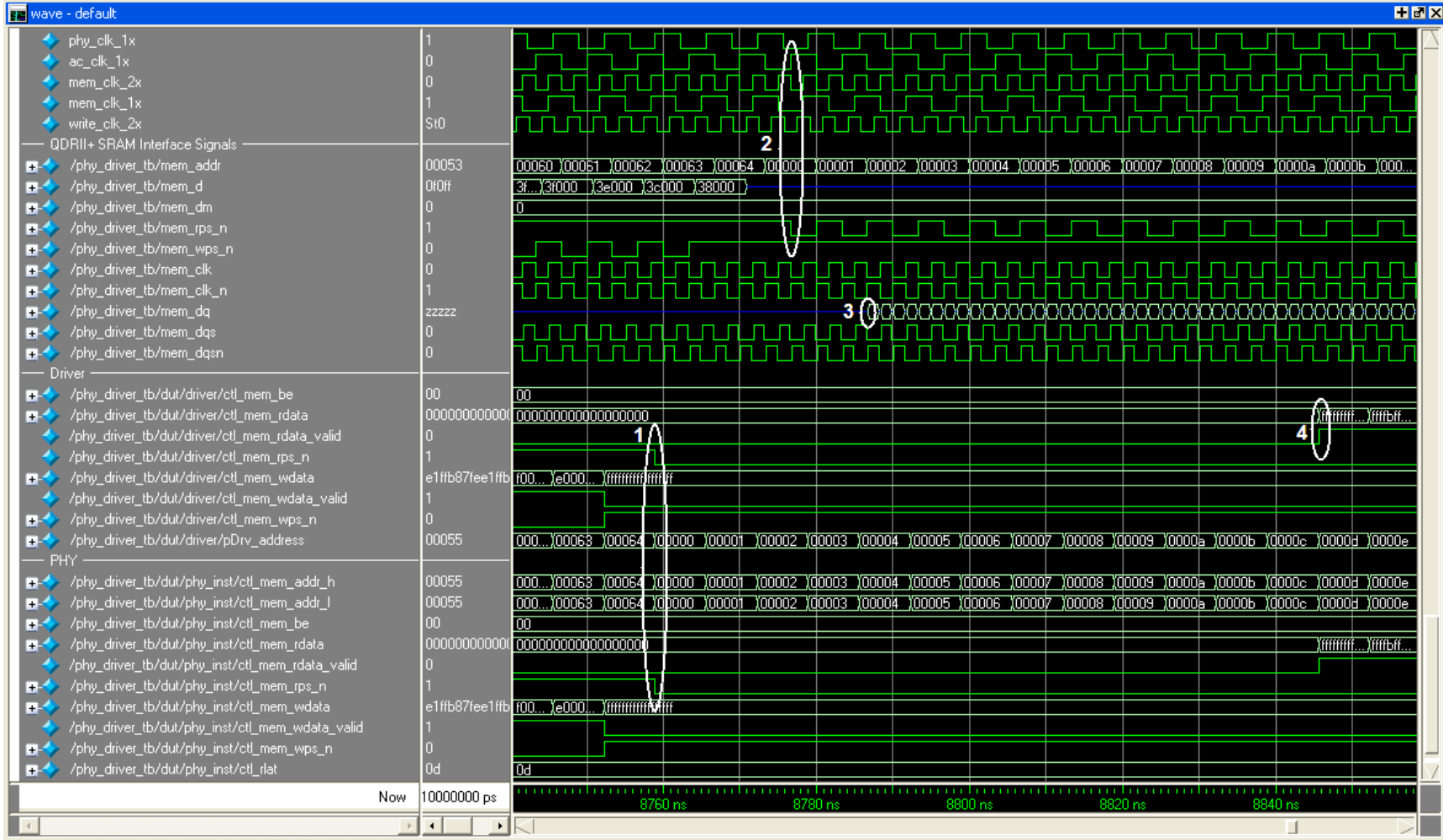


The waveform in [Figure 7-5](#) shows the QDRII+ SRAM write operation:

1. The driver asserts the `ctl_mem_wdata_valid` and the `ctl_mem_wps_n` signals to the PHY to indicate a write operation.
2. On the next `phy_clk_1x` clock cycle, the driver sends the write data and write address to the PHY.
3. The PHY sends the `mem_wps_n` signal and the write address bus 3.66 `phy_clk_1x` clock cycles later, aligned with the falling edge of the `ac_clk_1x` clock.
4. Finally, the PHY sends the write data, aligned with the rising edge of the `write_clk_2x` clock, one and a quarter memory clock cycles after sending the write command and write address bus.

Figure 7-6 shows a QDRII+ SRAM read operation.

Figure 7-6. QDRII+ SRAM Read Example



The waveform in [Figure 7-6](#) shows the QDRII+ SRAM read operation operation:

1. The driver asserts the `ctl1_mem_rps_n` signal along with the read address bus to the PHY to indicate a read operation.
2. The PHY sends the `mem_rps_n` signal and the write address bus two and a quarter memory clock cycles later, aligned with the rising edge of the `ac_clk_1x` clock.
3. The memory device puts the read data on the `mem_dq` bus three memory clock cycles later.
4. The PHY outputs the `ctl1_mem_rdata` signal and `ctl1_mem_rdata_valid` 17.66 memory clock cycles later, aligned with the `phy_clk_1x` clock.

## Introduction

This chapter describes the specifications of the ALTMEMPHY megafunctions that are common to non-AFI variations.

Altera recommends that you use the AFI for new designs; only use the non-AFI for existing designs.

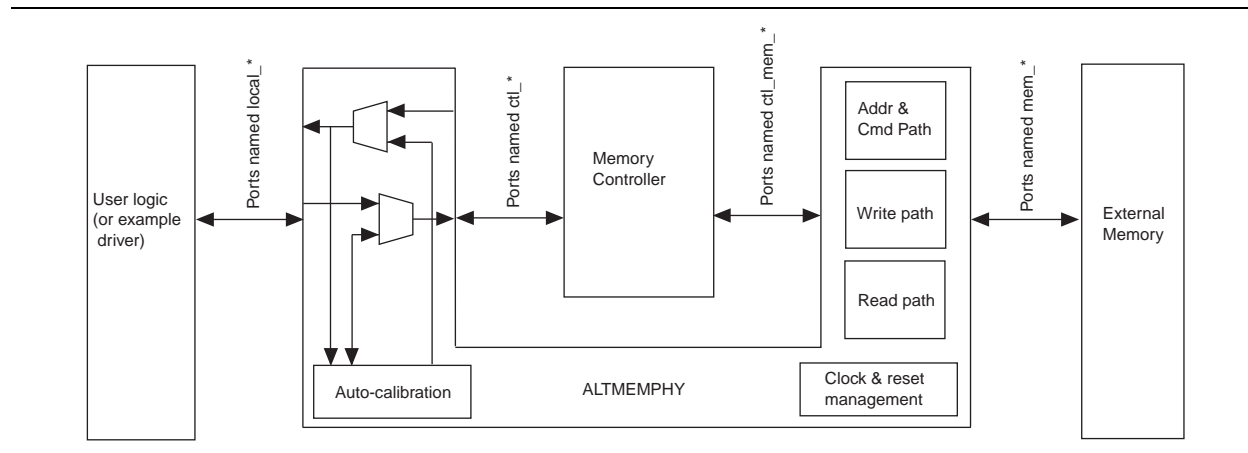
## PHY-to-Controller Interfaces

The following section describes the typical modules that are connected to the ALTMEMPHY variation and the port name prefixes each module uses for non-AFI variations.

The non-AFI's autocalibration logic relies on the services of the memory controller to perform its calibration writes, reads, and memory initialization, so it must have control of the controller's local interface during the initial calibration stage. The ALTMEMPHY megafunction has four interfaces that all must be connected appropriately. [Figure A-1](#) shows the four interfaces.

As an SRAM, the PHY for QDRII+/QDRII SRAM lacks most of the `ctl_` and `local_` ports as you can use a driver that acts as a controller to generate read and write commands and data in the QDRII+/QDRII SRAM PHY.

**Figure A-1.** The Four ALTMEMPHY Megafunction Interfaces



The four ALTMEMPHY interfaces, from left to right, are:

1. The local interface is the interface between the user logic and the memory controller. The signals between user logic and the controller traverse through the ALTMEMPHY megafunction. This can either be an Avalon® Memory-Mapped slave interface or a Native interface. All the ports on this interface have their names prefixed with `local_`; for example, `local_init_done`. During the initial


calibration period, the auto-calibration logic takes control of this interface and issues the write and read requests that the memory controller requires. When the calibration process is complete, control is handed back to the user logic and normal operation occurs. The ALTMEMPHY megafunction auto-calibration logic does not require any further access to the memory controller when the initial autocalibration is complete.

2. The ALTMEMPHY-controller local interface is the interface between the ALTMEMPHY megafunction and the controller local interface. All the port names on this interface are prefixed with `ctl_`; for example, `ctl_init_done`. This interface connects the ALTMEMPHY megafunction to the controller's local interface and is of the same type as the local interface, either an Avalon-MM interface or a native interface. When the calibration process is complete, this connection becomes a straight-through connection and you have complete control of the memory controller.
3. The ALTMEMPHY-controller command interface is the interface between the controller and ALTMEMPHY. All the ports on this interface are prefixed with `ctl_mem_`; for example, `ctl_mem_rdata`. They are clocked by the `phy_clk`. This interface contains the memory control and address signals from the controller to the memory. The controller also sends write data to, and receives read data from, the external memory through this interface. All the signals on this interface are clocked at the `phy_clk` rate. The ALTMEMPHY megafunction converts between this clock and the memory interface clock.
4. The fourth interface is between the ALTMEMPHY megafunction and the external memory devices and consists of the memory address, command, and data pins. These must be connected directly to the external pins of your Altera FPGA.

## Initialization Timing

DDR SDRAM initialization timing is different to DDR2 SDRAM initialization timing.

### DDR SDRAM Initialization Timing


 For DDR2 SDRAM initialization timing, see [“DDR2 SDRAM Initialization Timing”](#) on page A-4.

The DDR SDRAM high-performance controller initializes the SDRAM devices by issuing the following memory command sequence:

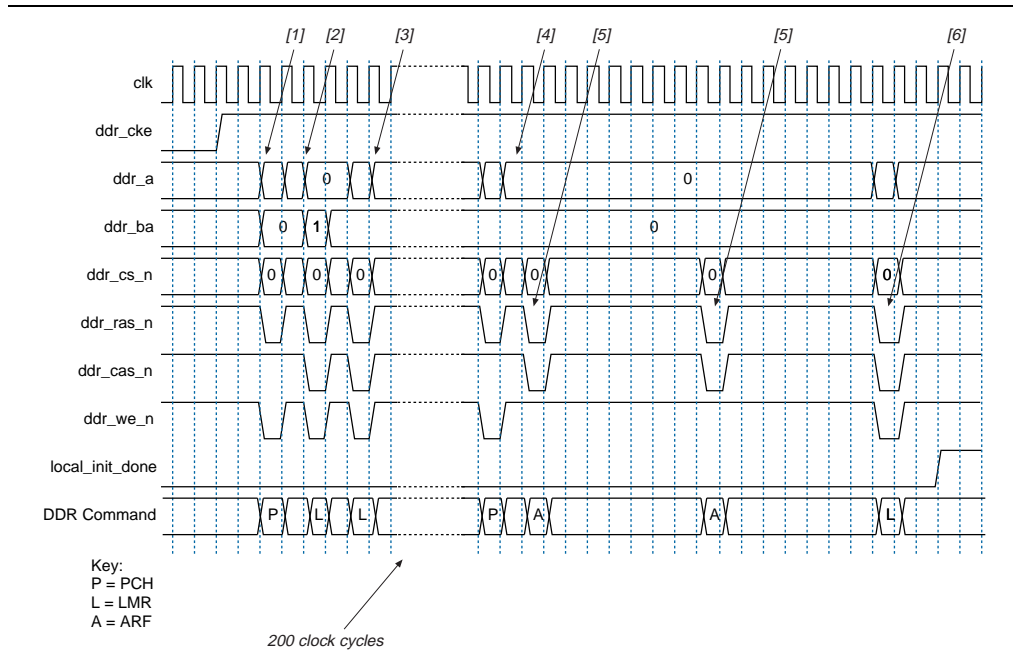
- NOP (for 200  $\mu$ s, programmable)
- PCH
- Extended LMR (ELMR)
- LMR
- NOP (for 200 clock cycles, fixed)
- PCH
- ARF
- ARF

■ LMR

Figure A-2 on page A-3 shows a typical initialization timing sequence. The length of time between the reset and the first PCH command should be 200  $\mu$ s. The value that you specify for the **Memory initialization time at power up (tINIT)** setting in the MegaWizard interface is only used for hardware that you generate. The controller simulation model is created with a much shorter  $t_{INIT}$  time to make simulation easier.

 Do not set tINIT to zero.

**Figure A-2.** DDR SDRAM Device Initialization Timing



The following sequence corresponds with the numbered items in Figure A-2.

1. A PCH command is sent to all banks by setting the precharge pin, the address bit a[10], or a[8] high.
2. An ELMR command is issued to enable the internal delay-locked loop (DLL) in the memory devices. An ELMR command is an LMR command with the bank address bits set to address the extended mode register.
3. An LMR command sets the operating parameters of the memory such as CAS latency and burst length. This LMR command also resets the internal memory device DLL. The DDR SDRAM high-performance controller allows 200 clock cycles to elapse after a DLL reset and before it issues the next command to the memory.
4. A further PCH command places all the banks in their idle state.
5. Two ARF commands must follow the PCH command.
6. The final LMR command programs the operating parameters without resetting the DLL.

After issuing the final LMR command, the memory controller hands over control of the memory to the ALTMEMPHY megafunction to allow it to carry out its calibration process.

When the ALTMEMPHY megafunction has finished calibrating, the memory controller asserts the `local_init_done` signal, which shows that it has initialized the memory devices.

### DDR2 SDRAM Initialization Timing

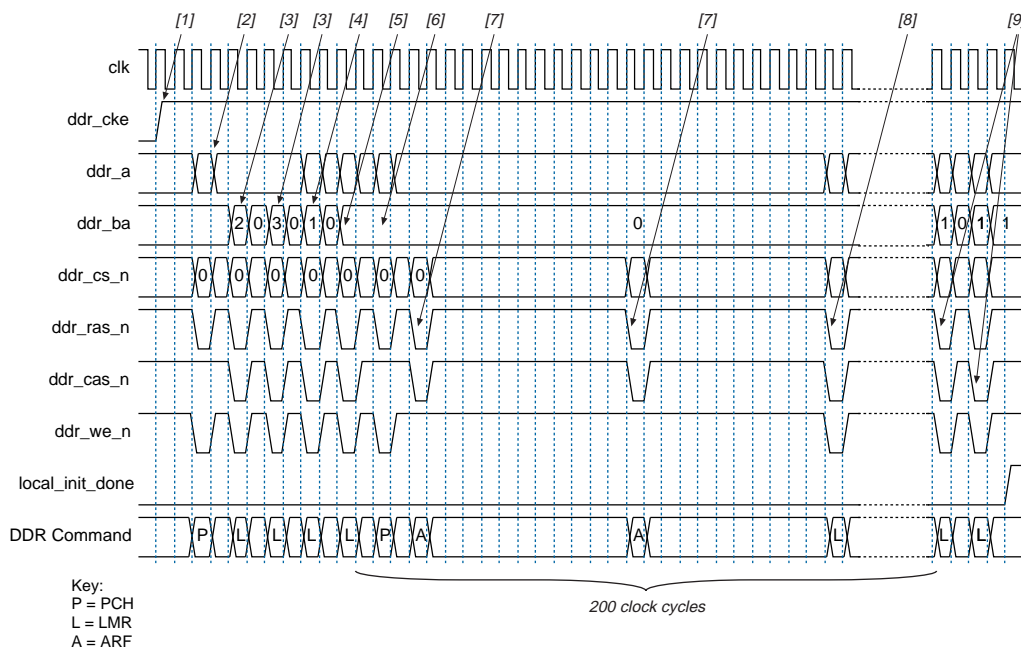
The DDR2 SDRAM high-performance controller initializes the memory devices by issuing the following command sequence:

- NOP (for 200  $\mu$ s, programmable)
- PCH
- ELMR, register 2
- ELMR, register 3
- ELMR, register 1
- LMR
- PCH
- ARF
- ARF
- LMR
- ELMR, register 1
- ELMR, register 1



Figure A-3 shows a typical DDR2 SDRAM initialization timing sequence, which is described below. The length of time between the reset and the clock enable signal going high should be 200  $\mu$ s. The value that you choose for the **Memory initialization time at power up ( $t_{INIT}$ )** setting in the MegaWizard interface is only used for hardware that you generate. The controller simulation model is created with a much shorter  $t_{INIT}$  time to make simulation easier.

**Figure A-3.** DDR2 SDRAM Device Initialization Timing



**Note to Figure A-3:**

(1) `local_init_done` only goes high when calibration has completed.

The following sequence corresponds with the numbered items in [Figure A-3 on page A-5](#).

1. The clock enable signal (CKE) is asserted 200  $\mu$ s after coming out of reset.
2. The controller then waits 400 ns and then issues the first PCH command by setting the precharge pin, the address bit a[10] or a[8] high. The 400 ns is calculated by taking the number of clock cycles calculated by the wizard for the 200  $\mu$ s delay and dividing this by 500. If a small initialization time is selected for simulation purposes, this delay is always at least 1 clock cycle.
3. Two ELMR commands are issued to load extend mode registers 2 and 3 with zeros.
4. An ELMR command is issued to extend mode register 1 to enable the internal DLL in the memory devices.
5. An LMR command is issued to set the operating parameters of the memory such as CAS latency and burst length. This LMR command is also used to reset the internal memory device DLL.
6. A further PCH command places all the banks in their idle state.
7. Two ARF commands must follow the PCH command.

8. A final LMR command is issued to program the operating parameters without resetting the DLL.
9. 200 clock cycles after step 5, two ELMR commands are issued to set the memory device off-chip driver (OCD) impedance to the default setting.

After issuing the final ELMR command, the memory controller hands over control of the memory to the ALTMEMPHY megafunction to allow it to carry out its calibration process.


When ALTMEMPHY megafunction has finished calibrating, the memory controller asserts the `local_init_done` signal, which shows that it has initialized the memory devices.

## Ports


This section describes the ALTMEMPHY megafunction ports for non-AFI variations.

Table A-1 through Table A-10 show the ports for non-AFIs. The port lists include the following signal groups:

- I/O interface to the external memory device
- Clock and reset signals
- PLL reconfiguration signals
- External DLL signals
- User-mode calibrated on-chip termination (OCT) control signals
- Interface to the memory controller
- Local interface signals
- Datapath interface for the controller
- ALTMEMPHY megafunction calibration status interface
- Additional calibration signals from the sequencer

 Ports with the prefix “mem\_” connect the PHY with the memory device; ports with the prefix “ctl\_” connect the PHY with the controller. Ports with prefix “ctl\_mem\_” indicate the datapath for the controller; ports with the prefix “local\_” indicate the signal to be connected with the example driver or user logic.

 Signals with suffix `_n` are active low; signals without suffix `_n` are active high.

 For more information about the simulation waveforms of a DDR2 SDRAM high-performance controller, refer to the figures in the Interface and Signals section and Appendix B in the *DDR and DDR2 SDRAM High-Performance Controller User Guide*.

**Table A-1.** I/O Interface for DDR2 and DDR SDRAM—non-AFI (Note 1) (Part 1 of 2)

Signal Name	Type	Width	Description
mem_addr	output	MEM_IF_ROWADDR_WIDTH	The memory row and column address bus.
mem_ba	output	MEM_IF_BANKADDR_WIDTH	The memory bank address bus.

**Table A-1.** I/O Interface for DDR2 and DDR SDRAM—non-AFI (Note 1) (Part 2 of 2)

Signal Name	Type	Width	Description
mem_cas_n	output	1	The memory column address strobe.
mem_cke	output	MEM_IF_CS_WIDTH	The memory clock enable.
mem_clk	bidir	MEM_IF_CLK_PAIR_COUNT	The memory clock, positive edge clock. (2)
mem_clk_n	bidir	MEM_IF_CLK_PAIR_COUNT	The memory clock, negative edge clock. (2)
mem_cs_n	output	MEM_IF_CS_WIDTH	The memory chip select signal.
mem_dm	output	MEM_IF_DM_WIDTH	The optional memory data mask bus.
mem_dq	bidir	MEM_IF_DWIDTH	The memory bidirectional data bus.
mem_dqs	bidir	MEM_IF_DWIDTH / MEM_IF_DQ_PER_DQS	The memory bidirectional data strobe bus.
mem_dqsn	bidir	MEM_IF_DWIDTH / MEM_IF_DQ_PER_DQS	The memory bidirectional data strobe bus. Not used in Arria GX, Arria II GX, HardCopy II, Stratix II, and Stratix II GX designs.
mem_odt	output	MEM_IF_CS_WIDTH	The memory on-die termination control signal.
mem_ras_n	output	1	The memory row address strobe.
mem_reset_n	output	1	The memory-reset signal.
mem_we_n	output	1	The memory write-enable signal.

**Notes to Table A-1:**

- (1) Connected to I/O pads.
- (2) Output is for memory device, and input path is fed back to ALTMEMPHY megafunction for VT tracking.

**Table A-2.** Clock and Reset Signals for DDR2/DDR SDRAM—non-AFI (Note 1) (Part 1 of 2)

Signal Name	Type	Width	Description
global_reset_n (1)	input	1	The asynchronous reset input to the controller. All other reset signals are derived from resynchronized versions of this. This signal holds the complete ALTMEMPHY megafunction, including the PLL, in reset while low.
soft_reset_n (1)	input	1	The asynchronous reset input to reset controller, for SOPC Builder use, or to be controlled by other system reset logic. This signal causes a complete reset of the PHY, but not the PLL in the PHY. In Arria GX, Arria II GX, Stratix II, and Stratix II GX devices, this signal also resets the PLL reconfiguration block on a falling-edge detection.
phy_clk	output	1	The ALTMEMPHY megafunction half-rate clock provided to the user. All user inputs and outputs to the ALTMEMPHY megafunction are synchronous to this clock in half-rate designs. However, this clock is not used in full-rate designs.
pll_ref_clk	input	1	The reference clock input to PLL.
reset_phy_clk_n (1)	output	1	Asynchronous reset, that is de-asserted synchronously with respect to the associated phy_clock clock domain. Use this to reset any additional user logic on that clock domain.

**Table A-2.** Clock and Reset Signals for DDR2/DDR SDRAM—non-AFI (*Note 1*) (Part 2 of 2)

Signal Name	Type	Width	Description
reset_request_n (1)	output	1	Directly connected to the locked output of the PLL and is intended for optional use either by automated tools such as SOPC Builder or could be manually ANDed with any other system-level signals and combined with any edge detect logic as required and then fed back to the global_reset_n input.  Reset request output that indicates when the PLL outputs are not locked. Use this as a reset request input to any system-level reset controller you may have. This signal is always low while the PLL is locking (but not locked), and so any reset logic using it is advised to detect a reset request on a falling-edge rather than by level detection.
aux_half_rate_clk	output	1	A copy of the phy_clk_1x signal that you can use in other parts of your design, same as phy_clk port.
aux_full_rate_clk	output	1	A copy of the mem_clk_2x signal that you can use in other parts of your design.

**Note to Table A-2:**

- (1) Refer to [Figure 4-3 on page 4-14](#) for the reset mechanism in Stratix III and Stratix IV devices or to [Figure 5-3](#) for the reset mechanism in Arria GX, Arria II GX, Cyclone III, Stratix II and Stratix II GX devices.

The ports in [Table A-3](#) exists for all DDR2/DDR SDRAM variations. However, these ports are unused for Stratix IV and Stratix III devices, hence they can be left unconnected when targeting these devices. Phase shifting in Stratix IV and Stratix III devices use PLL phase shift inputs directly instead of using an ALTPLL\_RECONFIG megafunction.

**Table A-3.** PLL Reconfiguration Signals for DDR2 and DDR SDRAM—non-AFI (Part 1 of 2)

Signal Name	Type	Width	Description
pll_reconfig_enable	input	1	Allows access to the PLL reconfiguration block. Hold this signal low in normal operation. While the ALTMEMPHY is held in reset (via the soft_reset_n signal), and the reset_request_n signal is 1, it is safe to reconfigure the PLL. To reconfigure the PLL, set this signal to 1 and use the other pll_reconfig signals to access the PLL. When finished reconfiguring, set this signal to 0, and then set the soft_reset_n signal to 1 to bring the ALTMEMPHY out of reset. For this signal to work, the PLL_RECONFIG_PORTS_EN parameter must be set to <b>TRUE</b> .
pll_reconfig_write_param	input	1	See the <a href="#">ALTPLL_RECONFIG User Guide</a> for more information.
pll_reconfig_read_param	input	1	See the <a href="#">ALTPLL_RECONFIG User Guide</a> for more information.
pll_reconfig	input	1	See the <a href="#">ALTPLL_RECONFIG User Guide</a> for more information.
pll_reconfig_counter_type	input	4	See the <a href="#">ALTPLL_RECONFIG User Guide</a> for more information.
pll_reconfig_counter_param	input	3	See the <a href="#">ALTPLL_RECONFIG User Guide</a> for more information.

**Table A-3.** PLL Reconfiguration Signals for DDR2 and DDR SDRAM—non-AFI (Part 2 of 2)

Signal Name	Type	Width	Description
pll_reconfig_data_in	input	9	See the <i>ALTPLL_RECONFIG User Guide</i> for more information.
pll_reconfig_soft_reset_en_n	input	1	The asynchronous reset input to the PLL reconfiguration block. This reset causes a PLL reconfiguration block reset and holds the reset if the ALTMEMPHY megafunction is in reset while the signal is low. This port only exists in the DDR2/DDR SDRAM variation for Arria GX, Arria II GX, Stratix II, and Stratix II GX devices.
pll_reconfig_busy	output	1	See the <i>ALTPLL_RECONFIG User Guide</i> for more information.
pll_reconfig_data_out	output	9	See the <i>ALTPLL_RECONFIG User Guide</i> for more information.
pll_reconfig_clk	output	1	Synchronous clock to use for any logic accessing the PLL reconfiguration interface.
pll_reconfig_reset	output	1	Resynchronized reset to use for any logic accessing the PLL reconfiguration interface.

The ports listed in [Table A-4](#) are not applicable in Cyclone III devices, even though they exist when you generate a DDR2/DDR SDRAM PHY variation for these devices. You can leave these signals unconnected. Cyclone III devices do not have a DLL.

**Table A-4.** External DLL Signals for DDR2 and DDR SDRAM—non-AFI

Signal Name	Type	Width	Description
dqs_delay_ctrl_export	output	6	Allows sharing DLL in this ALTMEMPHY instance with another ALTMEMPHY instance. Connect the dqs_delay_ctrl_export port on the ALTMEMPHY instance with a DLL to the dqs_delay_ctrl_import port on the other ALTMEMPHY instance.
dqs_delay_ctrl_import	input	6	Allows the use of DLL in another ALTMEMPHY instance in this ALTMEMPHY instance. Connect the dqs_delay_ctrl_export port on the ALTMEMPHY instance with a DLL to the dqs_delay_ctrl_import port on the other ALTMEMPHY instance.
dll_reference_clk	output	1	Reference clock to feed to an externally instantiated DLL. This clock is typically from one of the PHY PLL outputs.

The ports listed in [Table A-5](#) only exist when you target Stratix III and Stratix IV devices. You can leave them unconnected if you are not using user-mode calibrated OCT.

**Table A-5.** User-Mode Calibrated OCT Control Signals for DDR2/DDR SDRAM—non-AFI (Note 1), (2)

Signal Name	Type	Width	Description
oct_ctl_rs_value	input	14	Specifies serial termination value. Connects to the seriesterminationcontrol bus of the ALT_OCT megafunction. This port exists when you target Stratix IV and Stratix III devices only.
oct_ctl_rt_value	input	14	Specifies parallel termination value. Connects to the parallelerminationcontrol bus of the ALT_OCT megafunction. This port exists when you target Stratix IV and Stratix III devices only.

**Notes to Table A-5:**

- (1) These ports are available if you want to use user-mode OCT calibration. Otherwise, they can be left unconnected.
- (2) For more information on OCT, see the ALT\_OCT Megafunction User Guide.

**Table A-6.** Interface to the Memory Controller for DDR2 and DDR SDRAM—non-AFI (Note 1) (Part 1 of 3)

Signal Name	Type	Width	Description
ctl_add_1t_ac_lat	input	1	<p>When asserted, one extra address and command clock cycle (1T) of latency is inserted in the address and command path if the ADDR_CMD_ADD_1T parameter is set to <b>EXT_SELECT</b>, see “Handshake Mechanism Between Write Commands and Write Data” on page A-42.</p> <p>For DDR SDRAM, the write latency is fixed at one memory clock cycle, but for DDR2 SDRAM, this value changes with the read CAS latency. As the controller is running at half the rate of the memory clock, a latency change of one controller clock cycle is two memory clock cycles. The ALTMEMPHY megafunction allows you to dynamically insert an extra memory clock of delay in the address and command path to compensate.</p> <p>The insertion of delay is controlled by the ADDR_CMD_ADD_1T parameter and the ctl_add_1t_ac_lat signal. If ADDR_CMD_ADD_1T is set to the string <b>EXT_SELECT</b>, an extra cycle of latency can be dynamically inserted on the address and command outputs by asserting the ctl_add_1t_ac_lat input, which allows run-time control of the address and command latency. If ADDR_CMD_ADD_1T is set to the string value <b>TRUE</b>, the extra clock cycle of latency is always present. If it is set to the string value <b>FALSE</b>, the extra latency is never added.</p>
ctl_add_intermediate_regs	Input	1	<p>When asserted, an additional intermediate register or registers is included in the address and command path if the ADDR_CMD_ADD_INTERMEDIATE_REGS parameter is set to <b>EXT_SELECT</b>.</p> <p>For Stratix II and Cyclone III devices only, to maintain the clock cycle relationship between address/command and the write data. You must include the address/command phases where required.</p>
ctl_address	output	LOCAL_IF_AWIDTH	<p>The address corresponding to a write or read request. The ALTMEMPHY sequencer logic drives ctl_address during calibration. ctl_address has the same timing as local_address.</p>
ctl_be	output	LOCAL_IF_DWIDTH/8	<p>The output to the controller indicating the byte-enable flags. The ALTMEMPHY sequencer logic drives ctl_be during calibration. ctl_be is mandatory and has the same timing as local_be.</p>

**Table A-6.** Interface to the Memory Controller for DDR2 and DDR SDRAM—non-AFI (*Note 1*) (Part 2 of 3)

Signal Name	Type	Width	Description
ctl_doing_rd	input	1	The active-high signal from the controller specifying that a read command has been issued to the external RAM.  For more information, see “ <a href="#">Handshake Mechanism Between Read Commands and Read Data</a> ” on page A-40.
ctl_init_done	input	1	The memory controller drives this active-high signal to specify that the controller has initialized the memory and the calibration process should begin.
ctl_negedge_en	input	1	This signal is used if ADDR_CMD_NEGEDGE_EN is set to <b>EXT_SELECT</b> . If true, the address and command signals are output on the falling edge of the address and command clock, ac_clk_2x. If false, the address and command signals are output on the rising edge of the address and command clock. When set to <b>EXT_SELECT</b> , the ctl_negedge_en top level input determines whether the edge is used.
ctl_read_req	output	1	The active-high signal requesting a read command to the address on the ctl_address bus.
ctl_ready	input	1	The controller-ready signal which indicates that the currently asserted read or write request has been accepted. The address of the request is sampled when both the ready and request signals are high.
ctl_size	output	LOCAL_BURST_LEN_BITS	The output to the controller indicating the size (length) of the burst transfer, fixed at 1 for this version.
ctl_usr_mode_rdy	output	1	The ALTMEMPHY sequencer logic drives this active-high signal to specify the ALTMEMPHY has finished its calibration and is ready to accept user read or write requests.
ctl_wdata	output	LOCAL_IF_DWIDTH	The write data from the ALTMEMPHY to the controller. The ALTMEMPHY sequencer logic drives ctl_wdata during calibration. ctl_wdata has the same timing as local_wdata.
ctl_wdata_req	input	1	The controller-request for write data; not required when the controller has an Avalon-MM interface.  The memory controller that the the ALTMEMPHY sequencer uses during calibration drives ctl_wdata_req. Same timing as local_wdata_req.
ctl_write_req	output	1	The active-high signal specifying that a write command should be issued to the address on the ctl_address signal. The ALTMEMPHY sequencer logic drives ctl_write_req during calibration. Same timing as local_write_req.



**Table A-6.** Interface to the Memory Controller for DDR2 and DDR SDRAM—non-AFI (Note 1) (Part 3 of 3)

Signal Name	Type	Width	Description
ctl_refresh_ack	input	1	The active-high valid signal from the controller acknowledging the refresh request. The ALTMEMPHY sequencer logic uses <code>ctl_refresh_ack</code> during calibration.
ctl_refresh_req	output	1	The output to the controller requesting a refresh. Same timing as <code>local_refresh_req</code> .
ctl_burstbegin	output	1	The output to the controller indicating the start of a burst. Only available for the Avalon-MM interface.
ctl_rdata	input	LOCAL_IF_DWIDTH	The read data from the controller. The ALTMEMPHY sequencer logic uses <code>ctl_rdata</code> during calibration. <code>ctl_rdata</code> has the same timing as <code>local_rdata</code> .
ctl_rdata_valid	input	1	The active-high valid signal for the controller read data. Asserted coincident with the read data on <code>ctl_rdata</code> . The controller drives <code>ctl_rdata_valid</code> and has the same timing as <code>local_rdata_valid</code> .
ctl_add_1t_odt_lat	input	1	When asserted, one extra address and command clock cycle (1T) of latency is inserted in the address and command ODT path if <code>ODT_ADD_1T</code> is set to <b>EXT_SELECT</b> , see “Handshake Mechanism Between Write Commands and Write Data” on page A-42.  The timing of the <code>mem_odt</code> signal can be controlled in the same way as <code>mem_addr</code> , but is independent of the address and command latency. If the <code>ODT_ADD_1T</code> parameter is set to <b>EXT_SELECT</b> , an extra cycle of latency can be dynamically inserted on the ODT command outputs by asserting the <code>ctl_add_1t_odt_lat</code> input, which allows separate run-time control of the latency of the ODT signal. If <code>ODT_ADD_1T</code> is set to <b>TRUE</b> , the extra clock cycle of latency is always present. If <code>ODT_ADD_1T</code> is set to <b>FALSE</b> , the extra latency is never added.
ctl_rlat	output	READ_LAT_WIDTH	Unused port that exists when you target Stratix IV and Stratix III devices. The default <code>READ_LAT_WIDTH</code> is set to <b>4</b> .
ctl_self_rfsh_ack	input	1	Signal from the Altera high-performance controller that is passed through the PHY.
ctl_powerdn_ack	input	1	
ctl_autopch_req	output	1	
ctl_powerdn_req	output	1	
ctl_self_rfsh_req	output	1	

**Note to Table A-6:**

(1) Interface signals to the controller either through the sequencer or user interface.

**Table A-7.** Local Interface Signals for DDR2 and DDR SDRAM—non-AFI (Part 1 of 2) *(Note 1)*

Signal Name	Type	Width	Description
local_address	input	LOCAL_IF_ AWIDTH	The address corresponding to a write or read request.
local_be	input	LOCAL_IF_ DWIDTH / 8	The input to ALTMEMPHY megafunction indicating the byte-enable flags.
local_read_req	input	1	The active-high signal requesting a read command to the address on the <code>ctl_address</code> bus.
local_ready	output	1	The controller-ready signal which indicates that the currently asserted read or write request has been accepted. The address of the request is sampled when both the ready and request signals are high.
local_size	input	LOCAL_ BURST_LEN_ BITS	The controller signal to indicate the size (length) of the burst transfer, fixed at 1 for this version.
local_wdata	input	LOCAL_IF_ DWIDTH	The write data from the user to the ALTMEMPHY megafunction.
local_wdata_req	output	1	The controller request for write data; not required when the controller has an Avalon-MM interface.
local_write_req	input	1	The active-high signal specifying that a write command should be issued to the address on the <code>ctl_address</code> signal.
local_refresh_req	input	1	The ALTMEMPHY megafunction receives refresh requests from the local interface and passes them to the controller via <code>ctl_refresh_req</code> when in user mode ( <code>ctl_usr_mode</code> output is high).
local_burstbegin	input	1	The ALTMEMPHY megafunction receives the <code>burstbegin</code> signal from the local interface and passes it to the controller via <code>ctl_burstbegin</code> when in user mode ( <code>ctl_usr_mode</code> output is high).
local_rdata	output	LOCAL_IF_ DWIDTH	When <code>ctl_usr_mode</code> is high, this output passes through the read data from the controller to the local interface. Otherwise, it is tied low.
local_rdata_valid	output	1	When <code>ctl_usr_mode</code> is high, this output passes through the read data valid signal ( <code>ctl_rdata_valid</code> ) from the controller to the local interface. Otherwise, it is driven low.
local_init_done	output	1	When <code>ctl_usr_mode</code> is high, this output passes through the controller's initialization done signal ( <code>ctl_init_done</code> ) from the controller to the local interface. Otherwise, it is driven low.
local_refresh_ack	output	1	When <code>ctl_usr_mode</code> is high, this output passes through the controller's refresh acknowledge signal ( <code>ctl_refresh_ack</code> ) from the controller to the local interface. Otherwise, it is driven low.
local_autopch_req	input	1	User control for auto precharge to request the controller to issue an autoprecharge write or autoprecharge read command.

**Table A-7.** Local Interface Signals for DDR2 and DDR SDRAM—non-AFI (Part 2 of 2) *(Note 1)*

Signal Name	Type	Width	Description
local_powerdn_req	input	1	User control to power down the memory device to request the controller to place the memory devices into a power-down state as soon as it can without violating the relevant timing parameters and responds by asserting the local_powerdn_ack signal.
local_self_rfsh_req	input	1	User control of the self-refresh feature to request that the controller place the memory devices into a self-refresh state by asserting this signal.
local_self_rfsh_ack	output	1	Self-refresh request acknowledge signal. This signal is asserted and deasserted in response to the local_self_rfsh_req signal from the user.
local_powerdn_ack	output	1	Power-down request acknowledge signal. This signal is asserted and deasserted in response to the local_powerdn_req signal from the user.

**Note to Table A-7:**

(1) Passed through PHY to the controller.

**Table A-8.** Datapath Interface for DDR2 and DDR SDRAM—non-AFI (Part 1 of 3) *(Note 1)*

Signal Name	Type	Width	Description
ctl_mem_addr_h <i>(1)</i>	input	MEM_IF_ROWADDR_WIDTH	The row or column address that is sent to the external memory. Output during the high half-period of the address and command clock and driven by the memory controller.
ctl_mem_addr_l <i>(1)</i>	input	MEM_IF_ROWADDR_WIDTH	The row or column address that is sent to the external memory. Output during the low half-period of the address and command clock and driven by the memory controller.
ctl_mem_ba_h <i>(1)</i>	input	MEM_IF_BANKADDR_WIDTH	The bank address that is sent to the external memory. Output during the high half-period of the address and command clock and driven by the memory controller.
ctl_mem_ba_l <i>(1)</i>	input	MEM_IF_BANKADDR_WIDTH	The bank address that is sent to the external memory. Output during the low half-period of the address and command clock and driven by the memory controller.
ctl_mem_be	input	LOCAL_IF_DWIDTH/8	The optional byte-enable signals for the write data to the external memory. The ALTMEMPHY megafunction converts the byte enables into memory mem_dm signals. If mem_dm pins are not required (mem_dm_pins set to <b>FALSE</b> ), the mem_dm logic is not generated and the mem_dm pins are not instantiated.
ctl_mem_cas_n_h <i>(1)</i>	input	1	The column-address strobe signal from the controller to the memory. Output during the high half-period of the address and command clock and driven by the memory controller.

**Table A-8.** Datapath Interface for DDR2 and DDR SDRAM—non-AFI (Part 2 of 3) *(Note 1)*

Signal Name	Type	Width	Description
ctl_mem_cas_n_l (1)	input	1	The column-address strobe signal from the controller to the memory. Output during the low half-period of the address and command clock and driven by the memory controller.
ctl_mem_cke_h (1)	input	MEM_IF_CS_WIDTH	The clock-enable signal from the controller to the memory. Output during the high half-period of the address and command clock and driven by the memory controller.
ctl_mem_cke_l (1)	input	MEM_IF_CS_WIDTH	The clock-enable signal from the controller to the memory. Output during the low half-period of the address and command clock and driven by the memory controller.
ctl_mem_cs_n_h (1)	input	MEM_IF_CS_WIDTH	The chip-select signal from the controller to the memory. For half-rate designs, always tie <code>ctl_mem_cs_n_h</code> high, as even with 2T addressing, the chip select is only driven for the second clock cycle, to allow an extra clock cycle of setup time for the other address and command signals. Output during the high half-period of the address and command clock and driven by the memory controller.
ctl_mem_cs_n_l (1)	input	MEM_IF_CS_WIDTH	The chip-select signal from the controller to the memory. Output during the low half-period of the address and command clock and driven by the memory controller.
ctl_mem_dqs_burst	input	1	Controls the DQS output enables of the DQS pins. See chapter 7, “Integrating the ALTMEMPHY Variation” on page 7-1 for specific details of the timing of this signal.
ctl_mem_odt_h (1)	input	MEM_IF_CS_WIDTH	The on-die termination signal from the controller to the memory.
ctl_mem_odt_l (1)	input	MEM_IF_CS_WIDTH	The on-die termination signal from the controller to the memory.
ctl_mem_ras_n_h (1)	input	1	The row-address strobe signal from the controller to the memory. Output during the high half-period of the address and command clock and driven by the memory controller.
ctl_mem_ras_n_l (1)	input	1	The row-address strobe signal from the controller to the memory. Output during the low half-period of the address and command clock and driven by the memory controller.
ctl_mem_rdata	output	LOCAL_IF_DWIDTH	The <code>ctl_mem_rdata</code> signal is the captured, resynchronized, and demultiplexed read data from the ALTMEMPHY megafunction to the controller.  The <code>ctl_mem_rdata_valid</code> signal indicates when the <code>ctl_mem_rdata</code> is valid.  For more information, see “Handshake Mechanism Between Read Commands and Read Data” on page A-40.
ctl_mem_rdata_valid	output	1	

**Table A-8.** Datapath Interface for DDR2 and DDR SDRAM—non-AFI (Part 3 of 3) (Note 1)

Signal Name	Type	Width	Description
ctl_mem_wdata	input	MEM_IF_DWIDTH× DWIDTH_ RATIO	The write data bus, which has valid data in the same clock cycles that control_wdata_valid is asserted, see “Handshake Mechanism Between Read Commands and Read Data” on page A-40.
ctl_mem_wdata_valid	input	1	Generates the mem_dq output enable. For specific details of the timing of this signal, see “Integrating the ALTMEMPHY Variation” on page 7-1. When asserted, the ctl_mem_rdata_valid signal indicates that the coincident read data on ctl_mem_rdata is valid.
ctl_mem_we_n_h (1)	input	1	The write-enable signal from the controller to the memory. Output during the high half-period of the address and command clock and driven by the memory controller.
ctl_mem_we_n_l (1)	input	1	The write-enable signal from the controller to the memory. Output during the low half-period of the address and command clock and driven by the memory controller.

**Note to Table A-8:**

- (1) The “\_h” and “\_l” stand for high and low. They signify in which half of the clock cycle the data is output. The \_h data is output when the corresponding clock, for example ac\_clk\_2x, is high. The \_l data is output when the ac\_clk\_2x clock is low. The signals with \_h and \_l allow you to select between 1T and 2T addressing. For half-rate designs, 1T is where the address and command signals are driven for one clock; 2T is where they are driven for 2 clocks. For full-rate designs, ensure the same signal drives both \_h and \_l signals and 2T addressing is used. Also when high-performance controllers use ALTMEMPHY, 2T addressing is used.

**Table A-9.** Calibration Status Interface for DDR2 and DDR SDRAM—non-AFI

Signal Name	Type	Width	Description
resynchronisation_successful	output	1	Active-high signal that is set to indicate that calibration of the read data resynchronization clock phase was completed and successful.
postamble_successful	output	1	Active-high signal that is set to indicate that read postamble calibration was completed.
tracking_successful	output	1	Active-high signal that is set to indicate the completion of mimic path VT variation tracking operation.
tracking_adjustment_up	output	1	Active-high signal that is pulsed to indicate that the mimic path tracking has adjusted the resynchronization clock phasing upwards.
tracking_adjustment_down	output	1	Active-high signal that is pulsed to indicate that the mimic path tracking has adjusted the resynchronization clock phasing downwards.

Table A-10 shows additional calibration status signal outputs from the sequencer. You can either pull these signals out to the top level or observe them through the SignalTap® II logic analyzer.

**Table A-10.** Additional Calibration Status Signals—non-AFI

Signal	Description
<code>rsu_codvw_phase</code> (Center of data valid window)	The ALTMEMPHY resynchronization setup unit (RSU) sweeps the phase of a resynchronization clock across 360° (full-rate designs) or 720° (half-rate designs) of a memory clock cycle. Data reads from the DIMM are performed for each phase position and a data valid window is located, which is the set of resynchronization clock phase positions where data is successfully read. The final resynchronization clock phase is set at the center of this range: the center of the data valid window (CODVW). This output is set to the current calculated value for the CODVW and represents how many phase steps were performed by the PLL to offset the resynchronization clock from the memory clock.
<code>rsu_read_latency</code> (Read latency at the center of the window)	If the RSU can find one data valid window (and not more than one), the resynchronization clock is positioned at the center and the <code>rsu_read_latency</code> output is then set to the read latency (in <code>phy_clk</code> cycles) using that resynchronization clock phase. If calibration is unsuccessful, this signal remains at 0.
<code>rsu_no_dvw_err</code> (Calibration failed due to no window found)	If the RSU sweeps the resynchronization clock across every phase and does not see any valid data at any phase position, calibration fails and this output is set to 1.
<code>rsu_grt_one_dvw_err</code> (Calibration failed due to more than one valid window)	If the RSU sweeps the resynchronization clock across every phase and sees multiple data valid windows, this indicates unexpected read data (random bit errors) or an incorrectly configured PLL, which must be resolved. Calibration has failed and this output is set to 1.
<code>rsu_multiple_valid_latencies_err</code> (Calibration failed due to more than two read latencies)	If the RSU sweeps the resynchronization clock across every phase and sees valid data at more than two different latencies, calibration fails and this output is set to 1.

## Understanding the Testbench


Before the user logic (example driver) can read or write to the local interface, the external SDRAM must first be initialized and calibrated. Following power-up or a reset event, the following stages of operation take place:


- PLL initialization and lock
- Memory device initialization
- Interface training and calibration
  - Write training data
  - Calibration
- Functional memory use

### PLL Initialization and Lock

PLL initialization and lock is the first activity that takes place and completes when the signal `p11_locked` is first asserted. Typically this stage requires approximately 150 ns, but can take longer if the **PLL Option Hold 'locked' output low for <user entered number> cycles after the PLL initializes** is turned on.

The exact length of time required for `p11_locked` to become asserted depends on several factors including **Device Type**, **PLL Type**, and **PLL Configuration**.

 `p11_locked` is not included in the simulation default waveform view, and must be added manually.

 For more information refer to *ALTPLL Megafunction User Guide*.

## Memory Device Initialization

Memory devices must be initialized before functional use. The exact sequence is different for DDR2 and DDR SDRAM. The memory controller sets the operating parameters of the memory based on the parameters you specify in the MegaWizard interface. This parameter is fixed at generation time and is not dynamically editable via the local interface.

## Interface Training and Calibration

The sequencer element of the ALTMEMPHY megafunction performs path-delay analysis to correctly set up the resynchronization (**DQS mode devices**), capture (**Non DQS Mode devices**) clocks and the data alignment settings. The sequencer issues read and write commands to the memory controller over the `ctl_*` interface for DDR and DDR2 SDRAM high-performance memory controllers, which is performed in the following two stages:

- Write training data
- Calibration

### Write Training Data


A specific pattern of data is written to the memory so that each DQ pin may be calibrated. The same pattern is written to every DQ pin. The pattern takes the form: “10101010\_11111111\_00000000\_11111100” from the lowest address location to the highest address location left to right. Once the memory interface is initialized, a single write transaction is observed to the memory using the pattern above to write the same pattern to each DQ bit.

### Calibration

When the write training data process is completed, the sequencer then repeatedly reads the training pattern back from the memory. This action is performed on a per DQ pin basis, and for all available phase steps supported by the PLL configuration.

The sequencer phase steps through 360° for a full rate controller and through 720° for a half rate controller. For simulation purposes only, calibration can be performed on just a single DQ pin, which greatly reduces simulation run time. The sequencer stores a list of pass and fail training pattern results and when all phase comparisons have been made, sets the optimum clock phase to be centered in the available window of results.

Simulation of the calibration cycle cannot be bypassed, but setting it to single bit calibration speeds up the process significantly.

 The ALTMEMPHY megafunction only the center of data valid window and read latency. The amount of internal RAM required to store calibration results therefore, is significantly reduced compared to earlier versions. The total calibration time is also reduced.

## Functional Memory Use

When training and calibration completes, the ALTMEMPHY sequencer asserts `ctrl_usr_mode_rdy` to the memory controller, which is then copied to the local interface as the signal `local_init_done`. Local interface read and write transactions can now occur.

In the example testbench, the example driver now performs 16 writes followed by 16 reads to incremental address locations spanning column, row and bank locations using LFSR pattern based on the address being written.

## Functional Simulation—the ModelSim Wave and Transcript Window

To better understand the operation of the ALTMEMPHY megafunction and the memory controllers, identify the various stages of operation, the expected results, and the behavior of the IP. The following sections describe each of these stages in greater detail.

- “Full Window Stage Identification” on page A-20
- “Initialization Stage” on page A-22
- “Write Training Data Stage” on page A-24
- “Read Calibration Phase” on page A-26
- “Functional Memory Use Stage” on page A-30

### Full Window Stage Identification

Figure A-4 on page A-21 shows a standard appearance of an example testbench in the ModelSim software.



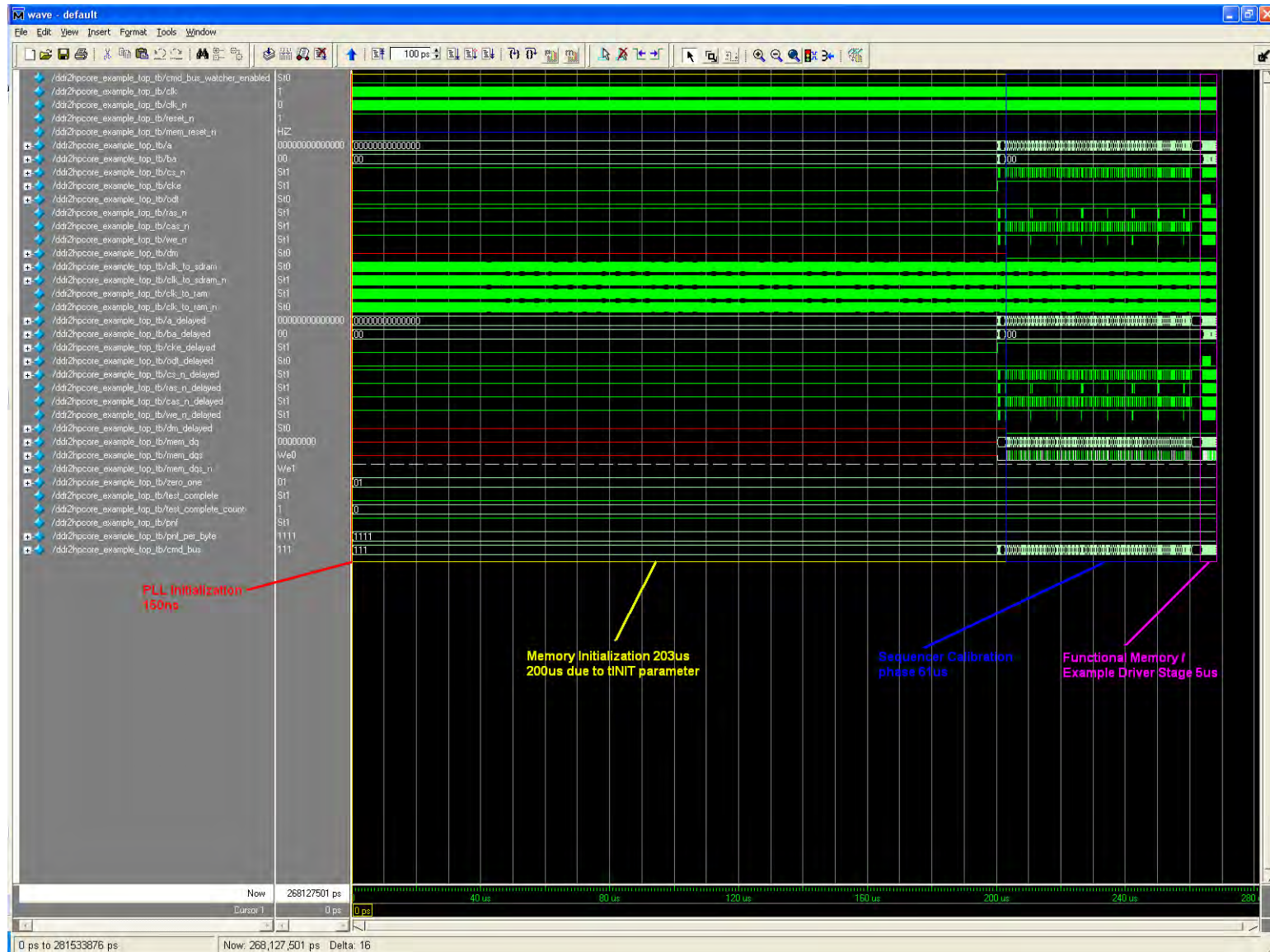
-  The total simulation time around 270  $\mu$ s to 203  $\mu$ s is used for the memory initialization requirements, and in this example 61  $\mu$ s is used by the calibration routine to calibrate using just a single DQ pin. In general, the PLL initialization phase has negligible impact on the overall simulation time.
-  For further information on simulating the PLL, refer to *ALTPLL Megafunction User Guide*.



Figure A-4. Example Testbench



## Initialization Stage

The full window stage shows that the Memory initialization stage is dominated by the NOP command where  $t_{\text{INIT}}$  is 200  $\mu\text{s}$ .



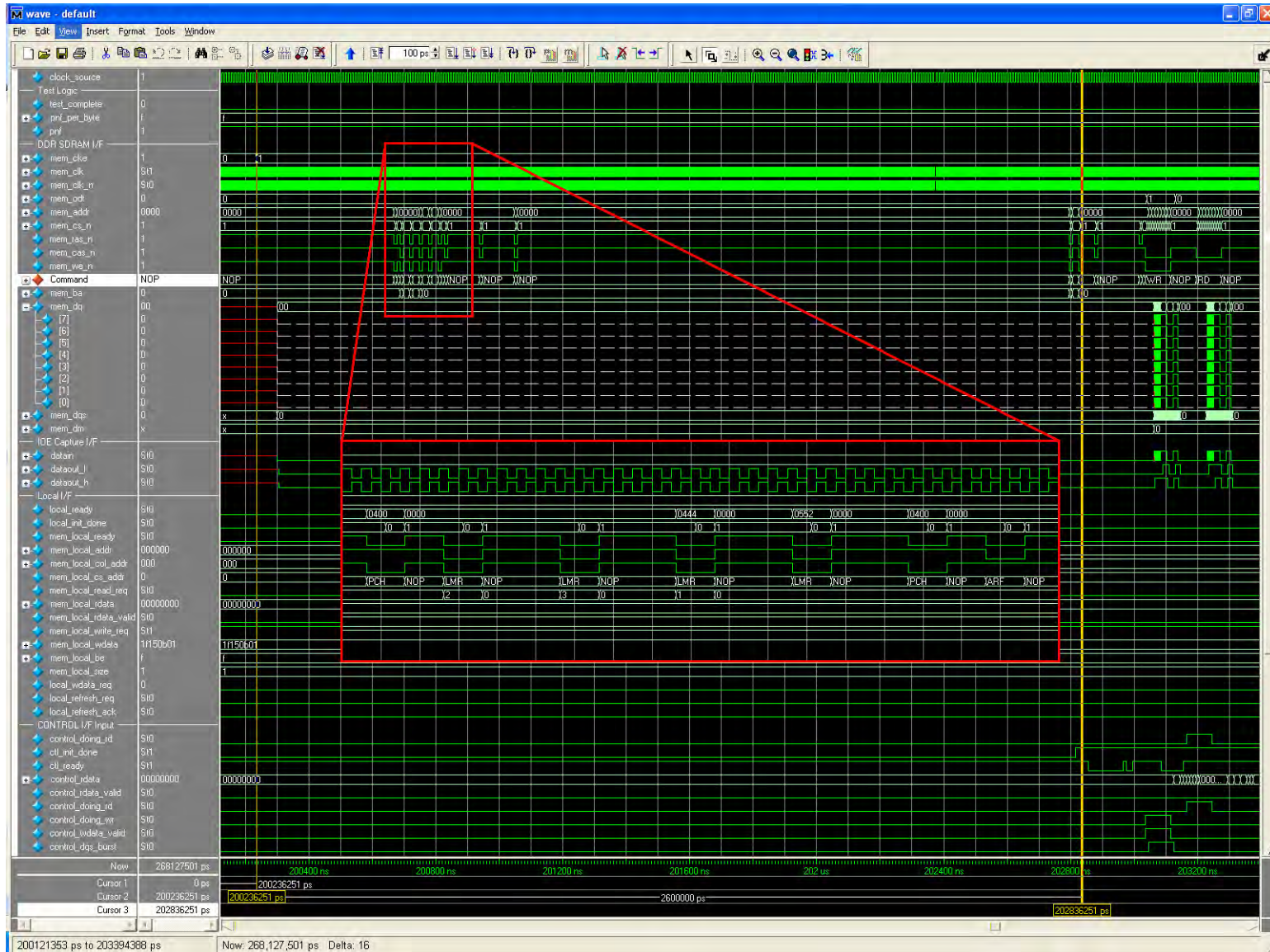
Version 8.1 automatically skips this stage in simulation.

The exact sequence of commands differs between the various external memory families (refer to the respective the device datasheets for further information). For this DDR2 SDRAM example, the following sequence applies:

1. Issue NOP commands for 200  $\mu\text{s}$ , programmable via  $t_{\text{INIT}}$  parameter.
2. Assert `mem_cke` (high).
3. Issue a PCH, then wait for 400 ns after  $t_{\text{INIT}}$  (400 ns is derived from dividing  $t_{\text{INIT}}$  counter by 500).
4. Issue an LMR command to ELMR register 2 = 0.
5. Issue an LMR command to ELMR register 3 = 0.
6. Issue an LMR command to ELMR register to enable the memory DLL and set Drive strength, AL, RTT, DQS#, RDQS, OE.
7. Issue an LMR command to MR register to reset DLL and set operating parameters.
8. Issue a PCH.
9. Issue an ARF.
10. Issue another ARF.
11. Issue an LMR command to MR register to set operating parameters.
12. Issue an LMR command to ELMR register to set default OCD and parameters. 200 clock cycles after DLL reset, the memory is initialized.

See [Figure A-5 on page A-23](#) for the expected waveform view of the initialization phase directly following the NOP of 200  $\mu\text{s}$ . Steps 2 to 9 are expanded to increase detail. Initialization is complete by the second yellow cursor. Additional signals are added to simplify debugging.

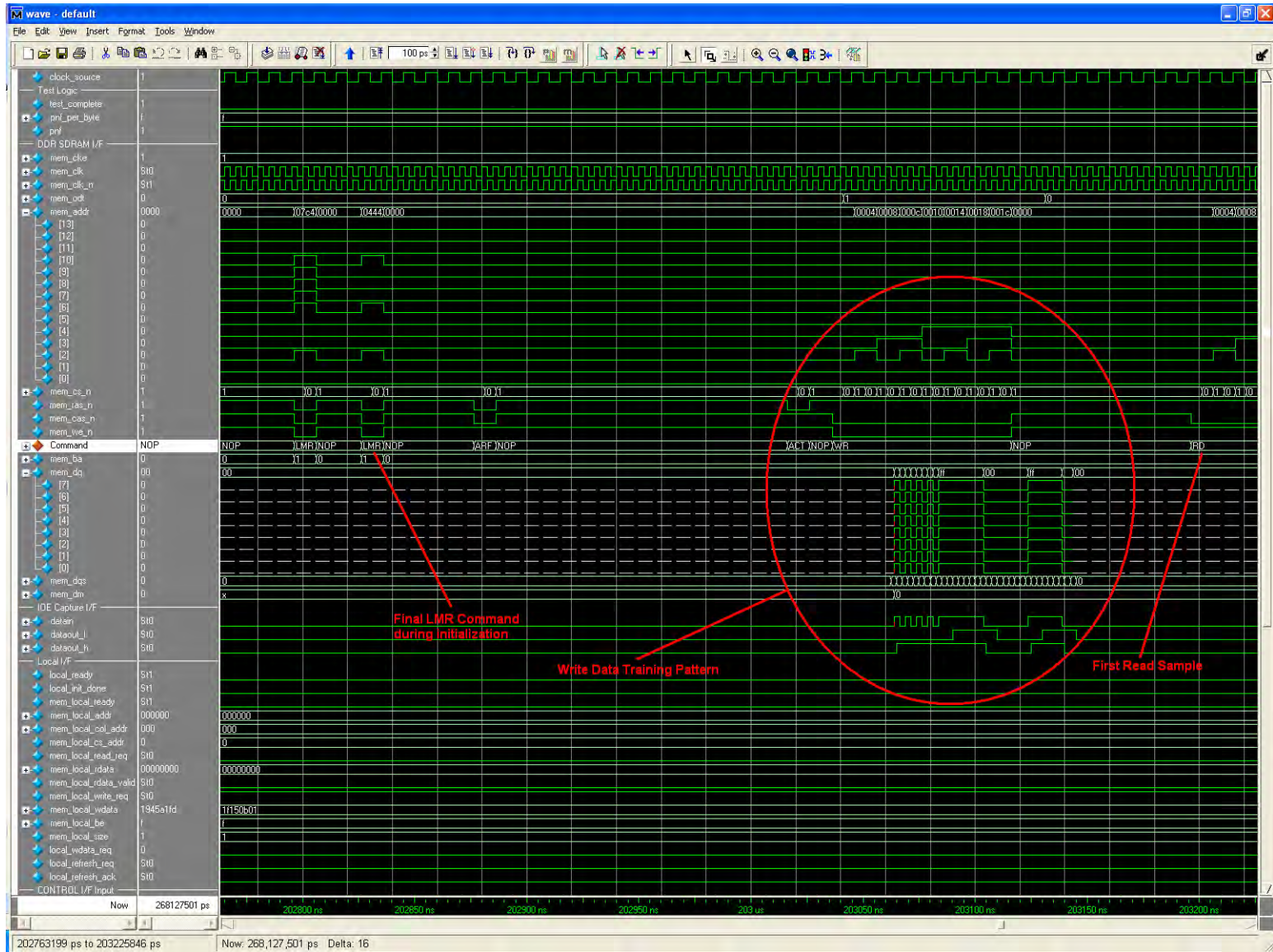
Figure A-5. DDR2 SDRAM Simulation Initialization Phase



## Write Training Data Stage

As shown in [Figure A-6 on page A-25](#), following memory initialization, the write data training stage is closely followed by the first read data calibration read sample.

Figure A-6. Writing Data Training

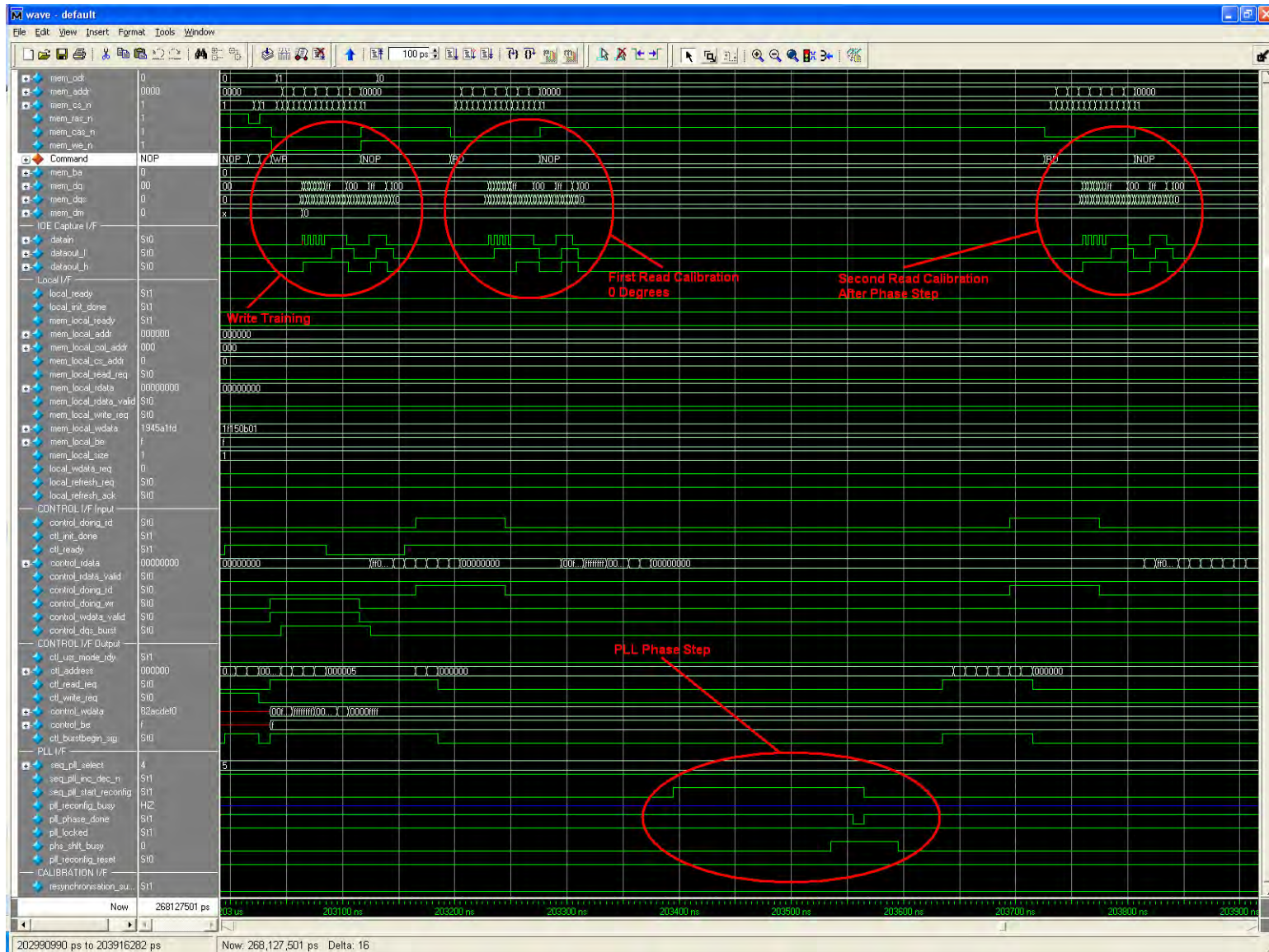


## Read Calibration Phase

Figure A-7 on page A-27 shows that the read data calibration stage closely follows the write data stage. The calibration repeatedly performs the following steps:

1. Reads back the data pattern.
2. Records pass/fail.
3. Increments the PLL phase step. ( $360^\circ$ /number of phase steps).
4. Repeats.

Figure A-7. Read Calibration Phase



This process continues until all 360° (full rate) or 720° (half rate) worth of phases are sampled and the available data window is measured. The length of time between each read is determined by the PLL reconfiguration requirements and the compare and store operation of the sequencer, which is device and configuration dependant. The gap observed between the first and subsequent reads is greater as the sequencer must set the PLL into phase stepping mode.

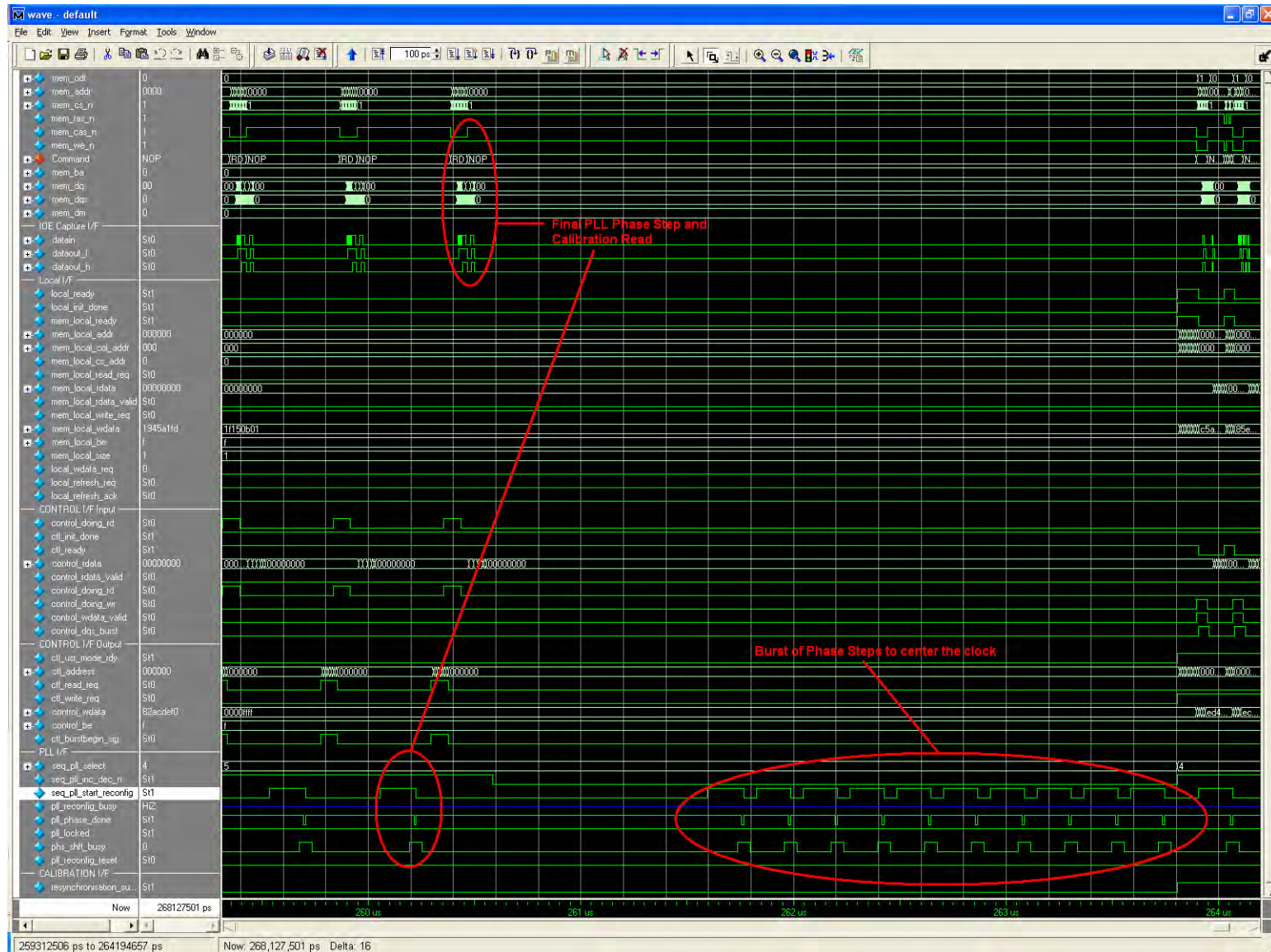
Once a successful range of phases has been sampled, the sequencer center aligns the PLL phase within this calibrated window. This can be observed on the PLL reconfiguration bus as a burst of phase changes without any read accesses. In the example, this calibration is setting on the capture clock phase, so you would (in a functional simulation) expect a phase of around  $-90^\circ$  to be selected. 48 phase steps were reported during the generation, and a burst of 10 PLL reconfigurations can be observed ( $360/48 * 10 = -75^\circ$ ). Refer to [Figure A-8 on page A-29](#).





In Cyclone III devices, calibration is performed on the capture clock and in Arria GX, Stratix II, and Stratix III devices, calibration is performed on the resynchronization clock.



Figure A-8. Calibration Phase - Setting the Optimum Phase




-  Until the completion of the calibration phase, the local interface remains static and all transactions take place over the control interface. Additional signals of interest are added to the wave view.
-  For simulation purposes, the ALTMEMPHY megafunction allows calibration of a single DQ pin. If you do not enable this option, then the time required for the calibration phase of the simulation is multiplied by the number of DQ pins used in your actual memory controller instance. To enable this option, select **Quick Calibration** under **Auto-Calibration Simulation Options** list at the **Memory Settings** tab in the **Parameter Settings** page.

## Functional Memory Use Stage

Once the calibration stage completes, indicated by the signals `local_init_done` and `ctl_usr_mode_rdy`, then the functional memory use stage begins.

The example driver, which is generated by the MegaWizard Plug-In Manager, is clear text HDL in the language of your choice. It can be used to test a custom controller and ALTMEMPHY megafunction combination. It performs a series of writes to the external memory, followed by a series of reads to the same locations, and compares the read and write data.

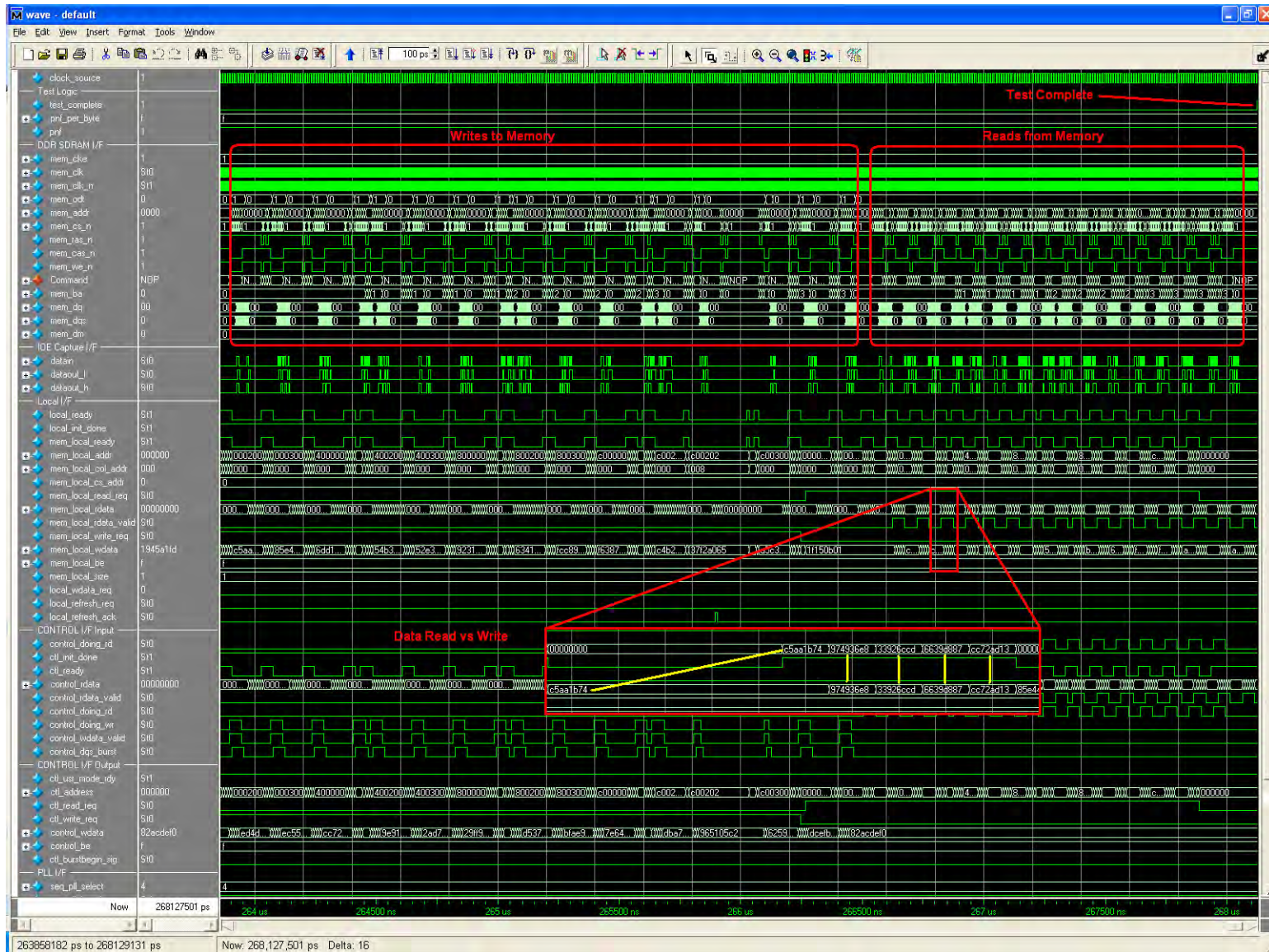
This comparison results in dynamic “pass not fail per byte” (`pnf_per_byte`) signals, and a latched combined pass not fail (`pnf`, 1=pass 0=fail) signal. Each completed series of writes and reads is signaled via the `test_complete` signal, and then the test repeats.

-  The example testbench stops when either `test_complete` is asserted or when 200,000 `mem_clk` cycles after the `tINIT` time.

In [Figure A-9 on page A-31](#), the series of writes followed by reads can be seen on both the local and memory interfaces, together with the test complete signals.

As the data written to the memory is simply an LFSR pattern, the example driver is able to generate expected read data from the memory to compare with that previously written to the same address. The data on the read data bus should match that on the write data bus during the read process.

Figure A-9. Functional Memory Use Stage



## Additional Debug Signals

This section discusses the following debug signals:

- “PLL and PLL Reconfiguration Signals”
- “Calibration Status Interface”
- “Additional Calibration Status Interface Signals”

### PLL and PLL Reconfiguration Signals


Before any design operates correctly, all clock and reset signals must be stable and configured correctly. Therefore, you must ensure that the various PLL ports are visible in your simulation. This is of increased values when simulating a DDR and DDR2 SDRAM high-performance memory controller-based design because of the PLL phase calibration stages that occur.

 Refer to *ALTPLL Megafunction User Guide* for a full description of each signal.

You must add the following ALTMEMPHY signals to your simulation:


- seq\_pll\_select
- phasecounterselect
- seq\_pll\_inc\_dec\_n
- phaseupdown
- seq\_pll\_start\_reconfig
- configupdate
- pll\_reconfig\_busy
- pll\_phase\_done
- pll\_locked
- phs\_shft\_busy
- pll\_reconfig\_reset


In Stratix II, Arria GX, and Stratix II GX designs where a separate `altpll_reconfig` instance is required, the following signals may be added to the simulation:

 As Stratix III and Cyclone II device PLLs include a dynamic phase configuration option directly, these signals do not exist.

- pll\_reconfig
- pll\_reconfig\_counter\_param
- pll\_reconfig\_counter\_type
- pll\_reconfig\_data\_in
- pll\_reconfig\_enable
- pll\_reconfig\_read\_param
- pll\_reconfig\_soft\_reset\_en\_n

- pll\_reconfig\_write\_param
- pll\_reconfig\_busy
- pll\_reconfig\_clk
- pll\_reconfig\_data\_out
- pll\_reconfig\_reset

 Refer to the respective Device Handbooks or *Phase-Locked Loops Reconfiguration (ALTPLL\_RECONFIG) Megafunction User Guide* for a full description of the signals listed and their required operation.

 The output clock order is fixed in the ALTMEMPHY (`seq_pll_select`) and a mapping takes place to align with the required clock port used for each different device family. Refer to `<variation name>_phy_alt_mem_phy.v(hd)` file and refer to the section that starts with the following code:

```
// NB This lookup table shall be different for CIII/SIII
// The PLL phasecounterselect is 3 bits wide, therefore hardcode the
output to 3 bits:
```

## Calibration Status Interface

The following calibration signals provide information for ALTMEMPHY megafunction:

- `ctl_cal_success` indicates calibration success
- `ctl_cal_fail` indicates calibration failure

## Additional Calibration Status Interface Signals

Add the signals in [Table A-11](#) to your simulation to provide additional information about calibration status:

**Table A-11.** Signals for Calibration Status

Signal	Description
<code>rsu_codvw_phase</code> (Center of data valid window)	The ALTMEMPHY resynchronization setup unit (RSU) sweeps the phase of a resynchronization clock across 360° (full-rate mode) or 720° (half-rate mode) of a memory clock cycle. Data reads from the DIMM are performed for each phase position, and a data valid window is located, which is the set of resynchronization clock phase positions where data is successfully read. The final resynchronization clock phase is set at the center of this range: the center of the data valid window or CODVW. This output is set to the current calculated value for the CODVW, and represents how many phase steps were performed by the PLL to offset the resynchronization clock from the memory clock.
<code>rsu_read_latency</code> (Read latency at the center of the window)	If the RSU can find one data valid window (and not more than one) then the resynchronization clock is positioned at the center and the <code>rsu_read_latency</code> output is then set to the read latency (in <code>phy_clk</code> cycles) using that resynchronization clock phase. If calibration is unsuccessful then this signal remains at 0.

**Table A-11.** Signals for Calibration Status

Signal	Description
<code>rsu_no_dvw_err</code> (Calibration failed due to no window found)	If the RSU sweeps the resynchronization clock across every phase and does not see any valid data at any phase position, then calibration fails and this output is set to 1.
<code>rsu_grt_one_dvw_err</code> (Calibration failed due to more than one valid window)	If the RSU sweeps the resynchronization clock across every phase and sees multiple data valid windows, this is indicative of unexpected read data (random bit errors) or an incorrectly configured PLL which must be resolved. Calibration has failed and this output is set to 1.
<code>rsu_multiple_valid_latencies_err</code> (Calibration failed due to more than two read latencies)	If the RSU sweeps the resynchronization clock across every phase and sees valid data at more than two different latencies, then calibration fails and this output is set to 1.

## Design Considerations

This section discusses the design considerations for non-AFI designs.

### Clocks and Resets

The ALTMEMPHY megafunction automatically generates a PLL instance, but you must still provide the reference clock input (`pll_ref_clk`) with a clock of the frequency that you specified in the MegaWizard Plug-In. An active-low global reset input is also provided, which you can de-assert asynchronously. The clock and reset management logic synchronizes this reset to the appropriate clock domains inside the ALTMEMPHY megafunction. A clock output, which is half the memory clock frequency for a half-rate controller and the same as the memory clock for a full-rate controller, is provided (`phy_clk` or `aux_half_rate_clk`) and all inputs and outputs of the ALTMEMPHY megafunction are synchronous to this clock. An active-low synchronous reset is also provided (`reset_phy_clk_n`). This `reset_phy_clk_n` signal is synchronously de-asserted with respect to the `phy_clk` clock domain and can reset any additional user logic on that clock domain. In addition, there is a full rate clock (`aux_full_rate_clk`) output available to use anywhere else in your design. This clock is derived from the `mem_clk_2x` PLL output signal.

### Calibration Process Requirements

As the autocalibration logic makes use of the controller to perform its calibration, you should follow these guidelines at power-up. When the global reset (`global_reset_n`) is released, the clock management logic waits for the PLL to lock and then releases the reset to the rest of the logic, including the controller. As the PLL locked output is gated inside the PLL, for approximately the first 10,000 cycles (by this time the PLL-locked output is stable) of the PLL reference clock, there is no activity at this time. When the reset to the controller (`reset_phy_clk_n`) is released, the controller begins its normal memory initialization sequence. When complete, the controller indicates to the ALTMEMPHY megafunction that it is ready to accept the calibration writes and reads by asserting the `ctl_init_done` and `ctl_ready` signals in DDR3/DDR2/DDR SDRAM interfaces. (There are no such signals in QDRII+/QDRII SRAM interfaces, as the SRAM device does not need an initialization sequence other than initializing the memory DLL.) The auto-calibration logic then

issues a series of writes and reads to the external memory. You do not have access to the memory controller during this period. When the autocalibration logic completes its calibration, the ALTMEMPHY megafunction asserts the `ctl_usr_mode_rdy`, `resynchronization_successful`, `local_init_done`, and `local_ready` signals in DDR3/DDR2/DDR SDRAM interfaces or `ctl_usr_mode_rdy` and `resynchronization_successful` signals in QDRII+/QDRII SRAM interfaces. You then have complete control of the memory controller.


 For more information about calibration process, refer to the the simulation waveforms in the *DDR and DDR2 SDRAM High-Performance Controller User Guide*.

## Local Interface Requirements

The autocalibration logic makes use of the controller to perform its calibration, so your controller must observe the following requirements.


The controller must have at least one Avalon-MM slave interface or a native interface, as defined in the *DDR3 SDRAM High-Performance Controller User Guide* and the *DDR and DDR2 SDRAM High-Performance User Guide*, which the ALTMEMPHY megafunction can control during the initial calibration process. For DDR3 SDRAM and QDRII+/QDRII SRAM variations of the ALTMEMPHY megafunction, only the Avalon-MM local interface is supported. When calibration is complete, no further access to this interface is required by the ALTMEMPHY megafunction.


The memory burst length can be two, four, or eight for DDR SDRAM devices; the memory burst length can be four or eight for DDR2 SDRAM devices; DDR3 SDRAM burst lengths can be set at either four or eight, when using the Altera high-performance controller. The QDRII+/QDRII SRAM variations of the ALTMEMPHY megafunction only support burst length of four. For a half-rate controller, the memory clock runs twice as fast as the clock provided to the local interface; so data buses on the local interface are four times as wide as the memory data bus. For a full-rate controller, the memory clock runs at the same speed as the clock provided to the local interface, so the data buses on the local interface are two times as wide as the memory data bus. Each read or write request on the local interface fits into a single memory read or write command on the memory interface, simplifying the controller design.

 The ALTMEMPHY megafunction with the non-AFI does not support burst lengths of eight.

## DDR2/DDR SDRAM Half-Rate Controller

The following sections describe the handshake mechanism between the read commands and read data for the controller in a DDR2/DDR SDRAM interface.

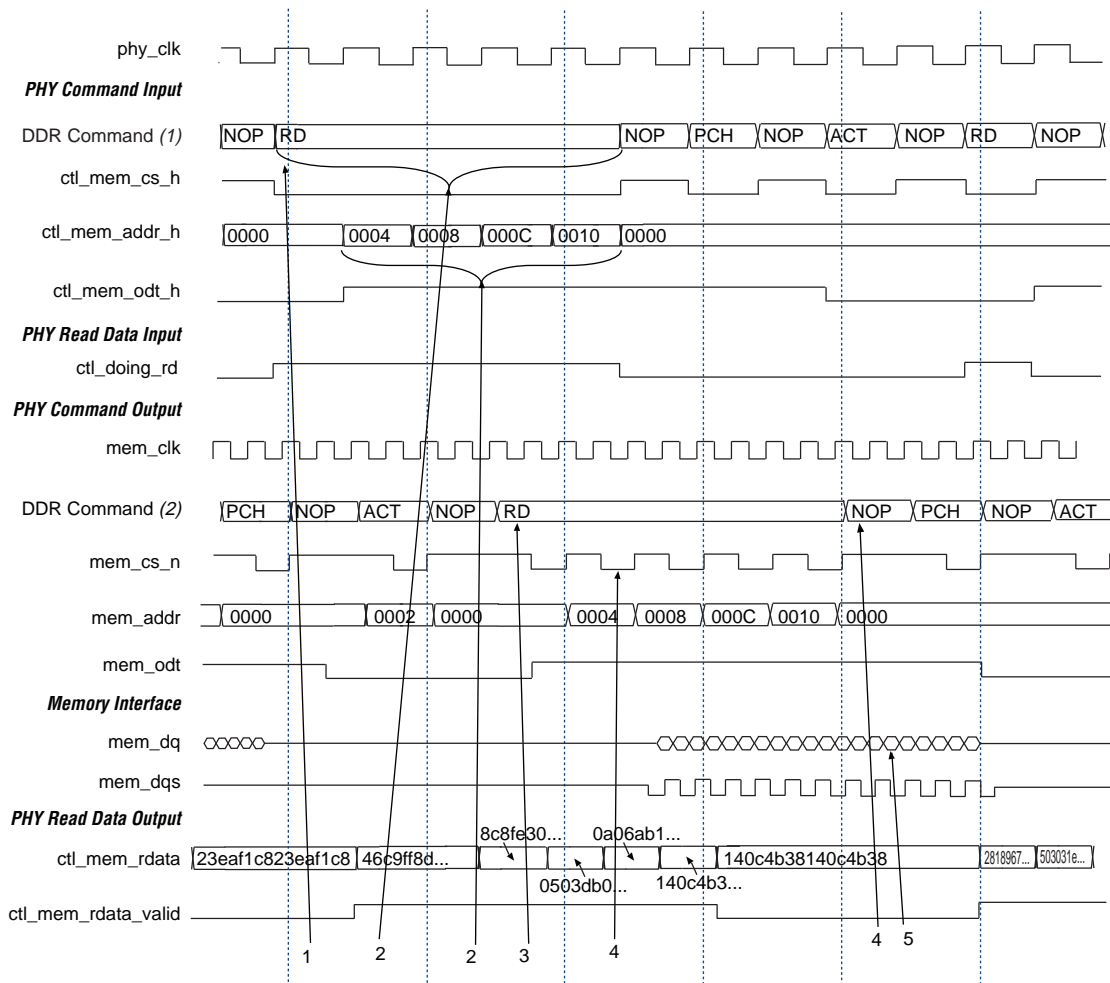
 For more information about the simulation waveforms of a DDR2 SDRAM High-Performance controller, refer to the figures in the Interfaces and Signals section and Appendix B of the *DDR and DDR2 SDRAM High-Performance Controller MegaCore Function User Guide*.

 The behavior of `ctl_*` signals is the same as `local_*` signals during calibration. These signals switch to `local_*` signals after calibration.

## Handshake Mechanism Between Read Commands and Read Data

The controller generates a signal (`ctl_doing_rd`) to the ALTMEMPHY megafunction which is asserted for one `phy_clk` cycle for every read command it issues. If there are two read commands, the signal `ctl_doing_rd` is asserted for two `phy_clk` cycles. This signal also enables the capture registers and generates the `ctl_mem_rdata_valid` signal. This signal should be issued at the same time the read command is sent to the ALTMEMPHY megafunction (refer to [Figure A-10](#)).

**Figure A-10.** Read Commands and Read Data (Half-Rate Controller)



### Notes to Figure A-10:

- (1) The DDR command shows the command comprised of the command signals (`ctl_mem_ras_n_h`, `ctl_mem_cas_n_h`, and `ctl_mem_we_n_h`) seen at the ALTMEMPHY input. There can be more than one clock cycle of no operation (NOP) between active (ACT) to RD depending on the value of  $t_{RCD}$  parameter of your memory device.
- (2) The DDR command shows the command comprised of the command signals (`mem_ras_n_h`, `mem_cas_n_h` and `mem_we_n_h`) seen at the memory interface. There can be more than one clock cycle of NOP between active ACT to RD depending on the value of the  $t_{RCD}$  parameter of your memory device.



The signals under the PHY Command Input label are the signals from the controller to the ALTMEMPHY megafunction. The signals under the PHY Command Output label are the signals coming out of the ALTMEMPHY megafunction and input to the memory device.



Some of the address and command signals generated by the controller are:

- `ctl_mem_addr_h`
- `ctl_mem_cas_h`
- `ctl_mem_cs_n_h`
- `ctl_mem_ras_n_h`
- `ctl_mem_we_n_h`
- `ctl_mem_odt_h`

Figure A-10 shows the No Operation (NOP) command followed by a series of five read commands.

1. The controller issues five consecutive read commands with a starting address of `0x0` with increments of four (`0000`, `0004`, `0008`, `000c`, `0010`), see the top of Figure A-10 under the **PHY Command Input** label.
2. The ALTMEMPHY megafunction generates the read command at the memory interface after five-to-seven memory clock (`mem_clk`) cycles. The address and commands are generated using the negative edge of the memory clock, see Figure A-10 under the **PHY Command Output** label.
3. The address and commands are of 2T period and the chip select is of 1T period (`mem_clk`).
4. The data (`mem_dq`) at the memory interface is presented after three memory clock cycles of read latency. The read latency is equal to the CAS latency. For this example, CAS latency is equal to three.

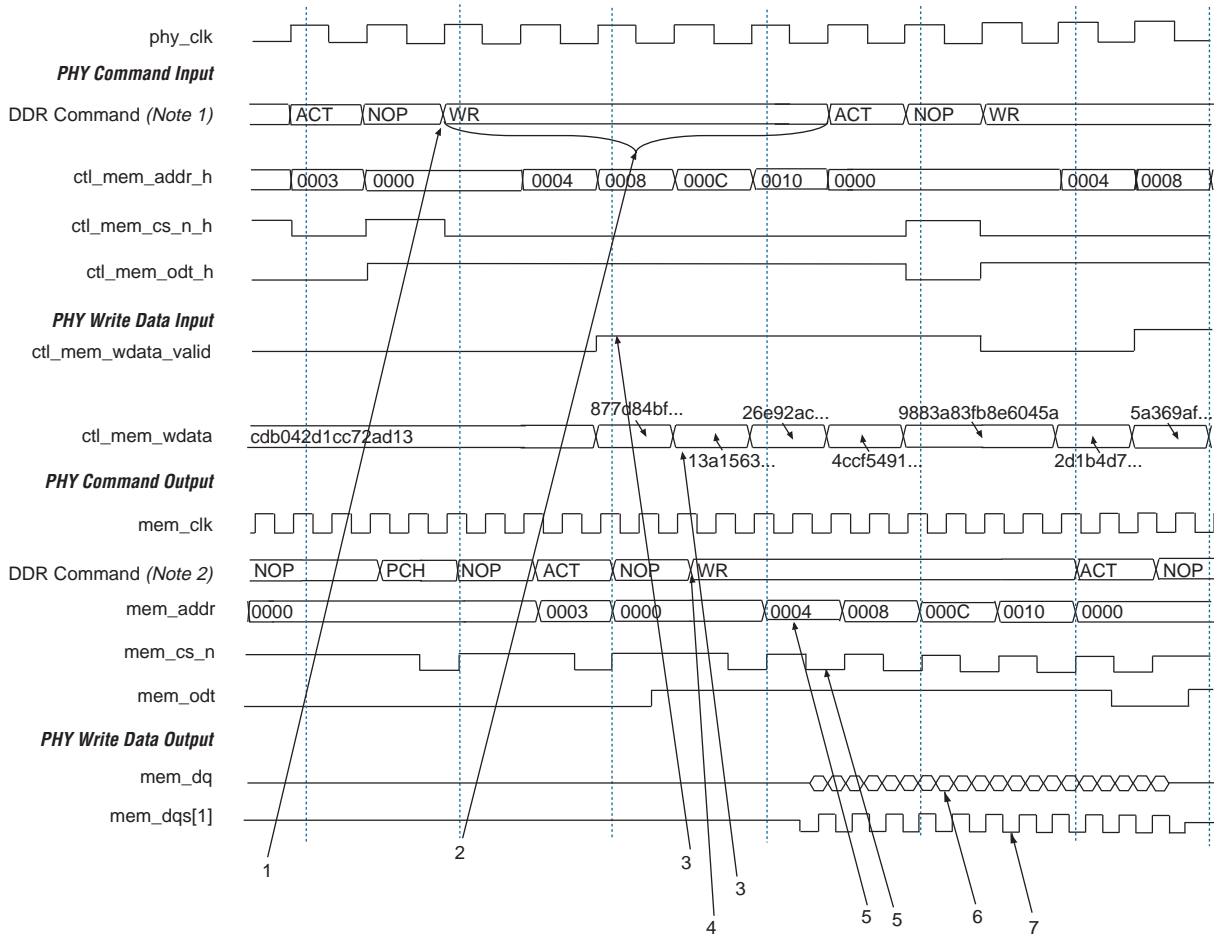
By default, the read data from the memory bypasses the controller and is sent directly to the user logic. If your controller requires access to the read data after it has been captured, but before it is sent to the user interface (for example, to perform error detection and correction), connect the `ctl_mem_rdata` and `ctl_mem_rdata_valid` outputs from the ALTMEMPHY megafunction to your controller. The controller must delay both `ctl_mem_rdata` and `ctl_mem_rdata_valid` signals by the same amount. Connect the read data and valid outputs of your controller to the `ctl_rdata` and `ctl_rdata_valid` inputs of the ALTMEMPHY megafunction, which passes straight through to the `local_rdata` and `local_rdata_valid` signals.

### Handshake Mechanism Between Write Commands and Write Data

The controller provides a signal (`ctl_mem_wdata_valid`) to the ALTMEMPHY megafunction to tell it when to enable the `mem_dq` and `mem_dqs` output enables. It is the controller's responsibility to control the relative timing of the memory command signals (for example, `mem_cas_n` and `mem_we_n`) and the `control_mem_wdata_valid` or `control_mem_dqs_burst` signals, to meet the required memory write latency; therefore, this exact relationship is very important.



The `ctl_mem_dqs_burst` signal controls the DQS output enables of the DQS pins.

**Figure A-11.** Write Commands and Write Data (Half-Rate Controller)**Notes to Figure A-11:**

- (1) The DDR command shows the command comprised of the command signals (ctl\_mem\_ras\_n\_h, ctl\_mem\_cas\_n\_h, and ctl\_mem\_we\_n\_h) seen at the ALTMEMPHY input. There can be more than one clock cycle of NOP between ACT to RD depending on the value of  $t_{\text{RCD}}$  parameter of your memory device.
- (2) The DDR command shows the command comprised of the command signals (mem\_ras\_n\_h, mem\_cas\_n\_h, and mem\_we\_n\_h) seen at the memory interface.

Figure A-11 shows the sequence of operations that happen during the write transactions. The write operation is explained step-by-step below. All the inputs to the ALTMEMPHY megafunction from the controller should be generated using phy\_clk.



The signals under the PHY command input label are the signals from the controller to the ALTMEMPHY megafunction and the signals under the PHY command output label are the signals coming out of the ALTMEMPHY megafunction and input to the memory device.

Some of the address and command signals generated by the controller are:

- ctl\_mem\_addr\_h
- ctl\_mem\_cas\_n\_h

- `ctl_mem_cs_n_h`
- `ctl_mem_ras_n_h`
- `ctl_mem_we_n_h`
- `ctl_mem_odt_h`

As can be seen in the waveform (Figure A-11), the sequence of commands are PreCharge (PCH), ACT, and NOP, followed by a series of write commands.

1. The controller puts the five consecutive write commands with a starting address of  $0 \times$  with increments of four (0000, 0004, 0008, 000c, 0010), see the top of Figure A-11 under the **PHY Command Input** label.
2. The controller generates the following signals two clock cycles after `ctl_mem_wdata_valid` and must supply the data `ctl_mem_wdata` along with these two clock cycles. Refer to Figure A-11 under the PHY Write Data Input label.
3. The ALTMEMPHY megafunction generates the write command at the memory interface after five-to-seven memory clock (`mem_clk`) cycles (to accommodate the write delay). In this example, the address and commands are generated using the negative edge of the memory clock, see Figure A-12 under the **PHY Command Output** label.
4. The address and commands are of 2T period and the chip select is of 1T period.
5. The data (`mem_dq`) at the memory interface is presented after two memory clock cycles of write latency. The write latency is equal to the CAS latency -1 for DDR2 SDRAM only (for DDR SDRAM it is always 1). For this example, CAS latency is equal to three.
6. The generation of DQS signals is controlled using the `control_mem_wdata_valid` signal, which is very important as the generation of the DQS signal is also dependent on the CAS latency parameter.

Figure A-11 shows that the controller state machine asserts the `ctl_mem_wdata_valid` signal to perform a write transaction. The write data (`ctl_mem_wdata`) should be available at the same time when the signal is asserted high. The `ctl_mem_wdata_valid` signal is asserted for five clock cycles (`phy_clk`) or ten clock cycles (`mem_clk`) to transfer five data transfers. The write data is only valid when the `ctl_mem_wdata_valid` is asserted high and is held in the `wdata` registers until the write occurs. In Figure A-11, the write data bus `ctl_mem_wdata` is of width 64 and each burst transfer is of the length four. The 64-bit wide data is transferred to the memory as four 16-bit wide data, as shown by `mem_dq`. The DQS clock is twice the frequency of the clock that clocks the `ctl_mem_wdata` and the DQ data is transferred during both the edges of DQS.

## DDR2/DDR SDRAM Full-Rate Controller

The following section details the handshake read and handshake write function for the DDR2/DDR SDRAM full-rate controller.



For more information about the simulation waveforms of a DDR2 SDRAM High-Performance controller, refer to the figures in the Interfaces and Signals section and Appendix B of the *DDR and DDR2 SDRAM High-Performance Controller MegaCore Function User Guide*.

## Handshake Mechanism Between Read Commands and Read Data

Figure A-12 shows the read operation for a full-rate controller. The handshake mechanism remains similar to that of the half-rate controller except for the following differences:

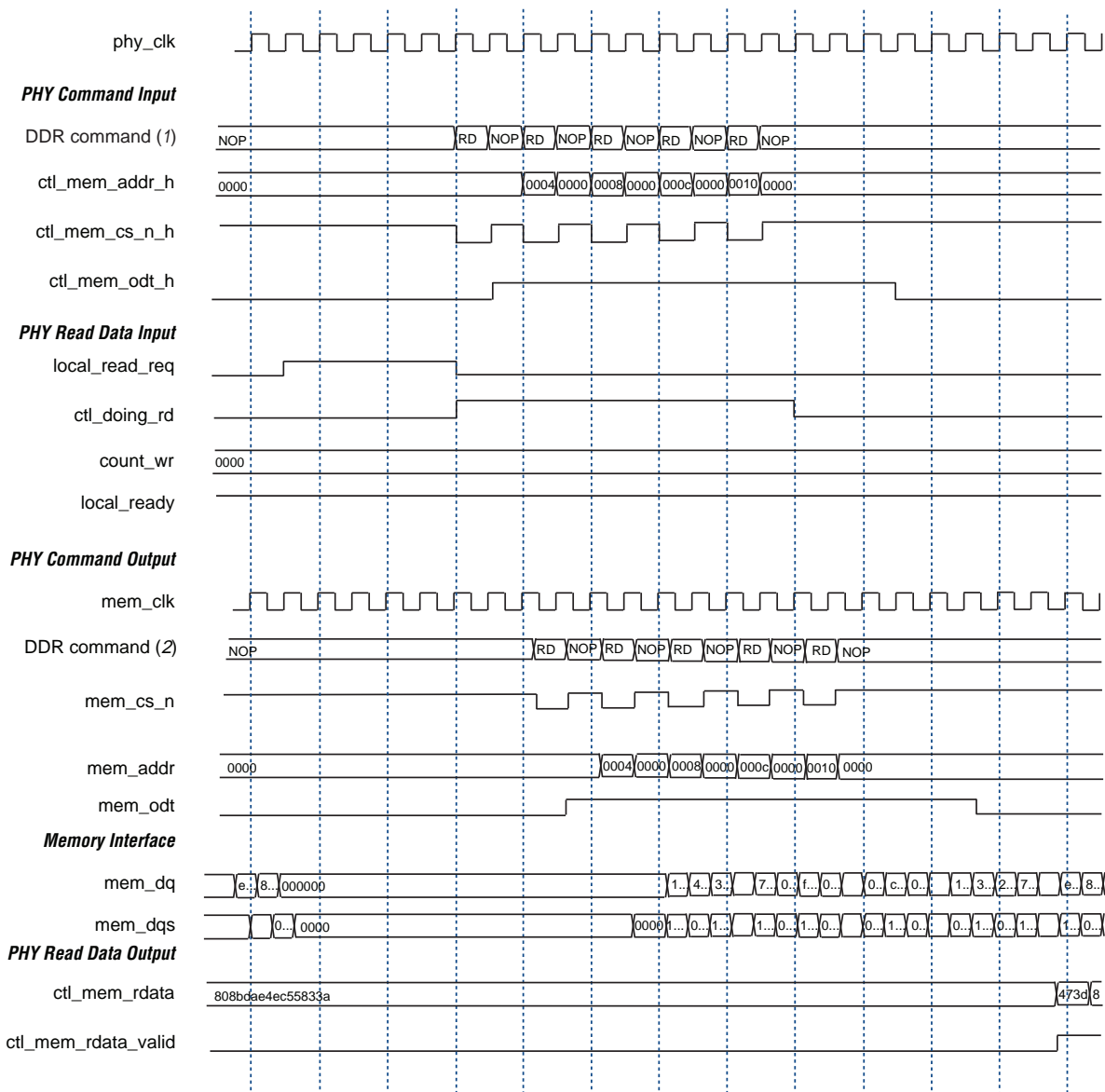
1. 1T versus 2T addressing.

As the burst size is fixed at four on the memory interface, and also the address and command datapath is based on 1T addressing, it takes two memory clock cycles to retrieve the data from the memory for each read command, see Figure A-12. The first memory cycle is the read command and the second memory cycle is the NOP command. Because of this arrangement, you see a NOP command between the read commands.

2. Assertion of the chip select signal.

The chip select signal is asserted along with the read command because of the 1T addressing.

**Figure A-12.** Read Commands and Read Data (Full-Rate Controller)



**Notes to Figure A-12:**

- (1) The DDR command shows the command comprised of the command signals (ctl\_mem\_ras\_n\_h, ctl\_mem\_cas\_n\_h, and ctl\_mem\_we\_n\_h) seen at the ALTMEMPHY input. There can be more than one clock cycle of NOP between ACT to RD depending on the value of  $t_{\text{RCD}}$  parameter of your memory device.
- (2) The DDR command shows the command comprised of the command signals (mem\_ras\_n\_h, mem\_cas\_n\_h, and mem\_we\_n\_h) seen at the memory interface. There can be more than one clock cycle of NOP between ACT to RD depending on the value of the  $t_{\text{RCD}}$  parameter of your memory device.

## Handshake Mechanism Between Write Commands and Write Data

Figure A-13 shows the write operation for the full-rate controller. The handshake mechanism remains similar to the half-rate controller except for the following differences:

1. 1T versus 2T addressing.

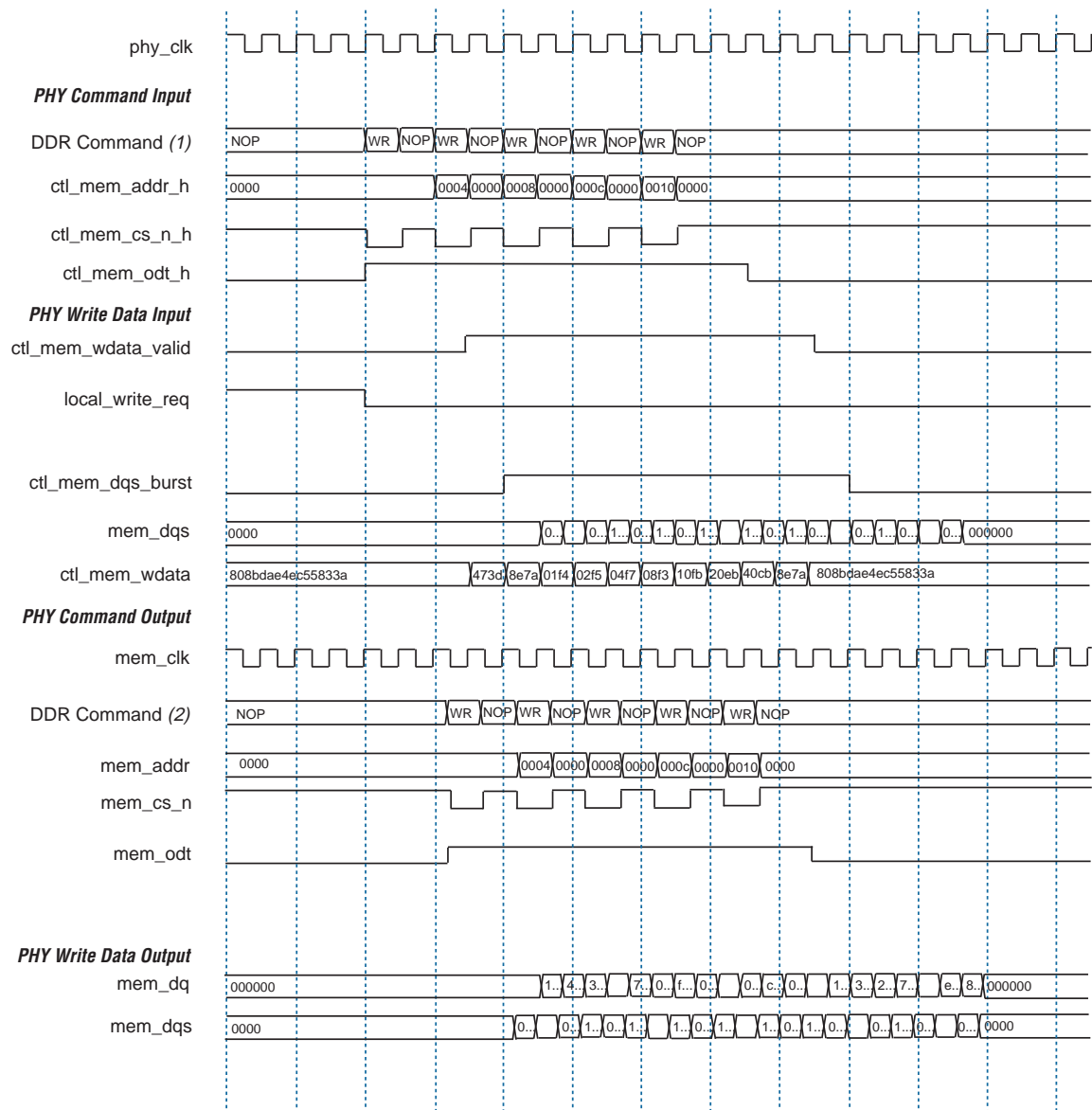
As the burst size is fixed at four on the memory interface, and also the address and command datapath is based on 1T addressing, it takes two memory clock cycles to write data into the memory for each of the write commands, see Figure A-13. The first memory cycle is the write command and the second memory cycle is the NOP command. Because of this arrangement, you see a NOP command between the write commands.

2. Assertion of the chip select signal.

The chip select signal is asserted along with the write command because of the 1T addressing.

3. To support full-rate, the controller must provide the `ctl1_mem_dqs_burst` signal. In full-rate mode, the PHY allows separates control of the DQ and DQS output enables to support incomplete bursts. For example, if the memory burst length is four and the local side burst length is two, you may ask for a write of length one. To support this, the controller must be able to enable the DQS outputs for the full memory burst length (two clock cycles, four DQS edges) while only enabling the DQ outputs for the number of cycles that you requested (one clock cycle, two beats of data).

**Figure A-13.** Write Commands and Write Data (Full-Rate Controller)



**Notes to Figure A-13:**

- (1) The DDR command shows the command comprised of the command signals (`ctl_mem_ras_n_h`, `ctl_mem_cas_n_h`, and `ctl_mem_we_n_h`) seen at the ALTMEMPHY input. There can be more than one clock cycle of NOP between ACT to RD depending on the value of  $t_{\text{RCD}}$  parameter of your memory device.
- (2) The DDR command shows the command comprised of the command signals (`mem_ras_n_h`, `mem_cas_n_h`, and `mem_we_n_h`) seen at the memory interface.

For DDR SDRAM, the write latency is fixed at one memory clock cycle, but for DDR2 SDRAM, this value changes with the read CAS latency. As the controller is running at half the rate of the memory clock, a latency change of one controller clock cycle is actually two memory clock cycles. The ALTMEMPHY megafunction allows you to dynamically insert an extra memory clock of delay in the address and command path to compensate for this. The insertion of delay is controlled by the `ADDR_CMD_ADD_1T` parameter and the `ctl_add_1t_ac_lat` signal. If `ADDR_CMD_ADD_1T` is set to the

string **EXT\_SELECT**, an extra cycle of latency can be dynamically inserted on the address and command outputs by asserting the `ctl_add_1t_ac_lat` input. This allows run-time control of the address and command latency. If `ADDR_CMD_ADD_1T` is set to the string value **TRUE**, the extra clock cycle of latency is always present, and if it is set to the string value **FALSE**, the extra latency is never added.



The `ADDR_CMD_ADD_1T` parameter value is set in `<variation_name>_phy.v` file and is passed on to `<project directory>\<variation_name>_alt_mem_phy.v` file.

For DDR2 SDRAM interfaces using unbuffered DIMMs or components, the value of `ADDR_CMD_ADD_1T` should be **TRUE** for odd CAS latencies (CL3 or CL5) and **FALSE** for even CAS latencies (CL4). For registered DIMMs, the value of `ADDR_CMD_ADD_1T` should be **FALSE** for odd CAS latencies (CL3 or CL5) and **TRUE** for even CAS latencies (CL4), see [Table A-12](#).

For DDR SDRAM interfaces, the write latency is fixed at one cycle. You should use the settings for CAS latencies see [Table A-12](#).

[Table A-12](#) shows the setting of `ADDR_CMD_ADD_1T` for different values of CAS latency and DIMM settings.

**Table A-12.** ADDR\_CMD\_ADD\_1T Settings

Memory and CL	DIMM Type	ADDR_CMD_ADD_1T
DDR2 SDRAM, CL3	Unbuffered	TRUE
	Registered	FALSE
DDR2 SDRAM, CL4	Unbuffered	FALSE
	Registered	TRUE
DDR2 SDRAM, CL5	Unbuffered	TRUE
	Registered	FALSE
DDR SDRAM	Unbuffered	FALSE
	Registered	TRUE

The timing of the ODT signal can be controlled in the same way but is independent of the address and command latency. If the `ODT_ADD_1T` parameter is set to **EXT\_SELECT**, an extra cycle of latency can be dynamically inserted on the ODT command outputs by asserting the `ctl_add_1t_odt_lat` input. This allows separate run-time control of the latency of the ODT signal. If `ODT_ADD_1T` is set to **TRUE**, the extra clock cycle of latency is always present. If `ODT_ADD_1T` is set to **FALSE**, the extra latency is never added.



## Revision History

The table below displays the revision history for the chapters in this user guide.

Date and Document Version	Changes Made	Summary of Changes
April 2009, v7.1	<ul style="list-style-type: none"> <li>■ Added DDR3 SDRAM without leveling</li> <li>■ Added new signals:               <ul style="list-style-type: none"> <li>→ aux_scan_clk_reset_n</li> <li>→ PLL reconfiguration signals</li> </ul> </li> </ul>	—
March 2009, v7.0	<ul style="list-style-type: none"> <li>■ Included Arria II GX information</li> <li>■ Updated generated files list</li> <li>■ Updated AFI information</li> <li>■ Moved non-AFI information to appendix</li> <li>■ Added AFI timing diagrams for reads</li> <li>■ Updated calibration process to indicate multiple chip select calibration is supported by all devices</li> <li>■ Added “DDR3 SDRAM (Discrete Device) and DDR2/DDR SDRAM Calibration” section</li> <li>■ Added clock sharing information</li> <li>■ Changed <b>pin_assignments.tcl</b> description</li> <li>■ Added section on dynamic OCT support</li> </ul>	—
November 2008, v6.0	<ul style="list-style-type: none"> <li>■ Included HardCopy IV information</li> <li>■ Added Altera PHY interface (AFI) information</li> <li>■ Amended HDL source file names to remove <i>&lt;device name&gt;</i></li> <li>■ Added ×36 emulation information</li> </ul>	—
July 2008, v5.0	<ul style="list-style-type: none"> <li>■ Included HardCopy III and Stratix IV information</li> <li>■ Updated to include changes in the Quartus II software version 8.0</li> <li>■ Moved Appendix A to the end of Chapter 1</li> <li>■ Updated all sections in Chapters 1, 2, and 3</li> <li>■ Added Chapters 4, 5, 6, and 7</li> </ul>	—
December 2007 v4.1	Updated Figure A–4.	—
December 2007 v4.0	Updated to include changes in the Quartus II software version 7.2.	—
June 2007 v3.0	Updated to include Arria GX and changes included in the Quartus II software version 7.1.	—

Date and Document Version	Changes Made	Summary of Changes
March 2007 v2.0	Updated to include Cyclone III information.	—
February 2007 v1.0	Initial release.	—

## How to Contact Altera

For the most up-to-date information about Altera products, see the following table.

Contact <i>(Note 1)</i>	Contact Method	Address
Technical support	Website	<a href="http://www.altera.com/support">www.altera.com/support</a>
Technical training	Website	<a href="http://www.altera.com/training">www.altera.com/training</a>
	Email	<a href="mailto:custrain@altera.com">custrain@altera.com</a>
Altera literature services	Email	<a href="mailto:literature@altera.com">literature@altera.com</a>
Non-technical support (General)	Email	<a href="mailto:nacomp@altera.com">nacomp@altera.com</a>
(Software Licensing)	Email	<a href="mailto:authorization@altera.com">authorization@altera.com</a>







**Note:**

(1) You can also contact your local Altera sales office or sales representative.

## Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
<b>Bold Type with Initial Capital Letters</b>	Indicates command names, dialog box titles, dialog box options, and other GUI labels. For example, <b>Save As</b> dialog box. For GUI elements, capitalization matches the GUI.
<b>bold type</b>	Indicates directory names, project names, disk drive names, file names, file name extensions, dialog box options, software utility names, and other GUI labels. For example, <b>qdesigns</b> directory, <b>d:</b> drive, and <b>chiptrip.gdf</b> file.
<i>Italic Type with Initial Capital Letters</i>	Indicates document titles. For example, <i>AN 519: Stratix IV Design Guidelines</i> .
<i>Italic type</i>	Indicates variables. For example, $n + 1$ . Variable names are enclosed in angle brackets (< >). For example, <file name> and <project name>.pof file.
Initial Capital Letters	Indicates keyboard keys and menu names. For example, Delete key and the Options menu.
“Subheading Title”	Quotation marks indicate references to sections within a document and titles of Quartus II Help topics. For example, “Typographic Conventions.”

Visual Cue	Meaning
Courier type	Indicates signal, port, register, bit, block, and primitive names. For example, <code>data1</code> , <code>tdi</code> , and <code>input</code> . Active-low signals are denoted by suffix <code>n</code> . For example, <code>resetn</code> .  Indicates command line commands and anything that must be typed exactly as it appears. For example, <code>c:\qdesigns\tutorial\chiptrip.gdf</code> .  Also indicates sections of an actual file, such as a Report File, references to parts of files (for example, the AHDL keyword <code>SUBDESIGN</code> ), and logic function names (for example, <code>TRI</code> ).
1., 2., 3., and a., b., c., and so on.	Numbered steps indicate a list of items when the sequence of the items is important, such as the steps listed in a procedure.
	Bullets indicate a list of items when the sequence of the items is not important.
	The hand points to information that requires special attention.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or your work.
	A warning calls attention to a condition or possible situation that can cause you injury.
	The angled arrow instructs you to press Enter.
	The feet direct you to more information about a particular topic.

## References

- [AN 328: Interfacing DDR2 SDRAM with Stratix II, Stratix II GX, and Arria GX Devices](#)
- [AN 435: Using DDR & DDR2 SDRAM in Stratix III and Stratix IV Devices](#)
- [AN436: Using DDR3 SDRAM with Stratix III and Stratix IV Devices](#)
- [AN 445: Design Guidelines for Implementing DDR and DDR2 SDRAM in Cyclone III Devices](#)
- [AN 461: Designing QDRII+ and QDRII SRAM Interfaces in Stratix III Devices](#)
- [AN 462: Implementing Multiple Memory Controllers Using the ALTMEMPHY Megafunction](#)
- [AN 517: Using High-Performance DDR, DDR2, DDR3 SDRAM with SOPC Builder](#)
- [External Memory Interfaces](#) chapter of the *Stratix III Device Handbook*
- [External Memory Interfaces](#) chapter of the *Stratix IV Device Handbook*
- [DDR and DDR2 SDRAM High-Performance Controller User Guide](#)
- [DDR and DDR2 SDRAM Controller Compiler User Guide](#)
- [ALTDLL and ALTDQ\\_DQS Megafunctions User Guide](#)