External Memory Interfaces Intel® Stratix® 10 FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **21.3**

IP Version: **19.2.4**

Online Version
Send Feedback

UG-S10EMI

ID: 683741
Version: 2022.03.11

Contents

1. Release Information 9

2. External Memory Interfaces Intel Stratix® 10 FPGA IP Introduction 10
2.1. Intel Stratix 10 EMIF IP Design Flow 11
2.2. Intel Stratix 10 EMIF IP Design Checklist 12

3. Intel Stratix 10 EMIF IP Product Architecture 14
3.1. Intel Stratix 10 EMIF Architecture: Introduction 14
3.1.1. Intel Stratix 10 EMIF Architecture: I/O Subsystem 15
3.1.2. Intel Stratix 10 EMIF Architecture: I/O Column 16
3.1.3. Intel Stratix 10 EMIF Architecture: I/O SSM 16
3.1.4. Intel Stratix 10 EMIF Architecture: I/O Bank 17
3.1.5. Intel Stratix 10 EMIF Architecture: I/O Lane 18
3.1.6. Intel Stratix 10 EMIF Architecture: Input DQS Clock Tree 20
3.1.7. Intel Stratix 10 EMIF Architecture: PHY Clock Tree 21
3.1.8. Intel Stratix 10 EMIF Architecture: PLL Reference Clock Networks 21
3.1.9. Intel Stratix 10 EMIF Architecture: Clock Phase Alignment 22
3.2. Intel Stratix 10 EMIF Sequencer 23
3.2.1. Intel Stratix 10 EMIF DQS Tracking 24
3.3. Intel Stratix 10 EMIF Calibration 24
3.3.1. Intel Stratix 10 Calibration Stages 25
3.3.2. Intel Stratix 10 Calibration Stages Descriptions 25
3.3.3. Intel Stratix 10 Calibration Flowchart 26
3.3.4. Intel Stratix 10 Calibration Algorithms 27
3.4. Intel Stratix 10 EMIF IP Controller 29
3.4.1. Hard Memory Controller 29
3.4.2. Intel Stratix 10 Hard Memory Controller Rate Conversion Feature 34
3.5. Hardware Resource Sharing Among Multiple Intel Stratix 10 EMIFs 34
3.5.1. I/O SSM Sharing 35
3.5.2. I/O Bank Sharing 35
3.5.3. PLL Reference Clock Sharing 36
3.5.4. Core Clock Network Sharing 37
3.6. User-requested Reset in Intel Stratix 10 EMIF IP 37
3.7. Intel Stratix 10 EMIF for Hard Processor Subsystem 39
3.7.1. Restrictions on I/O Bank Usage for Intel Stratix 10 EMIF IP with HPS 40
3.7.2. Using the Legacy EMIF Debug Toolkit with Intel Stratix 10 HPS Interfaces 44
3.7.3. HPS EMIF Simulation 44
3.8. Intel Stratix 10 EMIF Ping Pong PHY 45
3.8.1. Intel Stratix 10 Ping Pong PHY Feature Description 45
3.8.2. Intel Stratix 10 Ping Pong PHY Architecture 46
3.8.3. Intel Stratix 10 Ping Pong PHY Limitations 48
3.8.4. Intel Stratix 10 Ping Pong PHY Calibration 50
3.8.5. Using the Ping Pong PHY 50
3.8.6. Ping Pong PHY Simulation Example Design 50

4. Intel Stratix 10 EMIF IP End-User Signals 52
4.1. Interface and Signal Descriptions 52
4.1.1. Intel Stratix 10 EMIF IP Interfaces for DDR3 52

4.1.2. Intel Stratix 10 EMIF IP Interfaces for DDR4....64
4.1.3. Intel Stratix 10 EMIF IP Interfaces for QDR II/II+/II+ Xtreme....77
4.1.4. Intel Stratix 10 EMIF IP Interfaces for QDR-IV....84
4.1.5. Intel Stratix 10 EMIF IP Interfaces for RLDRAM 3....93
4.2. AFI Signals....101
4.2.1. AFI Clock and Reset Signals....102
4.2.2. AFI Address and Command Signals....102
4.2.3. AFI Write Data Signals....103
4.2.4. AFI Read Data Signals....104
4.2.5. AFI Calibration Status Signals....104
4.2.6. AFI Shadow Register Management Signals....105
4.3. AFI 4.0 Timing Diagrams....106
4.3.1. AFI Address and Command Timing Diagrams....106
4.3.2. AFI Write Sequence Timing Diagrams....109
4.3.3. AFI Read Sequence Timing Diagrams....117
4.3.4. AFI Calibration Status Timing Diagram....119
4.4. Intel Stratix 10 Memory Mapped Register (MMR) Tables....120
4.4.1. ctrlcfg0....121
4.4.2. ctrlcfg1....121
4.4.3. dramtiming0....123
4.4.4. caltiming0....123
4.4.5. caltiming1....123
4.4.6. caltiming2....123
4.4.7. caltiming3....124
4.4.8. caltiming4....124
4.4.9. caltiming9....125
4.4.10. dramaddrw....125
4.4.11. sideband0....125
4.4.12. sideband1....125
4.4.13. sideband4....126
4.4.14. sideband6....126
4.4.15. sideband7....126
4.4.16. sideband9....126
4.4.17. sideband11....127
4.4.18. sideband12....127
4.4.19. sideband13....127
4.4.20. sideband14....128
4.4.21. dramsts....128
4.4.22. niosreserve0....129
4.4.23. niosreserve1....129
4.4.24. sideband16....129
4.4.25. ecc3: ECC Error and Interrupt Configuration....129
4.4.26. ecc4: Status and Error Information....130
4.4.27. ecc5: Address of Most Recent SBE/DBE....131
4.4.28. ecc6: Address of Most Recent Correction Command Dropped....131
4.4.29. ecc7: Extension for Address of Most Recent SBE/DBE....132
4.4.30. ecc8: Extension for Address of Most Recent Correction Command Dropped...132

5. Intel Stratix 10 EMIF – Simulating Memory IP....133
5.1. Simulation Options....134
5.2. HPS EMIF Simulation....134

5.3. Simulation Walkthrough....135
5.3.1. Calibration Modes....135
5.3.2. Abstract PHY Simulation....136
5.3.3. Simulation Scripts....137
5.3.4. Functional Simulation with Verilog HDL....137
5.3.5. Functional Simulation with VHDL....138
5.3.6. Simulating the Design Example....138

6. Intel Stratix 10 EMIF IP for DDR3....142
6.1. Parameter Descriptions....142
6.1.1. Intel Stratix 10 EMIF IP DDR3 Parameters: General....142
6.1.2. Intel Stratix 10 EMIF IP DDR3 Parameters: FPGA I/O....144
6.1.3. Intel Stratix 10 EMIF IP DDR3 Parameters: Memory....146
6.1.4. Intel Stratix 10 EMIF IP DDR3 Parameters: Mem I/O....148
6.1.5. Intel Stratix 10 EMIF IP DDR3 Parameters: Mem Timing....148
6.1.6. Intel Stratix 10 EMIF IP DDR3 Parameters: Board....150
6.1.7. Intel Stratix 10 EMIF IP DDR3 Parameters: Controller....152
6.1.8. Intel Stratix 10 EMIF IP DDR3 Parameters: Diagnostics....154
6.1.9. Intel Stratix 10 EMIF IP DDR3 Parameters: Example Designs....157
6.2. Register Map IP-XACT Support for Intel Stratix 10 EMIF DDR3 IP....158
6.3. Board Skew Equations....159
6.3.1. Equations for DDR3 Board Skew Parameters....159
6.4. Pin and Resource Planning....161
6.4.1. Interface Pins....161
6.4.2. FPGA Resources....163
6.4.3. Pin Guidelines for Intel Stratix 10 EMIF IP....164
6.5. DDR3 Board Design Guidelines....177
6.5.1. Terminations and Slew Rates with Intel Stratix 10 Devices....178
6.5.2. Channel Signal Integrity Measurement....179
6.5.3. Layout Approach....183
6.5.4. Design Layout Guidelines....184
6.5.5. Package Deskew....193

7. Intel Stratix 10 EMIF IP for DDR4....197
7.1. Parameter Descriptions....197
7.1.1. Intel Stratix 10 EMIF IP DDR4 Parameters: General....197
7.1.2. Intel Stratix 10 EMIF IP DDR4 Parameters: FPGA I/O....199
7.1.3. Intel Stratix 10 EMIF IP DDR4 Parameters: Memory....201
7.1.4. Intel Stratix 10 EMIF IP DDR4 Parameters: Mem I/O....204
7.1.5. Intel Stratix 10 EMIF IP DDR4 Parameters: Mem Timing....207
7.1.6. Intel Stratix 10 EMIF IP DDR4 Parameters: Board....209
7.1.7. Intel Stratix 10 EMIF IP DDR4 Parameters: Controller....211
7.1.8. Intel Stratix 10 EMIF IP DDR4 Parameters: Diagnostics....214
7.1.9. Intel Stratix 10 EMIF IP DDR4 Parameters: Example Designs....216
7.2. Register Map IP-XACT Support for Intel Stratix 10 EMIF DDR4 IP....217
7.3. Board Skew Equations....218
7.3.1. Equations for DDR4 Board Skew Parameters....218
7.4. Pin and Resource Planning....219
7.4.1. Interface Pins....220
7.4.2. FPGA Resources....222
7.4.3. Pin Guidelines for Intel Stratix 10 EMIF IP....223

7.4.4. Resource Sharing Guidelines (Multiple Interfaces)....234
7.5. DDR4 Board Design Guidelines....235
7.5.1. Terminations and Slew Rates with Intel Stratix 10 Devices....236
7.5.2. Channel Signal Integrity Measurement....238
7.5.3. Layout Approach....242
7.5.4. Design Layout Guidelines....243
7.5.5. Package Deskew....255

8. Intel Stratix 10 EMIF IP for QDR II/II+/II+ Xtreme....260
8.1. Parameter Descriptions....260
8.1.1. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: General....260
8.1.2. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: FPGA I/O....262
8.1.3. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Memory....264
8.1.4. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Mem Timing....264
8.1.5. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Board....265
8.1.6. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Controller....267
8.1.7. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Diagnostics....267
8.1.8. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Example Designs....269
8.2. Board Skew Equations....270
8.2.1. Equations for QDRII, QDRII+, and QDRII+ Xtreme Board Skew Parameters....270
8.3. Pin and Resource Planning....271
8.3.1. Interface Pins....271
8.4. QDR II/II+/II+ Xtreme Board Design Guidelines....282
8.4.1. QDR II SRAM Configurations....282
8.4.2. General Layout Guidelines....284
8.4.3. QDR II Layout Guidelines....285
8.4.4. QDR II SRAM Layout Approach....286
8.4.5. Package Deskew....287
8.4.6. Package Migration....287
8.4.7. Slew Rates....288

9. Intel Stratix 10 EMIF IP for QDR-IV....290
9.1. Parameter Descriptions....290
9.1.1. Intel Stratix 10 EMIF IP QDR-IV Parameters: General....290
9.1.2. Intel Stratix 10 EMIF IP QDR-IV Parameters: FPGA I/O....292
9.1.3. Intel Stratix 10 EMIF IP QDR-IV Parameters: Memory....294
9.1.4. Intel Stratix 10 EMIF IP QDR-IV Parameters: Mem Timing....295
9.1.5. Intel Stratix 10 EMIF IP QDR-IV Parameters: Board....295
9.1.6. Intel Stratix 10 EMIF IP QDR-IV Parameters: Controller....297
9.1.7. Intel Stratix 10 EMIF IP QDR-IV Parameters: Diagnostics....298
9.1.8. Intel Stratix 10 EMIF IP QDR-IV Parameters: Example Designs....299
9.2. Board Skew Equations....301
9.2.1. Equations for QDR-IV Board Skew Parameters....301
9.3. Pin and Resource Planning....301
9.3.1. Interface Pins....302
9.4. QDR-IV Board Design Guidelines....312
9.4.1. QDR-IV Layout Approach....313
9.4.2. General Layout Guidelines....314
9.4.3. QDR-IV Layout Guidelines....314
9.4.4. Package Deskew....316
9.4.5. Package Migration....316

9.4.6. Slew Rates 317

10. Intel Stratix 10 EMIF IP for RLDRAM 3 318

10.1. Parameter Descriptions 318
10.1.1. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: General 318
10.1.2. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: FPGA I/O 320
10.1.3. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Memory 322
10.1.4. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Mem Timing 322
10.1.5. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Board 323
10.1.6. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Diagnostics 325
10.1.7. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Example Designs 327
10.2. Board Skew Equations 328
10.2.1. Equations for RLDRAM 3 Board Skew Parameters 328
10.3. Pin and Resource Planning 329
10.3.1. Interface Pins 329
10.4. RLDRAM 3 Board Design Guidelines 340
10.4.1. RLDRAM 3 Configurations 340
10.4.2. General Layout Guidelines 342
10.4.3. RLDRAM 3 Layout Guidelines 343
10.4.4. Layout Approach 345
10.4.5. Package Deskew 346
10.4.6. Package Migration 346
10.4.7. Slew Rates 347

11. Intel Stratix 10 EMIF IP Timing Closure 348

11.1. Timing Closure 348
11.1.1. Timing Analysis 349
11.2. Timing Report DDR 354
11.3. Optimizing Timing 355
11.4. Early I/O Timing Estimation 357
11.4.1. Performing Early I/O Timing Analysis 357

12. Optimizing Controller Performance 359

12.1. Interface Standard 359
12.2. Bank Management Efficiency 360
12.3. Data Transfer 362
12.4. Improving Controller Efficiency 363
12.4.1. Auto-Precharge Commands 364
12.4.2. Latency 366
12.4.3. Calibration 368
12.4.4. Bank Interleaving 368
12.4.5. Additive Latency and Bank Interleaving 370
12.4.6. User-Controlled Refresh 371
12.4.7. Frequency of Operation 373
12.4.8. Series of Reads or Writes 373
12.4.9. Data Reordering 373
12.4.10. Starvation Control 374
12.4.11. Command Reordering 374
12.4.12. Bandwidth 376
12.4.13. Enable Command Priority Control 376

13. Intel Stratix 10 EMIF IP Debugging..... 378
13.1. Interface Configuration Performance Issues..... 378
13.1.1. Interface Configuration Bottleneck and Efficiency Issues..... 379
13.2. Functional Issue Evaluation.....379
13.2.1. Intel IP Memory Model..... 380
13.2.2. Vendor Memory Model.....380
13.2.3. Transcript Window Messages.....380
13.2.4. Modifying the Example Driver to Replicate the Failure.....382
13.3. Timing Issue Characteristics..... 383
13.3.1. Evaluating FPGA Timing Issues.....383
13.3.2. Evaluating External Memory Interface Timing Issues..... 384
13.4. Verifying Memory IP Using the Signal Tap II Logic Analyzer..... 385
13.4.1. Signals to Monitor with the Signal Tap II Logic Analyzer.....386
13.5. Hardware Debugging Guidelines.....387
13.5.1. Create a Simplified Design that Demonstrates the Same Issue..... 387
13.5.2. Measure Power Distribution Network..... 388
13.5.3. Measure Signal Integrity and Setup and Hold Margin.....388
13.5.4. Vary Voltage..... 388
13.5.5. Operate at a Lower Speed..... 388
13.5.6. Determine Whether the Issue Exists in Previous Versions of Software..... 388
13.5.7. Determine Whether the Issue Exists in the Current Version of Software..... 389
13.5.8. Try A Different PCB.....389
13.5.9. Try Other Configurations.....390
13.5.10. Debugging Checklist.....390
13.6. Categorizing Hardware Issues..... 391
13.6.1. Signal Integrity Issues..... 391
13.6.2. Hardware and Calibration Issues.....393
13.7. Debugging Intel Stratix 10 EMIF IP..... 394
13.7.1. Debugging With the Legacy External Memory Interface Debug Toolkit.....395
13.7.2. Debugging with the External Memory Interface Unified Calibration Debug Toolkit..... 406
13.7.3. On-Chip Debug Port for Intel Stratix 10 EMIF IP.....429
13.7.4. Legacy Efficiency Monitor and Protocol Checker..... 431
13.7.5. New Efficiency Monitor..... 437
13.8. Using the Default Traffic Generator..... 444
13.8.1. Reading the Default Traffic Generator Status..... 445
13.8.2. Running Infinite Traffic using the Default Traffic Generator..... 447
13.8.3. Changing the Reset Trigger of the Default Traffic Generator.....447
13.8.4. Observing Generated Traffic with Signal Tap.....447
13.9. Using the Configurable Traffic Generator (TG2) 448
13.9.1. Enabling the Traffic Generator in a Design Example.....448
13.9.2. Traffic Generator Block Description.....449
13.9.3. Default Traffic Pattern..... 450
13.9.4. Configuration and Status Registers..... 450
13.9.5. User Pattern..... 458
13.9.6. Traffic Generator Status.....473
13.9.7. Starting Traffic with the Traffic Generator.....475
13.9.8. Traffic Generator Configuration User Interface..... 476
13.9.9. Examples of Configuring the TG2 Traffic Generator.....486

14. External Memory Interfaces Intel Stratix 10 FPGA IP User Guide Archives..... 490

15. Document Revision History for External Memory Interfaces Intel Stratix 10 FPGA IP User Guide..... 491

1. Release Information

IP versions are the same as the Intel® Quartus® Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

The IP version (X.Y.Z) number may change from one Intel Quartus Prime software version to another. A change in:

- X indicates a major revision of the IP. If you update your Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 1.

Item	Description
IP Version	19.2.4
Intel Quartus Prime	21.3
Release Date	2021.10.04

Related Information

External Memory Interfaces Intel Stratix 10 FPGA IP Core Release Notes

2. External Memory Interfaces Intel Stratix® 10 FPGA IP Introduction

Intel's fast, efficient, and low-latency external memory interface (EMIF) intellectual property (IP) cores easily interface with today's higher speed memory devices.

You can easily implement the EMIF IP core functions through the Intel Quartus Prime software. The Intel Quartus Prime software also provides external memory toolkits that help you test the implementation of the IP in the FPGA.

The *External Memory Interfaces Intel Stratix® 10 FPGA IP* (referred to hereafter as the *Intel Stratix 10 EMIF IP*) provides the following components:

- A physical layer interface (PHY) which builds the data path and manages timing transfers between the FPGA and the memory device.
- A memory controller which implements all the memory commands and protocol-level requirements.

For information on the maximum speeds supported by the external memory interface IP, refer to the External Memory Interface Spec Estimator.

Intel Stratix 10 EMIF IP Protocol and Feature Support

- Supports DDR4, DDR3, and DDR3L protocols with hard memory controller and hard PHY.
- Supports QDR-IV, QDR II + Xtreme, QDR II +, and QDR II using soft memory controller and hard PHY.
- Supports RLDRAM 3 using third-party soft controller.
- Supports UDIMM, RDIMM, LRDIMM and SODIMM memory devices.
- Supports 3D Stacked Die for DDR4 devices.
- Supports up to 4 physical ranks.
- Supports Ping Pong PHY mode, allowing two memory controllers to share command, address, and control pins.
- Supports error correction code (ECC) for both hard memory controller and soft memory controller.

Related Information

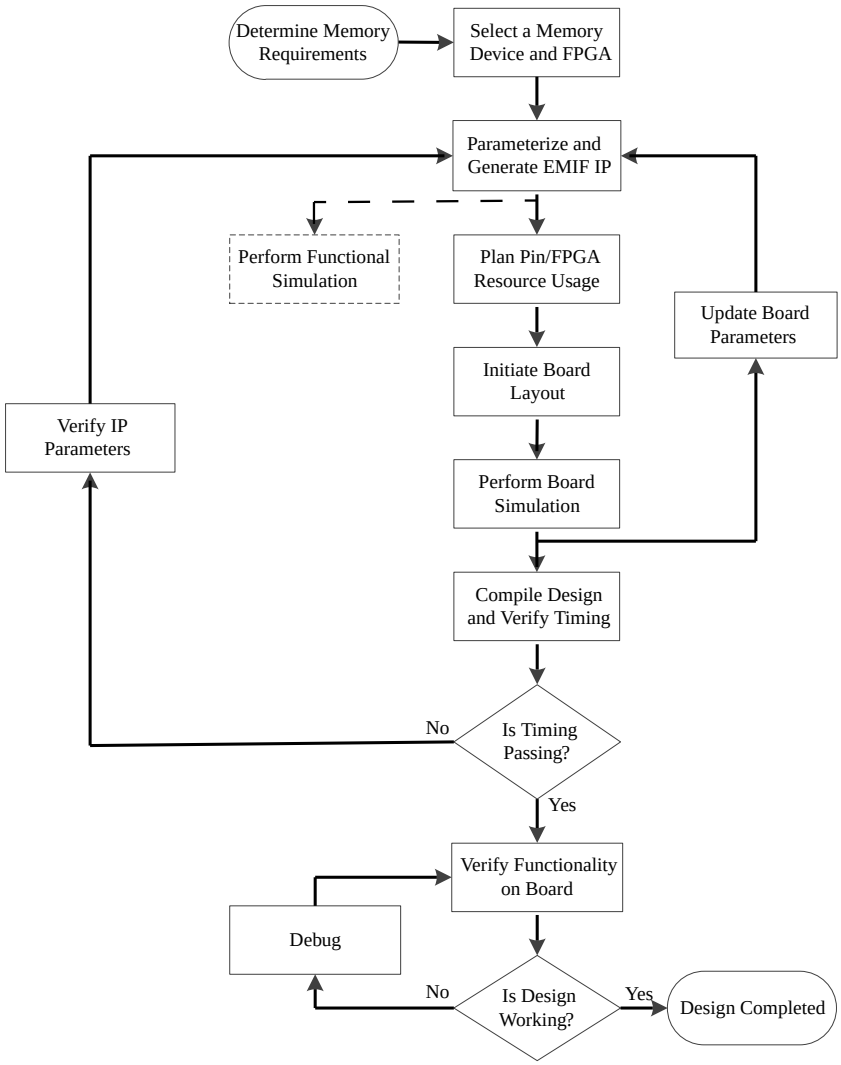
- Intel FPGA IP for External Memory Interfaces - Support Center
- Intel Stratix 10 General Purpose I/O User Guide

2.1. Intel Stratix 10 EMIF IP Design Flow

Intel recommends creating an example top-level file with the desired pin outs and all interface IPs instantiated. This enables the Intel Quartus Prime software to validate the design and resource allocation before PCB and schematic sign off.

The following figure shows the design flow to provide the fastest out-of-the-box experience with the EMIF IP.

Figure 1. EMIF IP Design Flow



Related Information

- Introduction to Intel FPGA IP Cores
- Generating a Combined Simulator Setup Script
- Project Management Best Practices

2.2. Intel Stratix 10 EMIF IP Design Checklist

Refer to the following checklist as a quick reference for information about each step in the EMIF design flow.

Table 2. EMIF Design Checklist

Design Step	Description	Resources
Select an FPGA	Not all Intel FPGAs support all memory types and configurations. To help with the FPGA selection process, refer to these resources.	• Intel FPGA Product Selector • External Memory Interface Device Selector • External Memory Interface Spec Estimator
Parameterize the IP	Correct IP parameterization is important for good EMIF IP operation. These resources define the memory parameters during IP generation.	• DDR3 Parameter Descriptions • DDR4 Parameter Descriptions • QDR II/II+/II+ Xtreme Parameter Descriptions • QDR-IV Parameter Descriptions • RLDRAM 3 Parameter Descriptions
Generate initial IP and example design	After you have parameterized the EMIF IP, you can generate the IP, along with an optional example design. Refer to the Quick-Start Guide for a walkthrough of this process.	• Design Example Quick Start Guide • Design Example Description
Perform functional simulation	Simulation of the EMIF design helps to determine correct operation. These resources explain how to perform simulation and what differences exist between simulation and hardware implementation.	• Design Example Quick Start Guide • Simulating Memory IP
Make pin assignments	For guidance on pin placement, refer to these resources.	• DDR3 Parameter Descriptions • DDR4 Parameter Descriptions • QDR II/II+/II+ Xtreme Parameter Descriptions • QDR-IV Parameter Descriptions • RLDRAM 3 Parameter Descriptions
Perform board simulation	Board simulation helps determine optimal settings for signal integrity, drive strength, as well as sufficient timing margins and eye openings. For guidance on board simulation, refer to these resources.	• DDR3 Board Design Guidelines • DDR4 Board Design Guidelines • QDR II/II+/II+ Xtreme Board Design Guidelines • QDR-IV Board Design Guidelines • RLDRAM 3 Board Design Guidelines • Board Skew Parameter Tool
Update board parameters in the IP	Board simulation is important to determine optimal settings for signal integrity, drive strength, and sufficient timing margins and eye openings. For guidance on board simulation refer to the mentioned resources.	• DDR3 Board Design Guidelines • DDR4 Board Design Guidelines • QDR II/II+/II+ Xtreme Board Design Guidelines • QDR-IV Board Design Guidelines • RLDRAM 3 Board Design Guidelines • Board Skew Parameter Tool

continued...

Design Step	Description	Resources
Verify timing closure	For information regarding compilation, system-level timing closure and timing reports refer to the Timing Closure section of this User Guide.	• Timing Closure
Run the design on hardware	For instructions on how to program a FPGA refer to the Quick-Start Guide section of this User Guide.	• Design Example Quick Start Guide
Debug issues with preceding steps	Operational problems can generally be attributed to one of the following: interface configuration, pin/resource planning, signal integrity, or timing. These resources contain information on typical debug procedures and available tools to help diagnose hardware issues.	• Debugging • External Memory Interfaces Support Center

3. Intel Stratix 10 EMIF IP Product Architecture

This chapter describes the Intel Stratix 10 product architecture.

3.1. Intel Stratix 10 EMIF Architecture: Introduction

The Intel Stratix 10 EMIF architecture contains many new hardware features designed to meet the high-speed requirements of emerging memory protocols, while consuming the smallest amount of core logic area and power.

The following are key hardware features of the Intel Stratix 10 EMIF architecture:

Hard Sequencer

The sequencer employs a hard Nios II processor, and can perform memory calibration for a wide range of protocols. You can share the sequencer among multiple memory interfaces of the same or different protocols.

Note: You cannot use the hard Nios II processor for any user applications after calibration is complete.

Hard PHY

The PHY circuitry in Intel Stratix 10 devices is hardened in the silicon, which simplifies the challenges of achieving timing closure and minimizing power consumption.

Hard Memory Controller

The hard memory controller reduces latency and minimizes core logic consumption in the external memory interface. The hard memory controller supports the DDR3 and DDR4 memory protocols.

PHY-Only Mode

Protocols that use a hard controller (DDR3, DDR4, and RLDRAM 3), provide a PHY-only option, which generates only the PHY and sequencer, but not the controller. This PHY-only mode provides a mechanism by which to integrate your own custom soft controller.

High-Speed PHY Clock Tree

Dedicated high speed PHY clock networks clock the I/O buffers in Intel Stratix 10 EMIF IP. The PHY clock trees exhibit low jitter and low duty cycle distortion, maximizing the data valid window.

Automatic Clock Phase Alignment

Automatic clock phase alignment circuitry dynamically adjusts the clock phase of core clock networks to match the clock phase of the PHY clock networks. The clock phase alignment circuitry minimizes clock skew that can complicate timing closure in transfers between the FPGA core and the periphery.

Resource Sharing

The Intel Stratix 10 architecture simplifies resource sharing between memory interfaces. Resources such as the OCT calibration block, PLL reference clock pin, and core clock can be shared. The hard Nios processor in the I/O subsystem manager (I/O SSM) must be shared across all interfaces in a column.

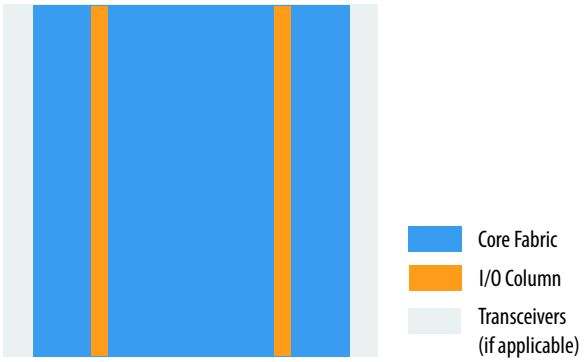
Related Information

- External Memory Interface Spec Estimator
- Introduction to Intel FPGA IP Cores
- Generating a Combined Simulator Setup Script
- Project Management Best Practices

3.1.1. Intel Stratix 10 EMIF Architecture: I/O Subsystem

Depending on the Intel Stratix 10 device, the I/O subsystem consists of either two or three columns inside the core.

Figure 2. Stratix 10 I/O Subsystem



The I/O subsystem provides the following features:

- General-purpose I/O registers and I/O buffers
- On-chip termination control (OCT)
- I/O PLLs for external memory interfaces and user logic
- Low-voltage differential signaling (LVDS)
- External memory interface components, as follows:
  - Hard memory controller
  - Hard PHY
  - Hard Nios processor and calibration logic
  - DLL

3.1.2. Intel Stratix 10 EMIF Architecture: I/O Column

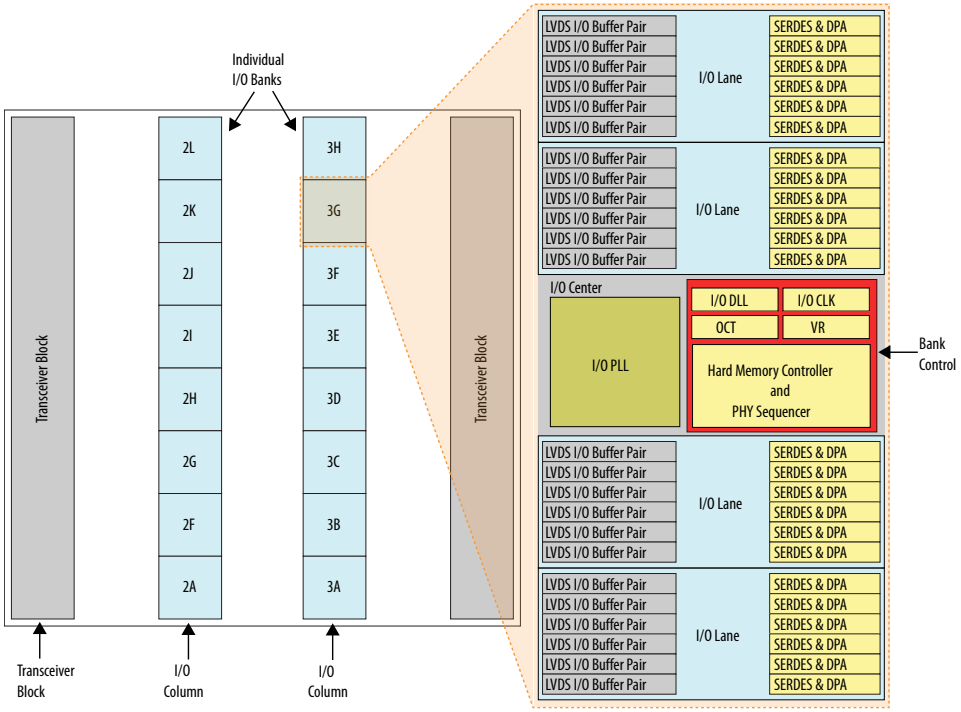
Most Intel Stratix 10 devices have two I/O columns, which contain the hardware related to external memory interfaces.

Each I/O column contains the following major parts:

- A hardened Nios processor with dedicated memory. This Nios block is referred to as the I/O SSM.
- Up to 13 I/O banks. Each I/O bank contains the hardware necessary for an external memory interface.

Note: Not all I/O banks on 1ST/SX/SG040 devices support external memory interfaces. On 1ST040 devices, banks 3A and 3D cannot be used for EMIF. On 1SX/SG040 devices, banks 3A, 3C, and 3D cannot be used for EMIF.

Figure 3. I/O Column



3.1.3. Intel Stratix 10 EMIF Architecture: I/O SSM

Each column includes one I/O subsystem manager (I/O SSM), which contains a hardened Nios II processor with dedicated memory. The I/O SSM is responsible for calibration of all the EMIFs in the column.

The I/O SSM includes dedicated memory which stores both the calibration algorithm and calibration run-time data. The hardened Nios II processor and the dedicated memory can be used only by an external memory interface, and cannot be employed for any other use. The I/O SSM can interface with soft logic, such as the debug toolkit, via an Avalon-MM bus.

The I/O SSM is clocked by an on-die oscillator, and therefore does not consume a PLL.

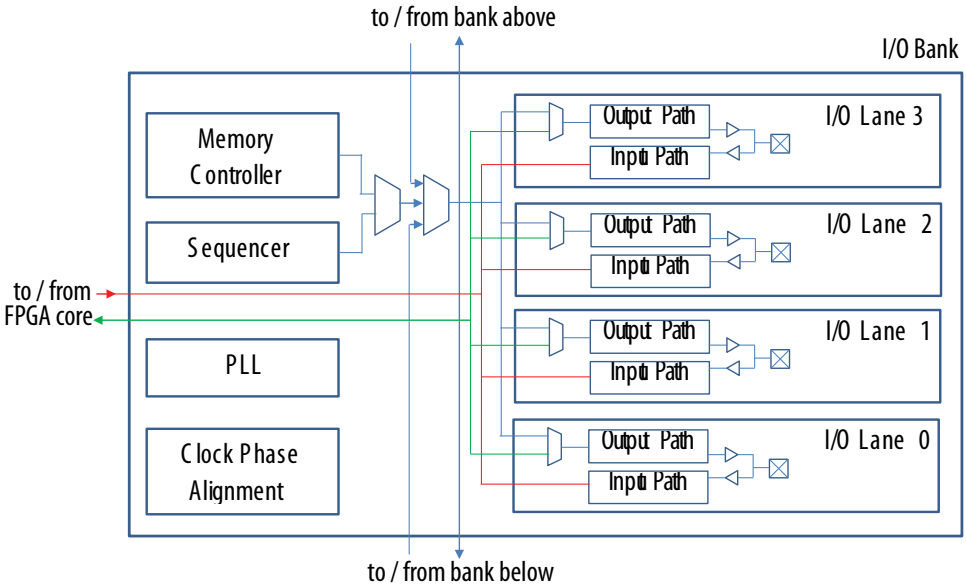
3.1.4. Intel Stratix 10 EMIF Architecture: I/O Bank

A single I/O bank contains all the hardware needed to build an external memory interface. Each I/O column contains up to 13 I/O banks; the exact number of banks depends on device size and pin package. You can make a wider interface by connecting multiple banks together.

Each I/O bank resides in an I/O column, and contains the following components:

- Hard memory controller
- Sequencer components
- PLL and PHY clock trees
- DLL
- Input DQS clock trees
- 48 pins, organized into four I/O lanes of 12 pins each

Figure 4. I/O Bank Architecture in Intel Stratix 10 Devices



I/O Bank Usage

The pins in an I/O bank can serve as address and command pins, data pins, or clock and strobe pins for an external memory interface. You can implement a narrow interface, such as a DDR3 or DDR4 x8 interface, with only a single I/O bank. A wider interface of up to 72 bits can be implemented by configuring multiple adjacent banks in a multi-bank interface. Any pins in a bank which are not used by the EMIF IP can serve as general-purpose I/O pins of uncalibrated I/O standard with the same voltage settings.

Every I/O bank includes a hard memory controller which you can configure for DDR3 or DDR4. In a multi-bank interface, only the controller of one bank is active; controllers in the remaining banks are turned off to conserve power.

To use a multi-bank Intel Stratix 10 EMIF interface, you must observe the following rules:

- Designate one bank as the address and command bank.
- The address and command bank must contain all the address and command pins.
- The locations of individual address and command pins within the address and command bank must adhere to the pin map defined in the pin table— regardless of whether you use the hard memory controller or not.
- If you do use the hard memory controller, the address and command bank contains the active hard controller.

All the I/O banks in a column are capable of functioning as the address and command bank. However, for minimal latency, you should select the center-most bank of the interface as the address and command bank.

Note: Not all I/O banks on 1ST/SX/SG040 devices support external memory interfaces. On 1ST040 devices, banks 3A and 3D cannot be used for EMIF. On 1SX/SG040 devices, banks 3A, 3C, and 3D cannot be used for EMIF.

3.1.5. Intel Stratix 10 EMIF Architecture: I/O Lane

An I/O bank contains 48 I/O pins, organized into four I/O lanes of 12 pins each.

Each I/O lane can implement one x8/x9 read capture group (DQS group), with two pins functioning as the read capture clock/strobe pair (DQS/DQS#), and up to 10 pins functioning as data pins (DQ and DM pins). To implement x18 and x36 groups, you can use multiple lanes within the same bank.

It is also possible to implement a pair of x4 groups in a lane. In this case, four pins function as clock/strobe pair, and 8 pins function as data pins. DM is not available for x4 groups. There must be an even number of x4 groups for each interface.

For x4 groups, DQS0 and DQS1 must be placed in the same I/O lane as a pair. Similarly, DQS2 and DQS3 must be paired. In general, DQS(x) and DQS(x+1) must be paired in the same I/O lane.

Table 3. Lanes Used Per Group

Group Size	Number of Lanes Used	Maximum Number of Data Pins per Group
x8 / x9	1	10
x18	2	22
x36	4	46
pair of x4	1	4 per group, 8 per lane

Figure 5. x4 Group

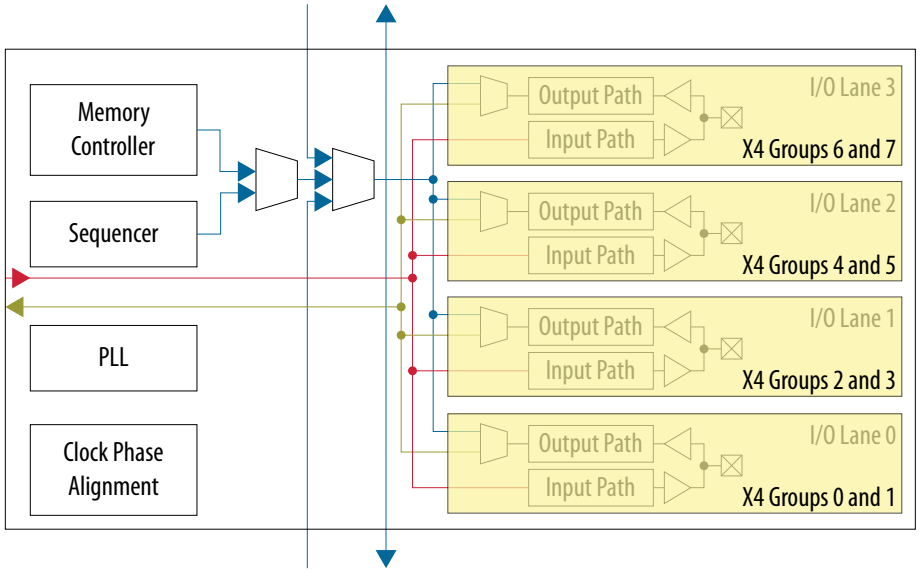


Figure 6. x8 Group

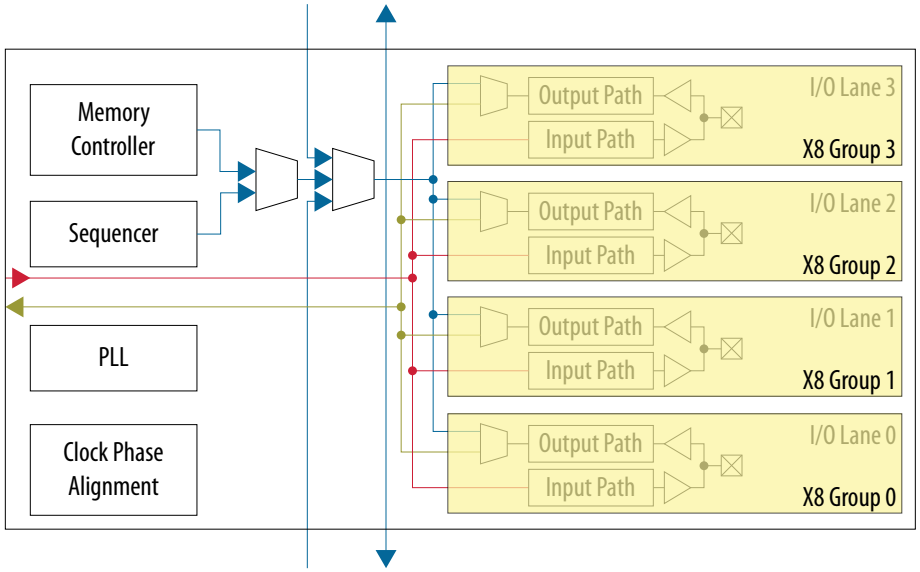


Figure 7. x18 Group

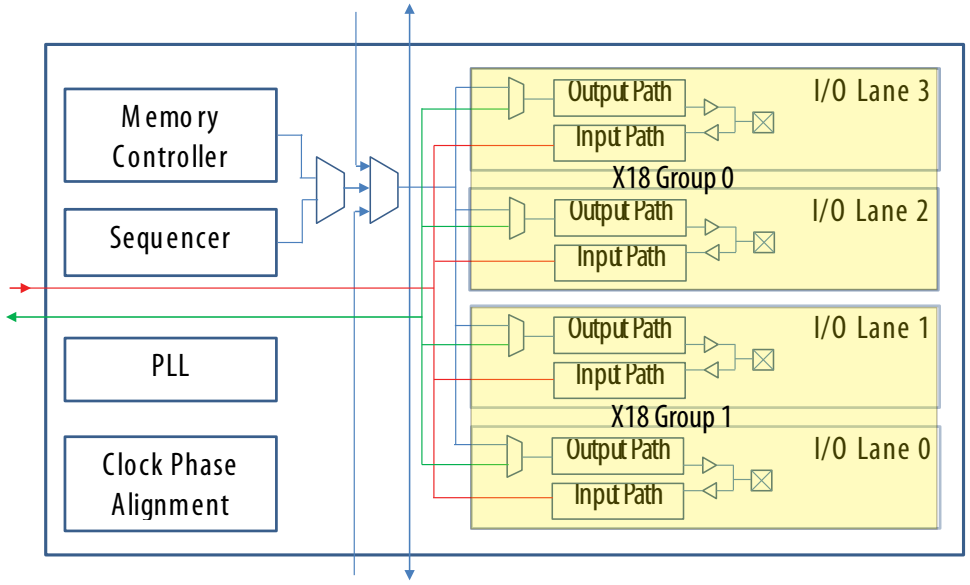
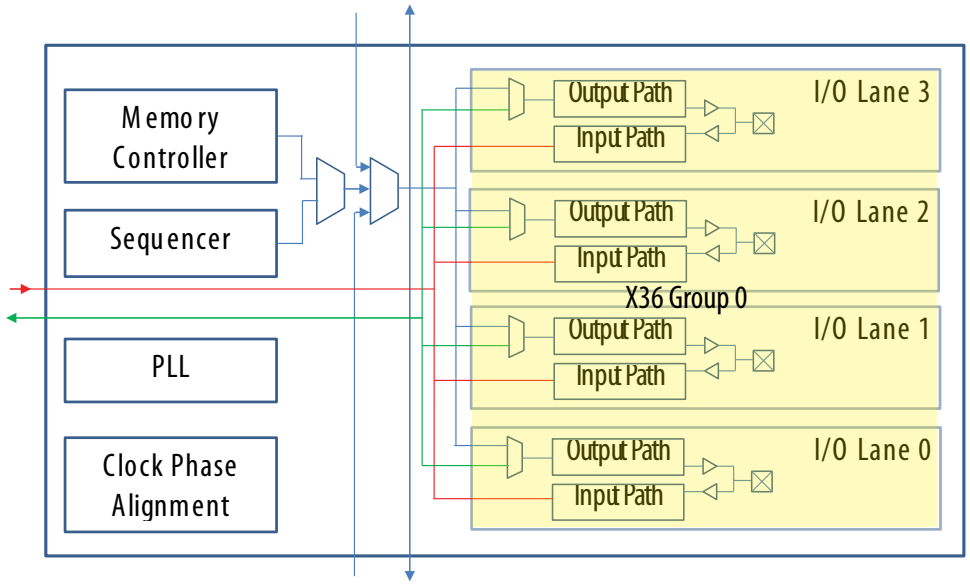


Figure 8. x36 Group



3.1.6. Intel Stratix 10 EMIF Architecture: Input DQS Clock Tree

The input DQS clock tree is a balanced clock network that distributes the read capture clock (such as CQ/CQ# or QK/QK# which are free-running read clocks) and strobe (such as DQS/DQS#) from the external memory device to the read capture registers inside the I/Os.

You can configure an input DQS clock tree in x4 mode, x8/x9 mode, x18 mode, or x36 mode.

Within every bank, only certain physical pins at specific locations can drive the input DQS clock trees. The pin locations that can drive the input DQS clock trees vary, depending on the size of the group.

Table 4. Pins Usable as Read Capture Clock / Strobe Pair

Group Size	Index of Lanes Spanned by Clock Tree	In-Bank Index of Pins Usable as Read Capture Clock / Strobe Pair	
		Positive Leg	Negative Leg
x4	0A	4	5
x4	0B	8	9
x4	1A	16	17
x4	1B	20	21
x4	2A	28	29
x4	2B	32	33
x4	3A	40	41
x4	3B	44	45
x8 / x9	0	4	5
x8 / x9	1	16	17
x8 / x9	2	28	29
x8 / x9	3	40	41
x18	0, 1	8	9
x18	2, 3	32	33
x36	0, 1, 2, 3	20	21

3.1.7. Intel Stratix 10 EMIF Architecture: PHY Clock Tree

Dedicated high-speed clock networks drive I/Os in Intel Stratix 10 EMIF. Each PHY clock network spans only one bank.

The relatively short span of the PHY clock trees results in low jitter and low duty-cycle distortion, maximizing the data valid window.

The PHY clock tree in Intel Stratix 10 devices can run as fast as 1.3 GHz. All Intel Stratix 10 external memory interfaces use the PHY clock trees.

3.1.8. Intel Stratix 10 EMIF Architecture: PLL Reference Clock Networks

Each I/O bank includes a PLL that can drive the PHY clock trees of that bank, through dedicated connections. In addition to supporting EMIF-specific functions, such PLLs can also serve as general-purpose PLLs for user logic.

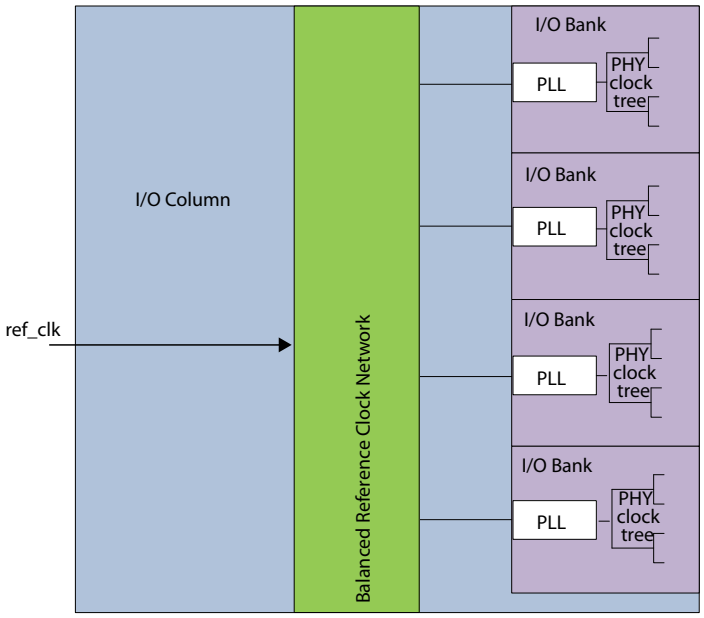
Intel Stratix 10 external memory interfaces that span multiple banks use the PLL in each bank. The Intel Stratix 10 architecture allows for relatively short PHY clock networks, reducing jitter and duty-cycle distortion.

The following mechanisms ensure that the clock outputs of individual PLLs in a multi-bank interface remain in phase:

- A single PLL reference clock source feeds all PLLs. The reference clock signal reaches the PLLs by a balanced PLL reference clock tree. The Intel Quartus Prime software automatically configures the PLL reference clock tree so that it spans the correct number of banks.
- The EMIF IP sets the PLL M and N values appropriately to maintain synchronization among the clock dividers across the PLLs. This requirement restricts the legal PLL reference clock frequencies for a given memory interface frequency and clock rate. The Stratix 10 EMIF IP parameter editor automatically calculates and displays the set of legal PLL reference clock frequencies. If you plan to use an on-board oscillator, you must ensure that its frequency matches the PLL reference clock frequency that you select from the displayed list. The correct M and N values of the PLLs are set automatically based on the PLL reference clock frequency that you select.

Note: The PLL reference clock pin may be placed in the address and command I/O bank or in a data I/O bank, there is no implication on timing. However, for debug flexibility, it is recommended to place the PLL reference clock in the address and command I/O bank.

Figure 9. PLL Balanced Reference Clock Tree



Related Information

Maximum Number of Interfaces on page 163

3.1.9. Intel Stratix 10 EMIF Architecture: Clock Phase Alignment

In Intel Stratix 10 external memory interfaces, a global clock network clocks registers inside the FPGA core, and the PHY clock network clocks registers inside the FPGA periphery. Clock phase alignment circuitry employs negative feedback to dynamically adjust the phase of the core clock signal to match the phase of the PHY clock signal.

The clock phase alignment feature effectively eliminates the clock skew effect in all transfers between the core and the periphery, facilitating timing closure. All Stratix 10 external memory interfaces employ clock phase alignment circuitry.

Figure 10. Clock Phase Alignment Illustration

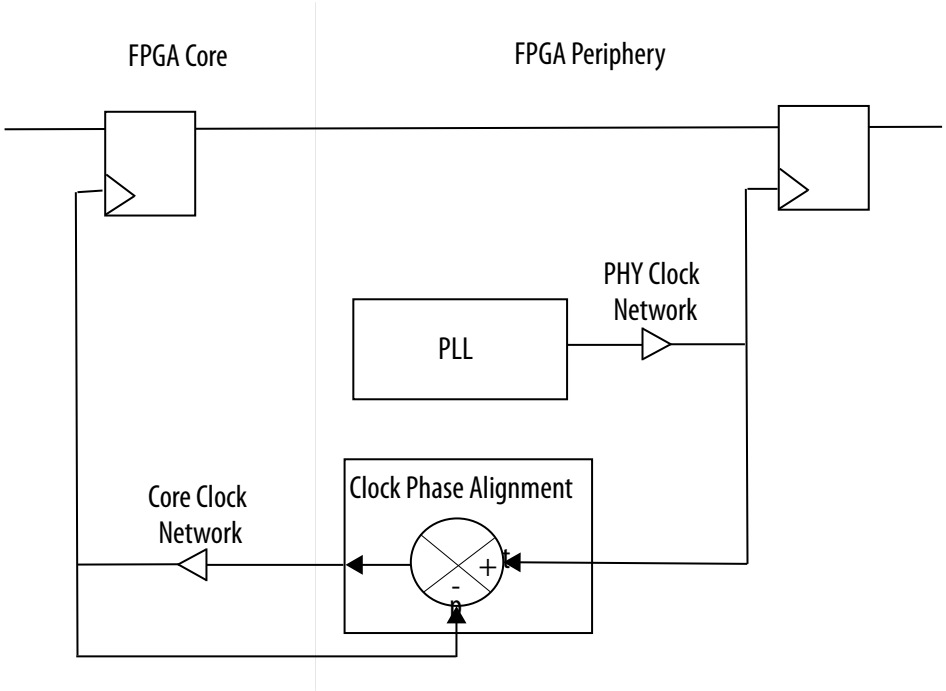
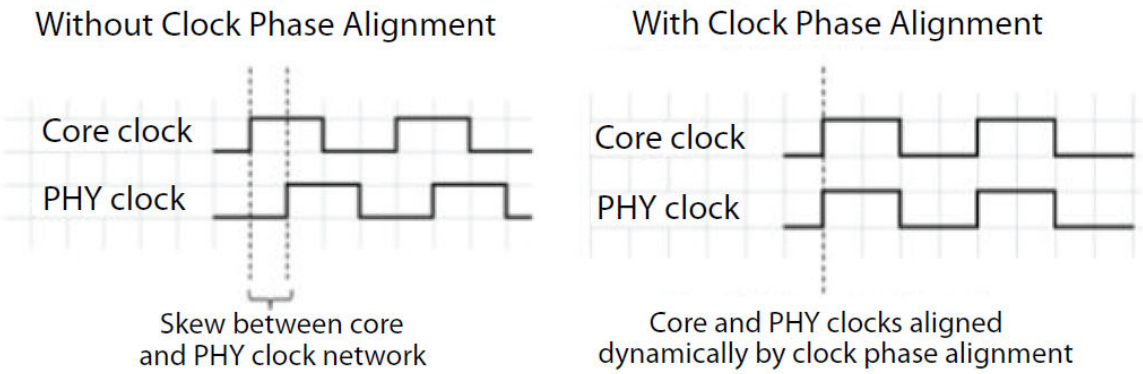


Figure 11. Effect of Clock Phase Alignment



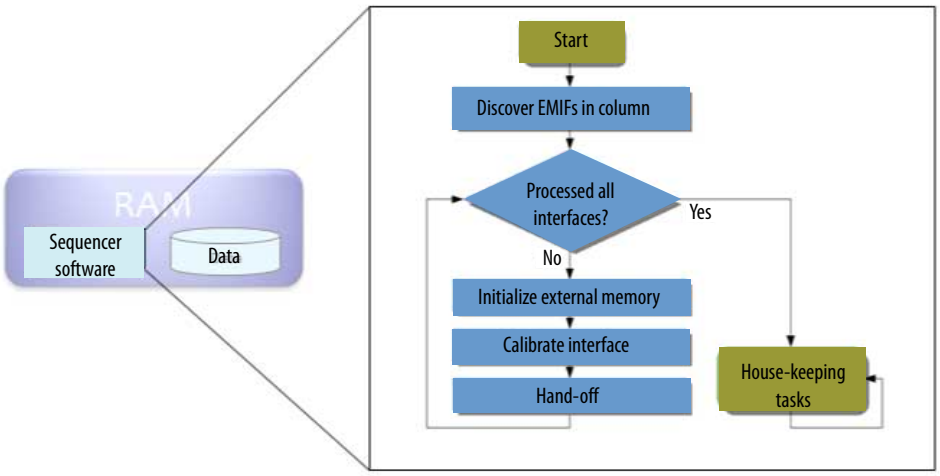
3.2. Intel Stratix 10 EMIF Sequencer

The Intel Stratix 10 EMIF sequencer is fully hardened in silicon, with executable code to handle protocols and topologies. Hardened RAM contains the calibration algorithm.

The Intel Stratix 10 EMIF sequencer is responsible for the following operations:

- Initializes memory devices.
- Calibrates the external memory interface.
- Governs the hand-off of control to the memory controller.
- Handles recalibration requests and debug requests.
- Handles all supported protocols and configurations.

Figure 12. Intel Stratix 10 EMIF Sequencer Operation



3.2.1. Intel Stratix 10 EMIF DQS Tracking

DQS tracking tracks read capture clock/strobe timing variation over time, for improved read capture I/O timing. This feature takes sufficient samples to confirm the variation and adjust the DQS-enable position to maintain adequate operating margins.

DQS tracking is enabled for QDRII/II+/II+ Xtreme, QDR-IV, and RLDRAM 3 protocols; it is not available for DDR3 and DDR4 protocols. For QDRII/II+/II+ Xtreme, QDR-IV, and RLDRAM 3, DQS tracking does not need a specific command to initiate tracking, because the read capture clock/strobe is free running. Tracking happens constantly and automatically when the circuitry is enabled.

3.3. Intel Stratix 10 EMIF Calibration

The calibration process compensates for skews and delays in the external memory interface.

The calibration process enables the system to compensate for the effects of factors such as the following:

- Timing and electrical constraints, such as setup/hold time and V_{ref} variations.
- Circuit board and package factors, such as skew, fly-by effects, and manufacturing variations.
- Environmental uncertainties, such as variations in voltage and temperature.
- The demanding effects of small margins associated with high-speed operation.

For a given external memory interface, calibration occurs in parallel for all DQS groups and I/O banks. For an I/O column containing multiple external memory interfaces, there is no particular calibration order in relation to the interfaces; however, for a given SRAM Object File (`.sof`), calibration always occurs in the same order.

Note: The calibration process is intended to maximize margins for robust EMIF operation; it cannot compensate for an inadequate PCB layout.

3.3.1. Intel Stratix 10 Calibration Stages

At a high level, the calibration routine consists of address and command calibration, read calibration, and write calibration.

The stages of calibration vary, depending on the protocol of the external memory interface.

Table 6. Calibration Stages by Protocol

Stage	DDR4	DDR3	RLDRAM 3	QDR-IV	QDR II/II+
Address and command					
Leveling	Yes	Yes	—	—	—
Deskew	Yes	—	—	Yes	—
Read					
DQSen	Yes	Yes	Yes	Yes	Yes
Deskew	Yes	Yes	Yes	Yes	Yes
VREF-In	Yes	—	—	Yes	—
LFIFO	Yes	Yes	Yes	Yes	Yes
Write					
Leveling	Yes	Yes	Yes	Yes	—
Deskew	Yes	Yes	Yes	Yes	Yes
VREF-Out	Yes	—	—	—	—

3.3.2. Intel Stratix 10 Calibration Stages Descriptions

The various stages of calibration perform address and command calibration, read calibration, and write calibration.

Address and Command Calibration

The goal of address and command calibration is to delay address and command signals as necessary to optimize the address and command window. This stage is not available for all protocols, and cannot compensate for a poorly implemented board design.

Address and command calibration consists of the following parts:

- Leveling calibration— Centers the CS# signal and the entire address and command bus, relative to the CK clock. This operation is available for DDR3 and DDR4 interfaces only.
- Deskew calibration— Provides per-bit deskew for the address and command bus (except CS#), relative to the CK clock. This operation is available for DDR4 and QDR-IV interfaces only.

Read Calibration

Read calibration consists of the following parts:

- DQSen calibration— Calibrates the timing of the read capture clock gating and ungating, so that the PHY can gate and ungate the read clock at precisely the correct time—if too early or too late, data corruption can occur. The algorithm for this stage varies, depending on the memory protocol.
- Deskew calibration— Performs per-bit deskew of read data relative to the read strobe or clock.
- VREF-In calibration— Calibrates the VREF level at the FPGA.
- LFIFO calibration: Normalizes differences in read delays between groups due to fly-by, skews, and other variables and uncertainties.

Write Calibration

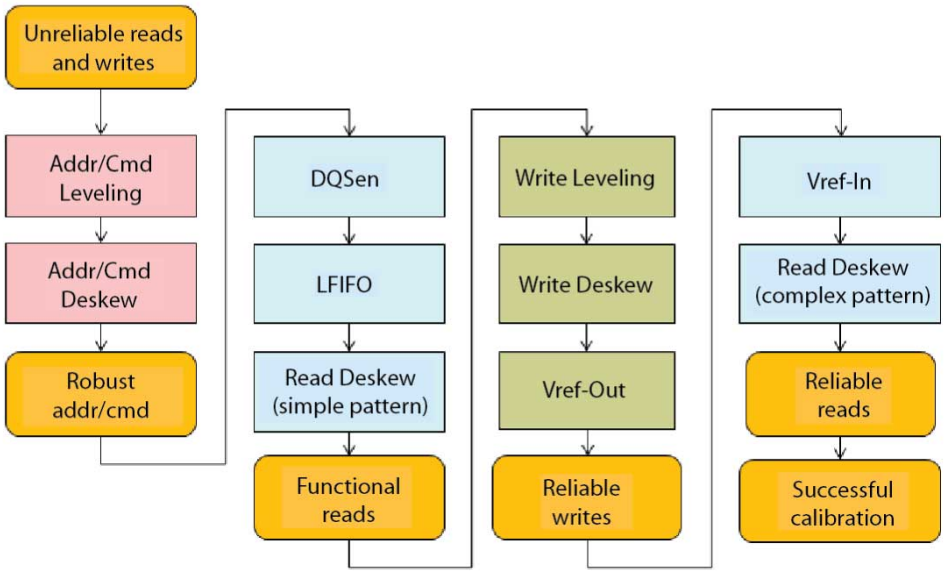
Write calibration consists of the following parts:

- Leveling calibration— Aligns the write strobe and clock to the memory clock, to compensate for skews, especially those associated with fly-by topology. The algorithm for this stage varies, depending on the memory protocol.
- Deskew calibration— Performs per-bit deskew of write data relative to the write strobe and clock.
- VREF-Out calibration— Calibrates the VREF level at the memory device.

3.3.3. Intel Stratix 10 Calibration Flowchart

The following flowchart illustrates the Intel Stratix 10 calibration flow.

Figure 13. Calibration Flowchart



3.3.4. Intel Stratix 10 Calibration Algorithms

The calibration algorithms sometimes vary, depending on the targeted memory protocol.

Address and Command Calibration

Address and command calibration consists of the following parts:

- Leveling calibration— (DDR3 and DDR4 only) Toggles the CS# and CAS# signals to send read commands while keeping other address and command signals constant. The algorithm monitors for incoming DQS signals, and if the DQS signal toggles, it indicates that the read commands have been accepted. The algorithm then repeats using different delay values, to find the optimal window.
- Deskew calibration— (DDR4 and QDR-IV only)
  - (DDR4) Uses the DDR4 address and command parity feature. The FPGA sends the address and command parity bit, and the DDR4 memory device responds with an alert signal if the parity bit is detected. The alert signal from the memory device tells the FPGA that the parity bit was received.

    Deskew calibration requires use of the PAR/ALERT# pins, so you should not omit these pins from your design. One limitation of deskew calibration is that it cannot deskew ODT and CKE pins.
  - (QDR-IV) Uses the QDR-IV loopback mode. The FPGA sends address and command signals, and the memory device sends back the address and command signals which it captures, via the read data pins. The returned signals indicate to the FPGA what the memory device has captured. Deskew calibration can deskew all synchronous address and command signals.

    Note: For more information about loopback mode, refer to your QDR-IV memory device data sheet.

Read Calibration

- DQSen calibration— (DDR3, DDR4, RLDRAM*x* and QDR*x*) DQSen calibration occurs before Read deskew, therefore only a single DQ bit is required to pass in order to achieve a successful read pass.
  - (DDR3 and DDR4) The DQSen calibration algorithm searches the DQS preamble using a hardware state machine. The algorithm sends many back-to-back reads with a one clock cycle gap between. The hardware state machine searches for the DQS gap while sweeping DQSen delay values. The algorithm then increments the VFIFO value, and repeats the process until a pattern is found. The process is then repeated for all other read DQS groups.
  - (RLDRAM*x* and QDR*x*) The DQSen calibration algorithm does not use a hardware state machine; rather, it calibrates cycle-level delays using software and subcycle delays using DQS tracking hardware. The algorithm requires good data in memory, and therefore relies on guaranteed writes. (Writing a burst of 0s to one location, and a burst of 1s to another; back-to-back reads from these two locations are used for read calibration.)

    The algorithm enables DQS tracking to calibrate the phase component of DQS enable, and then issues a guaranteed write, followed by back-to-back reads. The algorithm sweeps DQSen values cycle by cycle until the read operation succeeds. The process is then repeated for all other read groups.
- Deskew calibration— Read deskew calibration is performed before write leveling, and must be performed at least twice: once before write calibration, using simple data patterns from guaranteed writes, and again after write calibration, using complex data patterns.

  The deskew calibration algorithm performs a guaranteed write, and then sweeps `dqs_in` delay values from low to high, to find the right edge of the read window. The algorithm then sweeps `dq-in` delay values low to high, to find the left edge of the read window. Updated `dqs_in` and `dq_in` delay values are then applied to center the read window. The algorithm then repeats the process for all data pins.
- Vref-In calibration— Read `Vref-In` calibration begins by programming `Vref-In` with an arbitrary value. The algorithm then sweeps the `Vref-In` value from the starting value to both ends, and measures the read window for each value. The algorithm selects the `Vref-In` value which provides the maximum read window.
- LFIFO calibration— Read LFIFO calibration normalizes read delays between groups. The PHY must present all data to the controller as a single data bus. The LFIFO latency should be large enough for the slowest read data group, and large enough to allow proper synchronization across FIFOs.

Write Calibration

- Leveling calibration— Write leveling calibration aligns the write strobe and clock to the memory clock, to compensate for skews. In general, leveling calibration tries a variety of delay values to determine the edges of the write window, and then selects an appropriate value to center the window. The details of the algorithm vary, depending on the memory protocol.
  - (DDR*x*) Write leveling occurs before write deskew, therefore only one successful DQ bit is required to register a pass. Write leveling staggers the DQ bus to ensure that at least one DQ bit falls within the valid write window.
  - (RLDRAM*x*) Optimizes for the CK versus DK relationship.
  - (QDR-IV) Optimizes for the CK versus DK relationship. Is covered by address and command deskew using the loopback mode.
  - (QDR II/II+/Xtreme) The K clock is the only clock, therefore write leveling is not required.
- Deskew calibration— Performs per-bit deskew of write data relative to the write strobe and clock. Write deskew calibration does not change `dqs_out` delays; the write clock is aligned to the CK clock during write leveling.
- VREF-Out calibration— (DDR4) Calibrates the VREF level at the memory device. The VREF-Out calibration algorithm is similar to the VREF-In calibration algorithm.

3.4. Intel Stratix 10 EMIF IP Controller

3.4.1. Hard Memory Controller

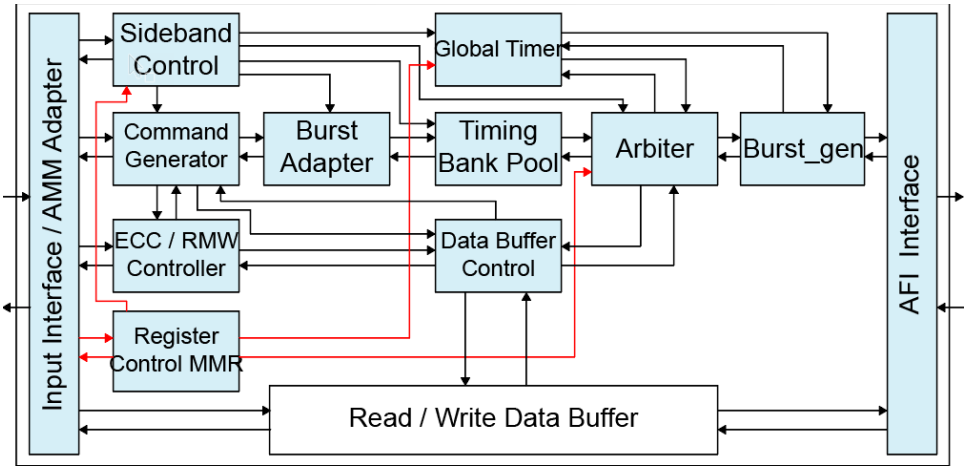
The Intel Stratix 10 hard memory controller is designed for high speed, high performance, high flexibility, and area efficiency. The Intel Stratix 10 hard memory controller supports DDR3 and DDR4 memory standards.

The hard memory controller implements efficient pipelining techniques and advanced dynamic command and data reordering algorithms to improve bandwidth usage and reduce latency, providing a high performance solution.

The controller architecture is modular and fits in a single I/O bank. The structure allows you to:

- Configure each I/O bank as either:
  - A control path that drives all the address and command pins for the memory interface.
  - A data path that drives up to 32 data pins for DDR-type interfaces.
- Place your memory controller in any location.
- Pack up multiple banks together to form memory interfaces of different widths up to 72 bits.
- Bypass the hard memory controller and use your own custom IP if required.

Figure 14. Hard Memory Controller Architecture



The hard memory controller consists of the following logic blocks:

- Core and PHY interfaces
- Main control path
- Data buffer controller
- Read and write data buffers

The core interface supports the Avalon® Memory-Mapped (Avalon-MM) interface. The interface communicates to the PHY using the Altera PHY Interface (AFI). The whole control path is split into the main control path and the data buffer controller.

3.4.1.1. Hard Memory Controller Features

Table 7. Features of the Intel Stratix 10 Hard Memory Controller

Feature	Description
Memory standards support	Supports the following memory standards: • DDR4 SDRAM • DDR3 SDRAM
Memory devices support	Supports the following memory devices: • Discrete • UDIMM • RDIMM • LRDIMM • SODIMM
3D Stacked Die support	Supports 2 and 4 height of 3D stacked die for DDR4 to increase memory capacity.
Memory controller bypass mode	You can use this configurable mode to bypass the hard memory controller and use your own customized controller.
Ping-Pong controller mode	You can use this configurable mode to enable two memory controllers to time-share the same set of address and command pins.

continued...

Feature	Description
Interface protocols support	• Supports Avalon-MM interface. • The PHY interface adheres to the AFI protocol.
Rate support	The hard memory controller runs at half rate. It can accept memory access commands from the core logic at half rate or quarter rate.
Configurable memory interface width	Supports data widths from 8 to 72 bits, in 8 bit increments
Multiple ranks support	Supports: • 4 ranks with single slot • 2 ranks with dual slots
Burst adapter	Able to accept burst lengths of 1–127 on the local interface of the controller and map the bursts to efficient memory commands. For applications that must strictly adhere to the Avalon-MM specification, the maximum burst length is 64. No burst chop support for DDR3 and DDR4.
Efficiency optimization features	• Open-page policy—by default, opens page on every access. However, the controller intelligently closes a row based on incoming traffic, which improves the efficiency of the controller especially for random traffic. • Pre-emptive bank management—the controller issues bank management commands early, which ensures that the required row is open when the read or write occurs. • Data reordering—the controller reorders read/write commands. • Additive latency—the controller can issue a `READ`/`WRITE` command after the `ACTIVATE` command to the memory bank prior to t_{RCD}, which increases the command efficiency.
User requested priority	You can assign priority to commands. This feature allows you to specify that higher priority commands are issued earlier to reduce latency.
Starvation counter	Ensures all requests are served after a predefined time out period, which ensures that low priority access are not left behind while reordering data for efficiency.
Timing for address/command bus	To maximize command bandwidth, you can double the number of memory commands in one controller clock cycle: • Quasi-1T addressing for half-rate address and command bus. • Quasi-2T addressing for quarter-rate address and command. *Note:* Quasi-1T and Quasi-2T addressing is not supported for Ping Pong PHY.
Bank interleaving	Able to issue read or write commands continuously to "random" addresses. You must correctly cycle the bank addresses.
On-die termination	The controller controls the on-die termination signal for the memory. This feature improves signal integrity and simplifies your board design.
Refresh features	• User-controlled refresh timing—optionally, you can control when refreshes occur and this allows you to prevent important read or write operations from clashing with the refresh lock-out time. • Per-rank refresh—allows refresh for each individual rank. • Controller-controlled refresh.

continued...

Feature	Description
ECC support	• 8 bit ECC code; single error correction, double error detection (SECDED). • User ECC supporting pass through user ECC bits as part of data bits.
DQS tracking	Tracks the DQS timing and makes auto adjustments to align to the DQS edges.
Power saving features	• Low power modes (power down and self-refresh)—optionally, you can request the controller to put the memory into one of the two low power states. • Automatic power down—puts the memory device in power down mode when the controller is idle. You can configure the idle waiting time. • Memory clock gating.
Mode register set	Access the memory mode register.
DDR4 features	• Bank group support—supports different timing parameters for between bank groups. • Command/Address parity—command and address bus parity check. • Alert reporting—responds to the error alert flag. • Low power auto self refresh— operating temperature triggered auto adjustment to self refresh rate. • Maximum power saving. • Support Direct Dual CS Mode and Direct QuadCS Mode for DDR4 LRDIMM devices. • Support Encoded Quad CSMode for single CS assertion memory mapping for DDR4 LRDIMM devices.
User ZQ calibration	Long or short ZQ calibration request for DDR3 or DDR4.

3.4.1.2. Hard Memory Controller Main Control Path

The main control path performs the following functions:

- Contains the command processing pipeline.
- Monitors all the timing parameters.
- Keeps track of dependencies between memory access commands.
- Guards against memory access hazards.

Table 8. Main Control Path Components

Component	Description
Input interface	• Accepts memory access commands from the core logic at half or quarter rate. • Uses the Avalon-MM protocol. • You can connect the Avalon-MM interface to the AXI bus master in the Platform Designer (formerly Qsys). To connect the Avalon-MM interface, implement the AXI bus master as a Platform Designer component and connect the AXI bus master to the Avalon-MM slave. The Platform Designer interconnect performs the bus translation between the AXI and Avalon-MM bus interfaces. • To support all bypass modes and keep the port count minimum, the super set of all port lists is used as the physical width. Ports are shared among the bypass modes.
Command generator and burst adapter	• Drains your commands from the input interface and feeds them to the timing bank pool. • If read-modify-write is required, inserts the necessary read-modify-write read and write commands into the stream. • The burst adapter chops your arbitrary burst length to the number specified by the memory types.
Timing Bank Pool	• Key component in the memory controller. • Sets parallel queues to track command dependencies. • Signals the ready status of each command being tracked to the arbiter for the final dispatch. • Big scoreboard structure. The number of entries is currently sized to 8 where it monitors up to 8 commands at the same time. • Handles the memory access hazards such as Read After Write (RAW), Write After Read (WAR), and Write After Write (WAW), while part of the timing constraints are being tracked. • Assist the arbiter in reordering row commands and column commands. • When the pool is full, a flow control signal is sent back upstream to stall the traffic.
Arbiter	• Enforces the arbitration rules. • Performs the final arbitration to select a command from all ready commands, and issues the selected command to the memory. • Supports Quasi-1T mode for half rate mode. • For the quasi modes, a row command must be paired with a column command.
Global Timer	Tracks the global timing constraints including: • t_{FAW}—the Four Activates Window parameter that specifies the time period in which only four activate commands are allowed. • t_{RRD}—the delay between back-to-back activate commands to different banks. • Some of the bus turnaround time parameters.
MMR/IOCSR	• The host of all the configuration registers. • Uses Avalon-MM bus to talk to the core. • Core logic can read and write all the configuration bits. • The debug bus is routed to the core through this block.
Sideband	Executes the refresh and power down features.
ECC controller	Although ECC encoding and decoding is performed in soft logic(1), the ECC controller maintains the read-modify-write state machine in the hard solution.
AFI interface	The memory controller communicates with the PHY using this interface.

3.4.1.3. Data Buffer Controller

The data buffer controller performs the following operations:

(1) ECC encoding and decoding is performed in soft logic to exempt the hard connection from routing data bits to a central ECC calculation location. Routing data to a central location removes the modular design benefits and reduces flexibility.

- Manages the read and write access to the data buffers:
  - Provides the data storing pointers to the buffers when the write data is accepted or the read return data arrives.
  - Provides the draining pointer when the write data is dispatched to memory or the read data is read out of the buffer and sent back to users.
- Satisfies the required write latency.
- If ECC support is enabled, assists the main control path to perform read-modify-write.

Data reordering is performed with the data buffer controller and the data buffers.

Each I/O bank contains two data buffer controller blocks for the data buffer lanes that are split within each bank. To improve your timing, place the data buffer controller physically close to the I/O lanes.

3.4.2. Intel Stratix 10 Hard Memory Controller Rate Conversion Feature

The hard memory controller's rate conversion feature allows the hard memory controller and PHY to run at half-rate, even though user logic is configured to run at quarter-rate.

To facilitate timing closure, you may choose to clock your core user logic at quarter-rate, resulting in easier timing closure at the expense of increased area and latency. To improve efficiency and help reduce overall latency, you can run the hard memory controller and PHY at half rate.

The rate conversion feature converts traffic from the FPGA core to the hard memory controller from quarter-rate to half-rate, and traffic from the hard memory controller to the FPGA core from half-rate to quarter-rate. From the perspective of user logic inside the FPGA core, the effect is the same as if the hard memory controller were running at quarter-rate.

The rate conversion feature is enabled automatically during IP generation whenever all of the following conditions are met:

- The hard memory controller is in use.
- User logic runs at quarter-rate.
- The interface targets either an ES2 or production device.
- Running the hard memory controller at half-rate does not exceed the fMax specification of the hard memory controller and hard PHY.

When the rate conversion feature is enabled, you should see the following info message displayed in the IP generation GUI:
`PHY and controller running at 2x the frequency of user logic for improved efficiency.`

3.5. Hardware Resource Sharing Among Multiple Intel Stratix 10 EMIFs

Often, it is necessary or desirable to share certain hardware resources between interfaces.

3.5.1. I/O SSM Sharing

The I/O SSM contains a hard Nios® II processor and dedicated memory storing the calibration software code and data.

When a column contains multiple memory interfaces, the Nios II processor calibrates each interface serially. Interfaces placed within the same I/O column always share the same I/O SSM. The Intel Quartus Prime Fitter handles I/O SSM sharing automatically.

3.5.2. I/O Bank Sharing

Data lanes from multiple compatible interfaces can share a physical I/O bank to achieve a more compact pin placement. To share an I/O bank, interfaces must use the same memory protocol, rate, frequency, I/O standard, and PLL reference clock signal.

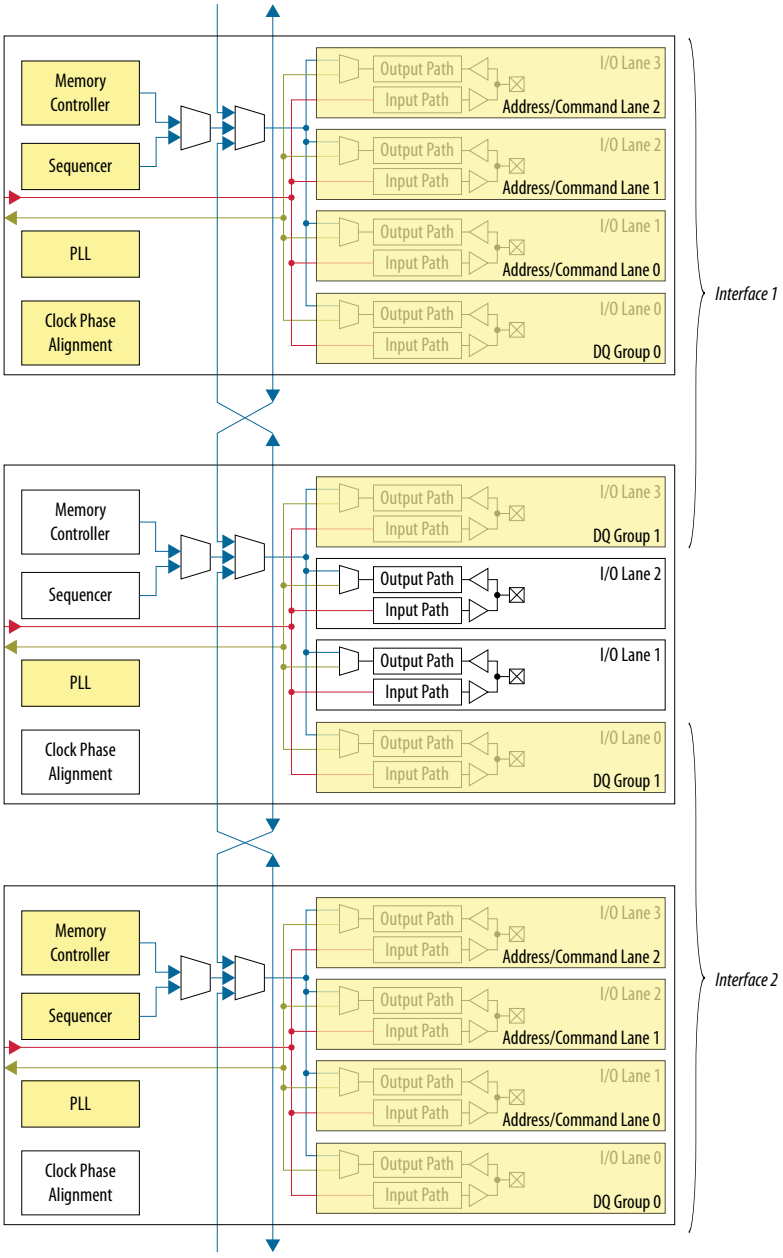
Rules for Sharing I/O Banks

- A bank cannot serve as the address and command bank for more than one interface. This means that lanes which implement address and command pins for different interfaces cannot be allocated to the same physical bank.

  Note: An exception to the above rule exists when two interfaces are configured in a Ping-Pong PHY fashion. In such a configuration, two interfaces share the same set of address and command pins, effectively meaning that they share the same address and command tile.
- Pins within a lane cannot be shared by multiple memory interfaces.
- Any pins in a bank which are not used by the EMIF IP can serve as general-purpose I/O pins of uncalibrated I/O standard with the same voltage settings.
- You can configure a bank as LVDS or as EMIF, but not both at the same time.
- Interfaces that share banks must reside at adjacent bank locations.

The following diagram illustrates two x16 interfaces sharing an I/O bank. The two interfaces share the same clock phase alignment block, so that one core clock signal can interact with both interfaces. Without sharing, the two interfaces would occupy a total of four physical banks instead of three.

Figure 15. I/O Bank Sharing



3.5.3. PLL Reference Clock Sharing

To implement PLL reference clock sharing, in your RTL code connect the PLL reference clock signal at your design's top-level to the PLL reference clock port of multiple interfaces.

To share a PLL reference clock, the following requirements must be met:

- Interfaces must expect a reference clock signal of the same frequency.
- Interfaces must be placed in the same column.
- Interfaces must be placed at adjacent bank locations.

3.5.4. Core Clock Network Sharing

It is often desirable or necessary for multiple memory interfaces to be accessible using a single clock domain in the FPGA core.

You might want to share core clock networks for the following reasons:

- To minimize the area and latency penalty associated with clock domain crossing.
- To minimize consumption of core clock networks.

Multiple memory interfaces can share the same core clock signals under the following conditions:

- The memory interfaces have the same protocol, rate, frequency, and PLL reference clock source.
- The interfaces reside in the same I/O column.
- The interfaces reside in adjacent bank locations.

For multiple memory interfaces to share core clocks, you must specify one of the interfaces as master and the remaining interfaces as slaves. Use the `Core clocks sharing` setting in the parameter editor to specify the master and slaves.

In your RTL, connect the `clks_sharing_master_out` signal from the master interface to the `clks_sharing_slave_in` signal of all the slave interfaces. Both the master and slave interfaces expose their own output clock ports in the RTL (e.g. `emif_usr_clk, afi_clk`), but the signals are equivalent, so it does not matter whether a clock port from a master or a slave is used.

Core clock sharing necessitates PLL reference clock sharing; therefore, only the master interface exposes an input port for the PLL reference clock. All slave interfaces use the same PLL reference clock signal.

3.6. User-requested Reset in Intel Stratix 10 EMIF IP

The following table summarizes information about the user-requested reset mechanism in the Intel Stratix 10 EMIF IP.

Table 9.

	Description
Reset-related signals	`local_reset_req` (input) `local_reset_done` (output)
When can user logic request a reset?	`local_reset_req` has effect only when `local_reset_done` is high. After device power-on, the `local_reset_done` signal transitions high after the completion of the first calibration, whether the calibration is successful or not. In subsequent

continued...

	Description
	calibration in user mode, the `local_reset_done` signal transitions high once the calibration is completed. The `local_reset_done` signal takes more time to transition high in first calibration after device power-on as more operations are required put the PHY into working state.
Is user-requested reset a requirement?	A user-requested reset is optional. The I/O SSM automatically ensures that the memory interface begins from a known state as part of the device power-on sequence. A user-requested reset is necessarily only if the user logic must explicitly reset a memory interface after the device power-on sequence.
When does a user-requested reset actually happen?	A reset request is handled by the I/O SSM. If the I/O SSM receives a reset request from multiple interfaces within the same I/O column, it must serialize the reset sequence of the individual interfaces. You should not make assumptions about when the reset sequence will begin after a request is issued.
Timing requirement and triggering mechanism.	Reset request is sent by transitioning the `local_reset_req` signal from low to high, then keeping the signal at the high state for a minimum of 2 EMIF core clock cycles, then transitioning the signal from high to low. `local_reset_req` is asynchronous in that there is no setup/hold timing to meet, but it must meet the minimum pulse width requirement of 2 EMIF core clock cycles.
How long can an external memory interface be kept in reset?	It is not possible to keep an external memory interface in reset indefinitely. Asserting `local_reset_req` high continuously has no effect as a reset request is completed by a full 0->1->0 pulse.
Delaying initial calibration.	Initial calibration cannot be skipped. The `local_reset_done` signal is driven high only after initial calibration has completed.
Reset scope (within an external memory interface).	Only circuits that are required to restore EMIF to power-up state are reset. Excluded from the reset sequence are the IOSSM, the IOPLL(s), the DLL(s), and the CPA.
Reset scope (within an I/O column).	`local_reset_req` is a per-interface reset.

Method for Initiating a User-requested Reset

Step 1 - Precondition

Before asserting `local_reset_req`, user logic must ensure that the `local_reset_done` signal is high.

As part of the device power-on sequence, the `local_reset_done` signal automatically transitions to high upon the completion of the interface calibration sequence, regardless of whether calibration is successful or not.

Note: When targeting a group of interfaces that share the same core clocks, user logic must ensure that the `local_reset_done` signal of every interface is high.

Step 2 - Reset Request

After the pre-condition is satisfied, user logic can send a reset request by driving the `local_cal_req` signal from low to high and then low again (that is, by sending a pulse of 1).

- The low-to-high and high-to-low transitions can occur asychronously; that is, they need not happen in relation to any clock edges. However, the pulse must meet a minimum pulse width of at least 2 EMIF core clock cycles. For example, if the `emif_usr_clk` has a period of 4ns, then the `local_reset_req` pulse must last at least 8ns (that is, two `emif_usr_clk` periods).
- The reset request is considered complete only after the high-to-low transition. The EMIF IP does not initiate the reset sequence when the `local_reset_req` is simply held high.
- Additional pulses to `local_reset_req` are ignored until the reset sequence is completed.

Optional - Detecting local_reset_done deassertion and assertion

If you want, you can monitor the status of the `local_reset_done` signal to explicitly detect the status of the reset sequence.

- After the EMIF IP receives a reset request, it deasserts the `local_reset_done` signal. After initial power-up calibration, `local_reset_done` is de-asserted only in response to a user-requested reset. The reset sequence is imminent when `local_reset_done` has transitioned to low, although the exact timing depends on the current state of the I/O SSM. As part of the EMIF reset sequence, the core reset signal (`emif_usr_reset_n`, `afi_reset_n`) is driven low. Do not use a register reset by the core reset signal to sample `local_reset_done`.
- After the reset sequence has completed, `local_reset_done` is driven high again. `local_reset_done` being driven high indicates the completion of the reset sequence and the readiness to accept a new reset request; however, it does not imply that calibration was successful or that the hard memory controller is ready to accept requests. For these purposes, user logic must check signals such as `afi_cal_success`, `afi_cal_fail`, and `amm_ready`.

3.7. Intel Stratix 10 EMIF for Hard Processor Subsystem

The Intel Stratix 10 EMIF IP can enable the Intel Stratix 10 Hard Processor Subsystem (HPS) to access external DRAM memory devices.

To enable connectivity between the Intel Stratix 10 HPS and the Intel Stratix 10 EMIF IP, you must create and configure an instance of the Intel Stratix 10 External Memory Interface for HPS IP core, and use Platform Designer to connect it to the Intel Stratix 10 Hard Processor Subsystem instance in your system.

Supported Modes

The Intel Stratix 10 Hard Processor Subsystem is compatible with the following external memory configurations:

Table 10. Intel Stratix 10 Hard Processor Subsystem Compatibility

Protocol	DDR3, DDR4
Maximum memory clock frequency	DDR3: 933 MHz DDR4: 1200 MHz
Configuration	Hard PHY with hard memory controller
Clock rate of PHY and hard memory controller	Half-rate
Data width (without ECC)	16-bit, 32-bit, 64-bit
Data width (with ECC)	24-bit, 40-bit, 72-bit
DQ width per group	x8
Maximum number of I/O lanes for address/command	3
Memory format	Discrete, UDIMM, SODIMM, RDIMM
Ranks / CS# width	Up to 2

Note: You must provide a free running and stable reference clock source to external memory interface cores before the start of device configuration.

3.7.1. Restrictions on I/O Bank Usage for Intel Stratix 10 EMIF IP with HPS

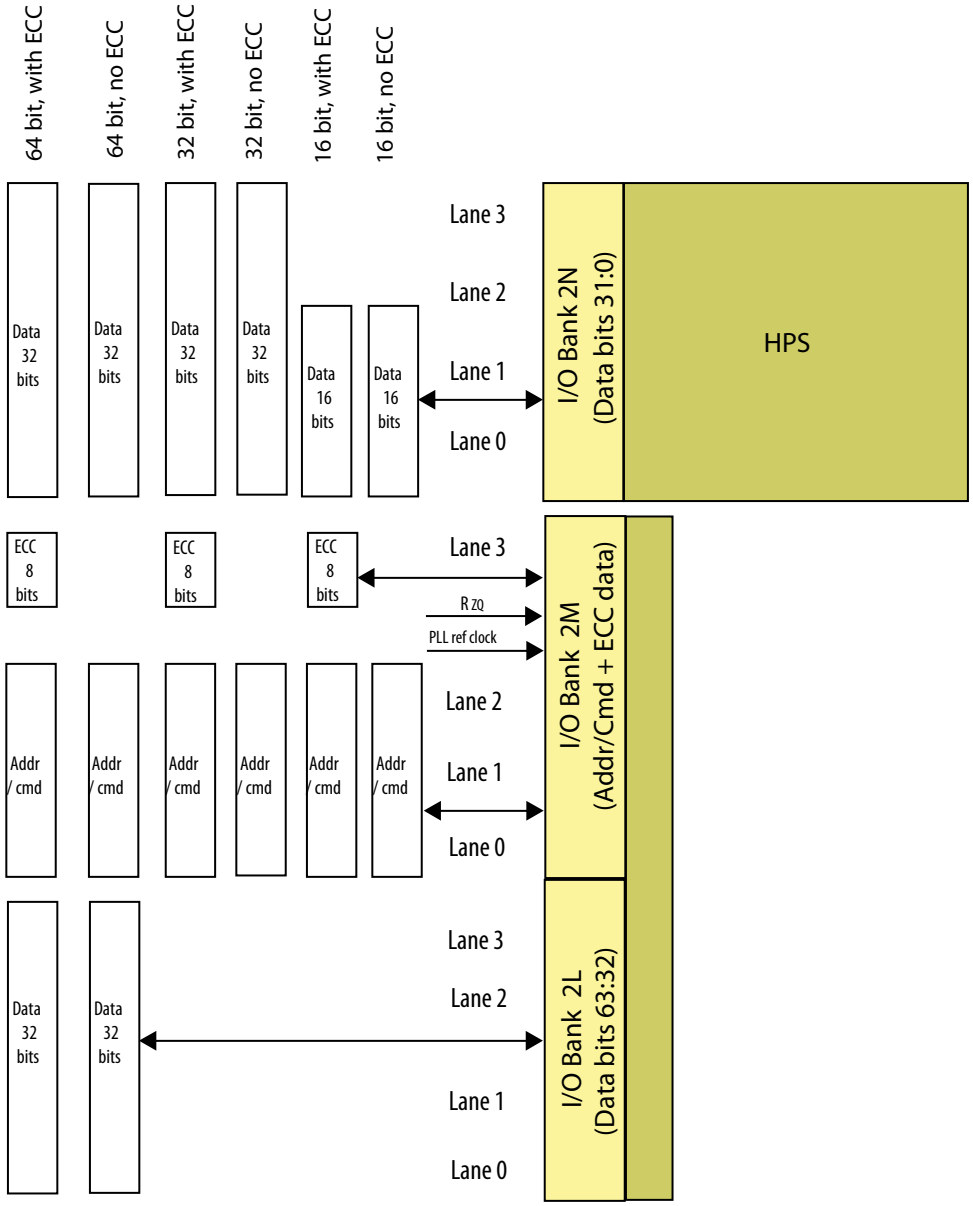
You can use only certain Intel Stratix 10 I/O banks to implement Intel Stratix 10 EMIF IP with the Intel Stratix 10 Hard Processor Subsystem (HPS).

The restrictions on I/O bank usage result from the Intel Stratix 10 HPS having hard-wired connections to the EMIF circuits in the I/O banks closest to the HPS. For any given EMIF configuration, the pin-out of the EMIF-to-HPS interface is fixed.

Note: The restrictions described in this topic apply to all new designs. If you have an existing board that already works, you do not need to change it to comply with these restrictions.

The following diagram illustrates the use of I/O banks and lanes for various EMIF-HPS data widths:

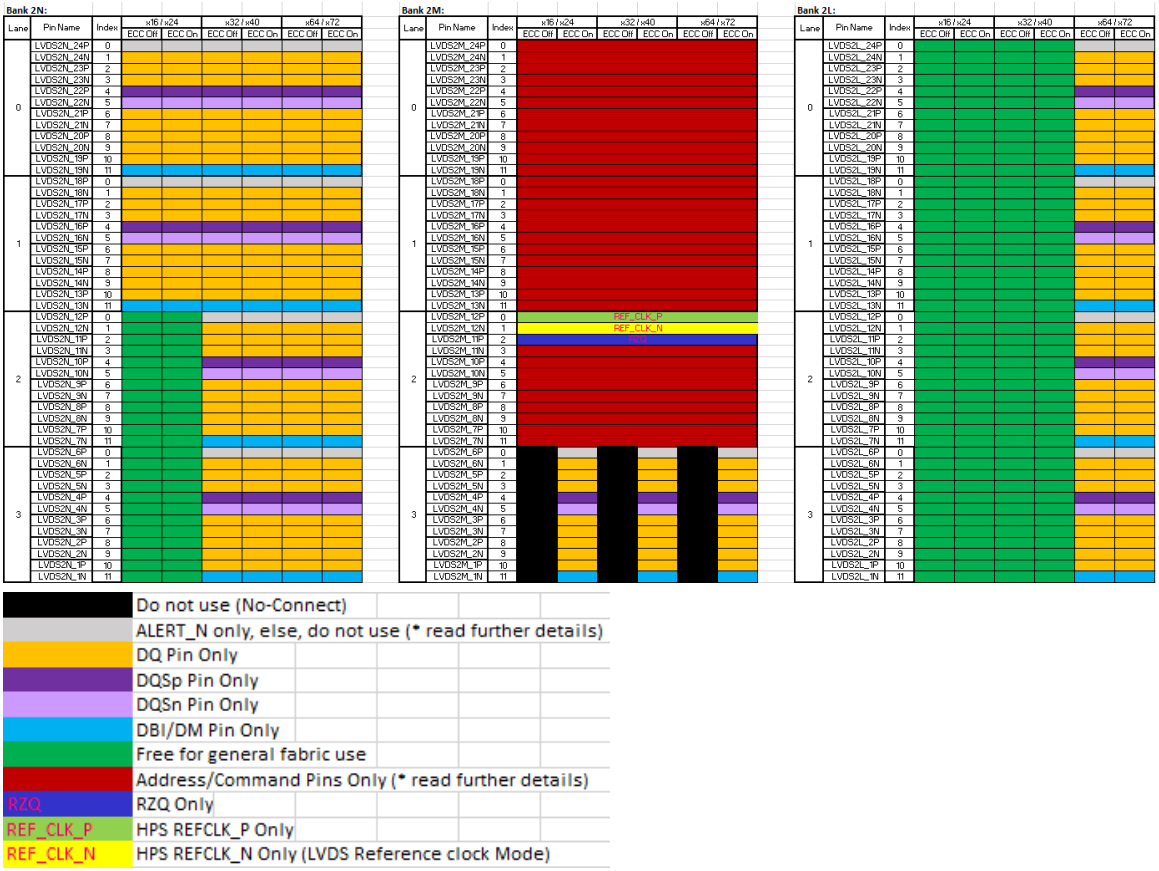
Figure 16. Intel Stratix 10 External Memory Interfaces I/O Bank and Lanes Usage



The HPS EMIF uses the closest located external memory interfaces I/O banks to connect to SDRAM.

The following diagram illustrates restrictions on I/O pin usage. Refer to the text following the diagram for a detailed explanation of these restrictions.

Figure 17. I/O Pin Usage Restrictions for Intel Stratix 10 External Memory Interface with HPS



If no HPS EMIF is used in a system, the entire HPS EMIF bank can be used as FPGA general purpose I/O. If there is an HPS EMIF in a system, the unused HPS EMIF pins can be used as FPGA general purpose I/O, with the following restrictions:

- Bank 2M:
  - Lane 3 of Bank 2M is used for data bits only when ECC mode is active. Whether ECC is active or not, you must not put general purpose I/Os in this lane.
  - Lanes 2, 1, and 0 are used for SDRAM address and command. Unused pins in these lanes should not be used by the FPGA fabric, because their operation cannot be guaranteed.
- Bank 2N and Bank 2L :
  - Lanes 3, 2, 1, and 0 are used for data bits.
  - With 64-bit data widths, unused pins in these banks should not be used by the FPGA fabric, because their operation cannot be guaranteed.
  - With 32-bit data widths, unused pins in Bank 2N should not be used by the FPGA fabric, because their operation cannot be guaranteed. Lanes 0-3 of bank 2L are not used by the HPS EMIF, therefore any pins within these lanes can be used by the FPGA fabric.
  - With 16-bit data widths, Intel Quartus Prime assigns lane 0 and lane 1 as data lanes in bank 2N. Unused pins in these two lanes should not be used by the FPGA fabric, because their operation cannot be guaranteed. Lanes 2 and 3 are not used by the HPS EMIF, therefore pins within these lanes can be used by the FPGA fabric.

By default, the Intel Stratix 10 External Memory Interface for HPS IP core together with the Intel Quartus Prime Fitter automatically implements a starting point placement which you may need to modify. You must adhere to the following requirements, which are specific to HPS EMIF:

1. Within a single data lane (which implements a single x8 DQS group):
   - DQ pins must use pins at indices 1, 2, 3, 6, 7, 8, 9, 10. You may swap the locations between the DQ bits (that is, you may swap location of DQ[0] and DQ[3]) so long as the resulting pin-out uses pins at these indices only.
   - DM/DBI pin must use pin at index 11. There is no flexibility.
   - DQS and DQS# must use pins at index 4 and 5, respectively. There is no flexibility.
   - Pin index 0 must have no connection, unless used for `alert#` or HPS REFCLK_P, or address/command, or general-purpose I/O, where allowed.
2. The above figures show an overview of how the data lanes are used, depending on the width of the interface. The following table shows the I/O bank and I/O lanes that you must use, depending on the width and configuration of the interface.

Configuration	DQS Group Placement
16 bit	Must be placed in I/O lanes 0 and 1 of 2N.
16 bit + ECC	Must be placed in I/O lanes 0 and 1 of 2N and I/O lane 3 of 2M.
32 bit	Must be placed in 2N.

continued...

Configuration	DQS Group Placement
32 bit + ECC	Must be placed in 2N and I/O lane 3 of 2M.
64 bit	Must be placed in 2N and 2L.
64 bit + ECC	Must be placed in 2N, 2L, and I/O lane 3 of 2M.

Note: a. In all cases, the DQS groups can be swapped around in the I/O banks shown. There is no requirement for the ECC DQS group to be placed in bank 2M.

b. I/O lane 3 of bank 2M cannot be used if ECC is turned off. You must not put general purpose I/Os in lane 3 of bank 2M.

3. You must not change placement of the address and command pins from the default placement in I/O bank 2M.
4. The `alert#` pin must be at index 0 (of any lane of any bank) and must be grouped with its Intel Quartus Prime software-assigned DQS group, or must be in any unused pin within address and command section. Bank 2N, lane 0, index 0 or bank 2N, lane 1, index 0 are the recommended locations for `alert#` for new designs. This allows for maximum flexibility of different interface widths. Existing (working) designs with `alert#`, on unused address and command pins, or in other data I/O lanes at index 0, is allowed.
5. The PLL reference clock must be placed in I/O bank 2M with the address and command pins. Failure to do this results in device configuration problems. The PLL reference clock must be running at the correct frequency before device configuration occurs.
6. The R_{ZQ} pin must be placed in I/O bank 2M with the address and command pins. Failure to do this will cause Fitter or device configuration problems.

To override the default generated pin assignments, comment out the relevant `HPS_LOCATION` assignments in the `.qip` file, and add your own location assignments (using `set_location_assignment`) in the `.qsf` file.

3.7.2. Using the Legacy EMIF Debug Toolkit with Intel Stratix 10 HPS Interfaces

The Legacy External Memory Interface Debug Toolkit is not directly compatible with Intel Stratix 10 HPS interfaces.

To debug your Intel Stratix 10 HPS interface using the Legacy EMIF Debug Toolkit, you should create an identically parameterized, non-HPS version of your interface, and apply the toolkit to that interface. When you finish debugging this non-HPS interface, you can then apply any needed changes to your HPS interface, and continue your design development.

3.7.3. HPS EMIF Simulation

Simulation of a design containing Intel Stratix 10 HPS EMIF is not supported.

3.8. Intel Stratix 10 EMIF Ping Pong PHY

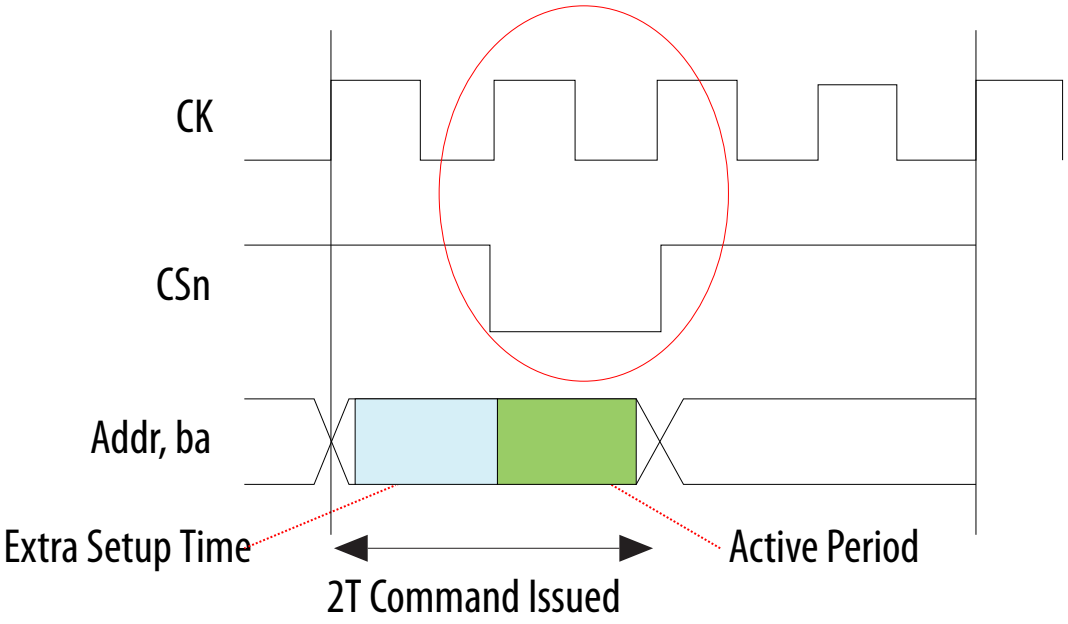
Ping Pong PHY allows two memory interfaces to share the address and command bus through time multiplexing. Compared to having two independent interfaces that allocate address and command lanes separately, Ping Pong PHY achieves the same throughput with fewer resources, by sharing the address and command lanes.

In Intel Stratix 10 EMIF, Ping Pong PHY supports both half-rate and quarter-rate interfaces for DDR3, and quarter-rate for DDR4.

3.8.1. Intel Stratix 10 Ping Pong PHY Feature Description

Conventionally, the address and command buses of a DDR3 or DDR4 half-rate or quarter-rate interface use 2T time—meaning that commands are issued for two full-rate clock cycles, as illustrated below.

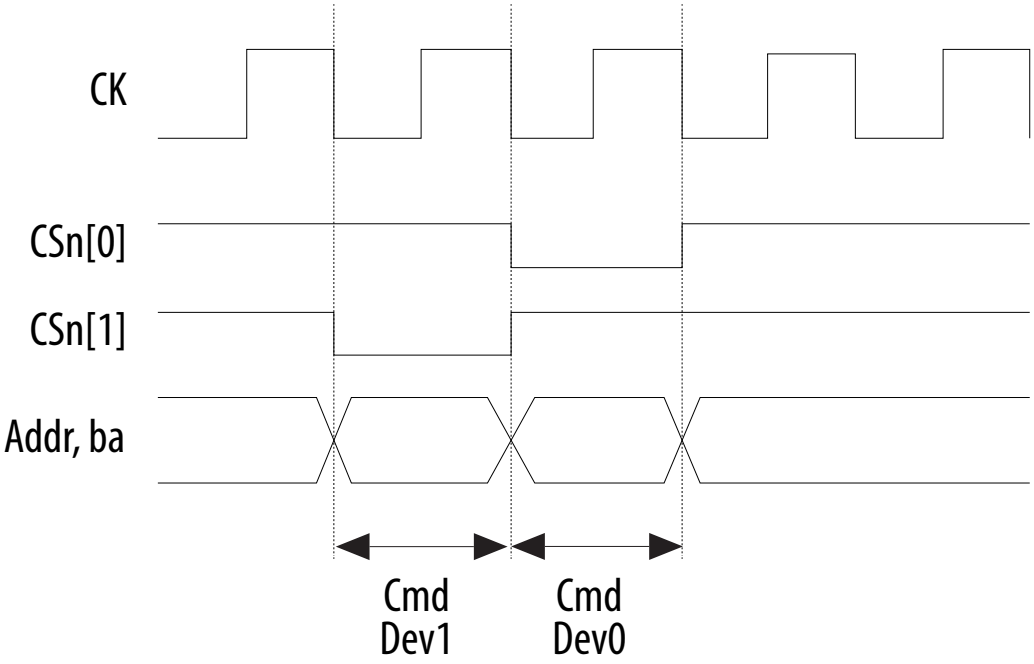
Figure 18. 2T Command Timing



With the Ping Pong PHY, address and command signals from two independent controllers are multiplexed onto shared buses by delaying one of the controller outputs by one full-rate clock cycle. The result is 1T timing, with a new command being issued on each full-rate clock cycle. The following figure shows address and command timing for the Ping Pong PHY.

The command signals CS, ODT, and CKE have two signals (one for ping and one for pong); the other address and command signals are shared.

Figure 19. 1T Command Timing Use by Ping Pong PHY



3.8.2. Intel Stratix 10 Ping Pong PHY Architecture

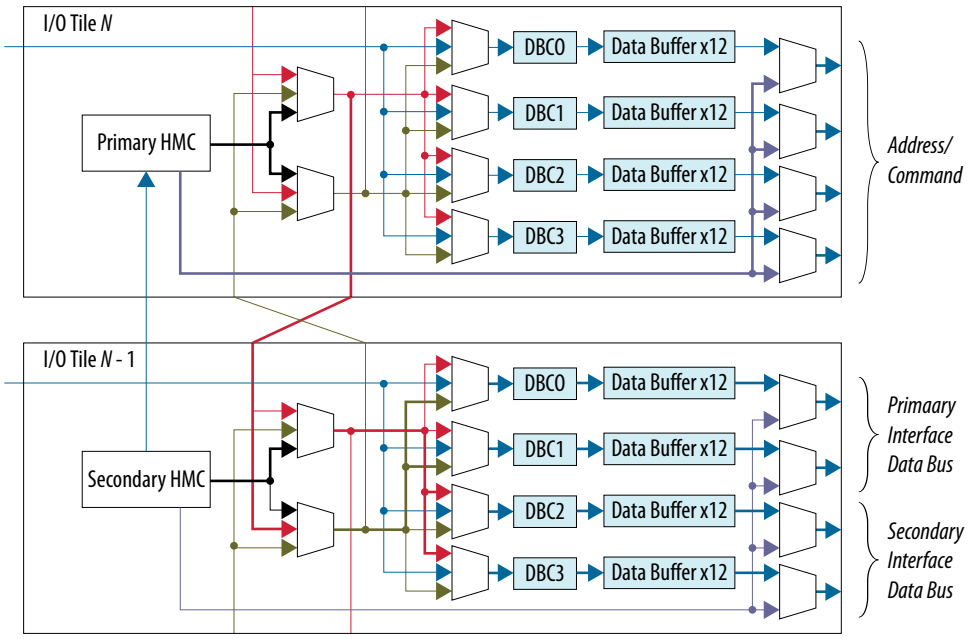
In Intel Stratix 10 EMIF, the Ping Pong PHY feature can be enabled only with the hard memory controller, where two hard memory controllers are instantiated—one for the primary interface and one for the secondary interface.

The hard memory controller I/O bank of the primary interface is used for address and command and is always adjacent and above the hard memory controller bank of the secondary interface. All four lanes of the primary hard memory controller bank are used for address and command.

The following example shows a 2x16 Ping Pong PHY bank-lane configuration. The upper bank (I/O bank N) is the address and command bank, which serves both the primary and secondary interfaces. The primary hard memory controller is linked to the secondary interface by the Ping Pong bus. The lower bank (I/O bank N-1) is the secondary interface bank, which carries the data buses for both primary and secondary interfaces. In the 2x16 case a total of four I/O banks are required for data, hence two banks in total are sufficient for the implementation.

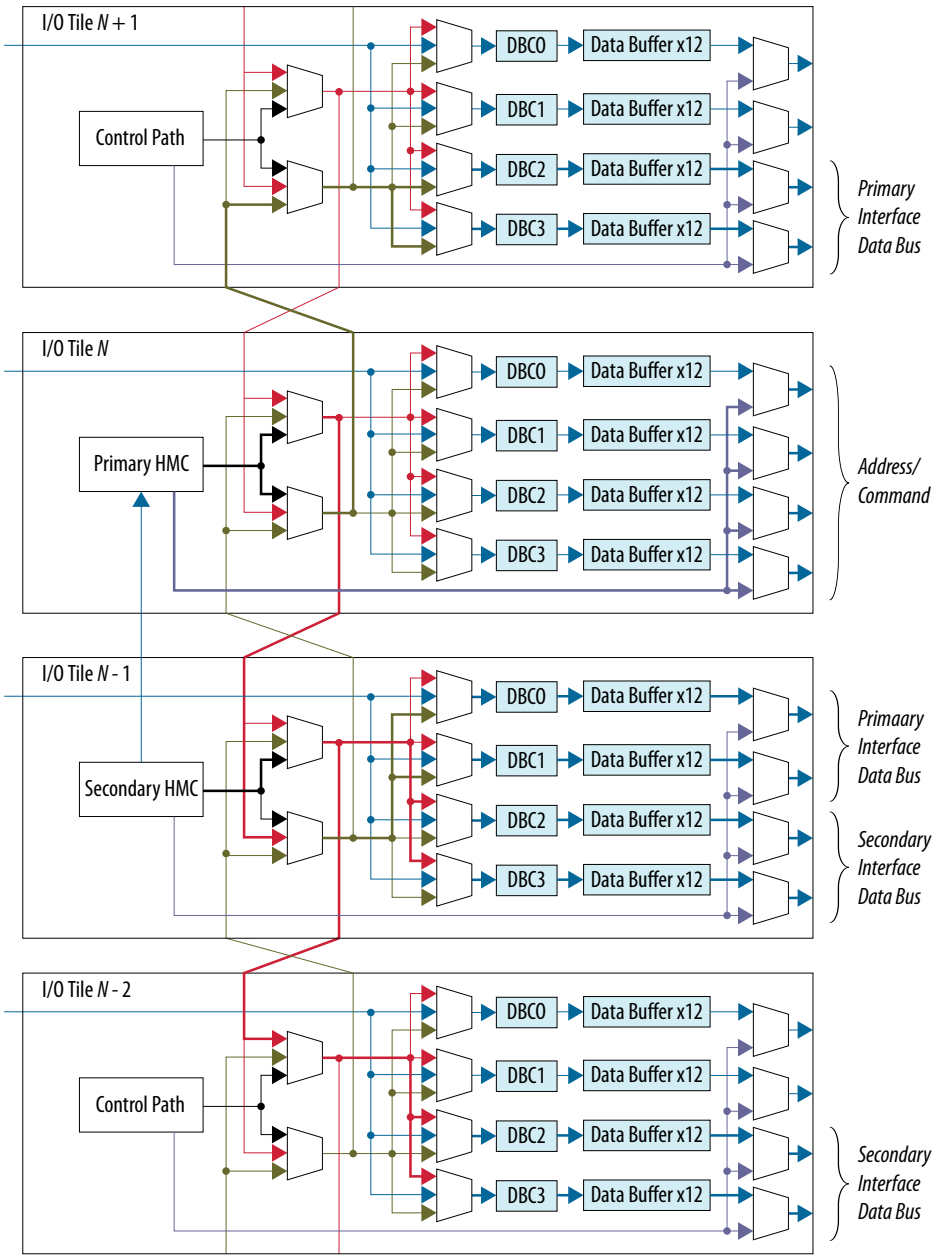
The data for the primary interface is routed down to the top two lanes of the secondary I/O bank, and the data for the secondary interface is routed to the bottom two lanes of the secondary I/O bank.

Figure 20. 2x16 Ping Pong PHY I/O Bank-Lane Configuration



A 2x32 interface can be implemented similarly, with the additional data lanes placed above and below the primary and secondary I/O banks, such that primary data lanes are placed above the primary bank and secondary data lanes are placed below the secondary bank.

Figure 21. 2x32 Ping Pong PHY I/O Bank-Lane Configuration.



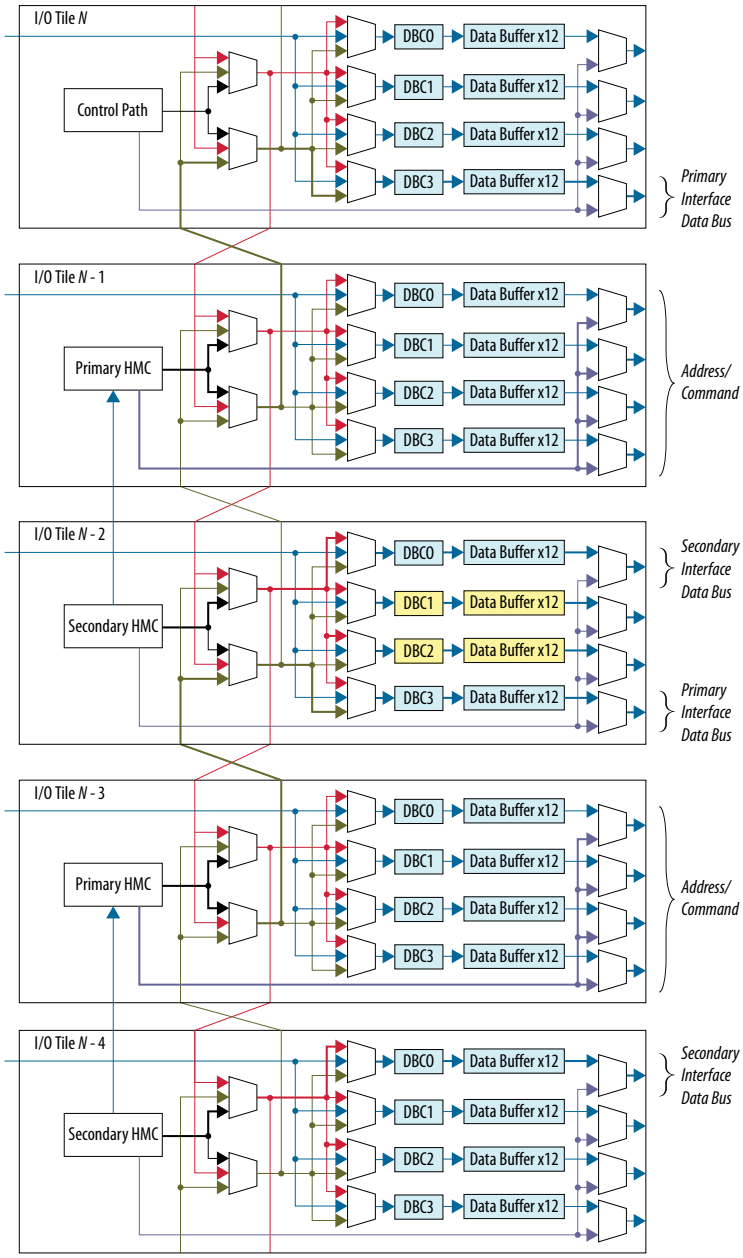
3.8.3. Intel Stratix 10 Ping Pong PHY Limitations

Ping Pong PHY supports up to two ranks per memory interface. In addition, the maximum data width is x72, which is half the maximum width of x144 for a single interface.

Ping Pong PHY uses all lanes of the address and command I/O bank as address and command. For information on pin allocations, refer to the pin-out file for your device, at *Pin-Out Files for Intel FPGA Devices* on `www.altera.com`.

An additional limitation is that I/O lanes may be left unused when you instantiate multiple pairs of Ping Pong PHY interfaces. The following diagram shows two pairs of x8 Pin Pong controllers (a total of 4 interfaces). Lanes highlighted in yellow are not driven by any memory interfaces (unused lanes and pins can still serve as general purpose I/Os). Even with some I/O lanes left unused, the Ping Pong PHY approach is still beneficial in terms of resource usage, compared to independent interfaces. Memory widths of 24 bits and 40 bits have a similar situation, while 16 bit, 32 bit, and 64 bit memory widths do not suffer this limitation.

Figure 22. Two Pairs of x8 Pin-Pong PHY Controllers



Related Information

Pin-Out Files for Intel FPGA Devices

3.8.4. Intel Stratix 10 Ping Pong PHY Calibration

A Ping Pong PHY interface is calibrated as a regular interface of double width.

Calibration of a Ping Pong PHY interface incorporates two sequencers, one on the primary hard memory controller I/O bank, and one on the secondary hard memory controller I/O bank. To ensure that the two sequencers issue instructions on the same memory clock cycle, the Nios II processor configures the sequencer on the primary hard memory controller to receive a token from the secondary interface, ignoring any commands from the Avalon bus. Additional delays are programmed on the secondary interface to allow for the passing of the token from the sequencer on the secondary hard memory controller tile to the sequencer on the primary hard memory controller tile. During calibration, the Nios II processor assumes that commands are always issued from the sequencer on the primary hard memory controller I/O bank. After calibration, the Nios II processor adjusts the delays for use with the primary and secondary hard memory controllers.

3.8.5. Using the Ping Pong PHY

The following steps describe how to use the Ping Pong PHY for Intel Stratix 10 EMIF.

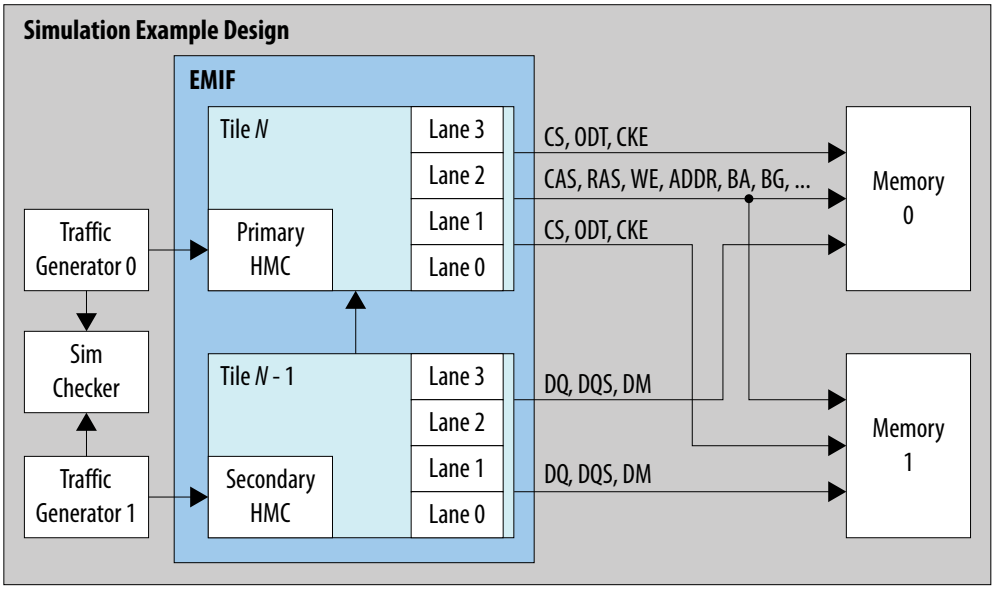
1. Configure a single memory interface according to your requirements.
2. Select **Instantiate two controllers sharing a Ping Pong PHY** on the **General** tab in the parameter editor.
   The Intel Quartus Prime software replicates the interface, resulting in two memory controllers and a shared PHY. The system configures the I/O bank-lane structure, without further input from you.

3.8.6. Ping Pong PHY Simulation Example Design

The following figure illustrates a top-level block diagram of a generated Ping Pong PHY simulation example design, using two I/O banks.

Functionally, the IP interfaces with user traffic separately, as it would with two independent memory interfaces. You can also generate synthesizable example designs, where the external memory interface IP interfaces with a traffic generator.

Figure 23. Ping Pong PHY Simulation Example Design



4. Intel Stratix 10 EMIF IP End-User Signals

4.1. Interface and Signal Descriptions

The following sections describe each of the interfaces and their signals, by protocol, for the Intel Stratix 10 EMIF IP.

4.1.1. Intel Stratix 10 EMIF IP Interfaces for DDR3

The interfaces in the Intel Stratix 10 External Memory Interface IP each have signals that can be connected in Platform Designer. The following table lists the interfaces and corresponding interface types for DDR3.

Table 11. Interfaces for DDR3

Interface Name	Interface Type	Description
local_reset_req	Conduit	Local reset request. Output signal from local_reset_combiner
local_reset_status	Conduit	Local reset status. Input signal to the local_reset_combiner
pll_ref_clk	Clock Input	PLL reference clock input
pll_locked	Conduit	PLL locked signal
pll_extra_clk_0	Clock Output	Additional core clock 0
pll_extra_clk_1	Clock Output	Additional core clock 1
pll_extra_clk_2	Clock Output	Additional core clock 2
pll_extra_clk_3	Clock Output	Additional core clock 3
oct	Conduit	On-Chip Termination (OCT) interface
mem	Conduit	Interface between FPGA and external memory
status	Conduit	PHY calibration status interface
afi_reset_n	Reset Output	AFI reset interface
afi_clk	Clock Output	AFI clock interface
afi_half_clk	Clock Output	AFI half-rate clock interface
afi	Conduit	Altera PHY Interface (AFI)
emif_usr_reset_n	Reset Output	User clock domain reset interface
emif_usr_clk	Clock Output	User clock interface

continued...

Interface Name	Interface Type	Description
emif_usr_reset_n_sec	Reset Output	User clock domain reset interface (for the secondary interface in ping-pong configuration)
emif_usr_clk_sec	Clock Output	User clock interface (for the secondary interface in ping-pong configuration)
cal_debug_reset_n	Reset Input	User calibration debug clock domain reset interface
cal_debug_clk	Clock Input	User calibration debug clock interface
cal_debug_out_reset_n	Reset Output	User calibration debug clock domain reset interface
cal_debug_out_clk	Clock Output	User calibration debug clock interface
clks_sharing_master_out	Conduit	Core clocks sharing master interface
clks_sharing_slave_in	Conduit	Core clocks sharing slave input interface
clks_sharing_slave_out	Conduit	Core clocks sharing slave output interface
ctrl_amm	Avalon Memory-Mapped Slave	Controller Avalon Memory-Mapped interface
ctrl_auto_precharge	Conduit	Controller auto-precharge interface
ctrl_user_priority	Conduit	Controller user-requested priority interface
ctrl_ecc_user_interrupt	Conduit	Controller ECC user interrupt interface
ctrl_ecc_readdataerror	Conduit	Controller ECC read data error indication interface
ctrl_ecc_status	Conduit	Controller ECC status interface
ctrl_mmr_slave	Avalon Memory-Mapped Slave	Controller MMR slave interface
hps_emif	Conduit	Conduit between Hard Processor Subsystem and memory interface
cal_debug	Avalon Memory-Mapped Slave	Calibration debug interface
cal_debug_out	Avalon Memory-Mapped Master	Calibration debug interface

4.1.1.1. local_reset_req for DDR3

Local reset request. Output signal from local_reset_combiner

Table 12. Interface: local_reset_req

Interface type: Conduit

Port Name	Direction	Description
local_reset_req	Input	Signal from user logic to request the memory interface to be reset and recalibrated. Reset request is sent by transitioning the local_reset_req signal from low to high, then keeping the signal at the high state for a minimum of 2 EMIF core clock cycles, then transitioning the signal from high to low. local_reset_req is asynchronous in that there is no setup/hold timing to meet, but it must meet the minimum pulse width requirement of 2 EMIF core clock cycles.

4.1.1.2. local_reset_status for DDR3

Local reset status. Input signal to the local_reset_combiner

Table 13. Interface: local_reset_status

Interface type: Conduit

Port Name	Direction	Description
local_reset_done	Output	Signal from memory interface to indicate whether it has completed a reset sequence, is currently out of reset, and is ready for a new reset request. When local_reset_done is low, the memory interface is in reset.

4.1.1.3. pll_ref_clk for DDR3

PLL reference clock input

Table 14. Interface: pll_ref_clk

Interface type: Clock Input

Port Name	Direction	Description
pll_ref_clk	Input	PLL reference clock input

4.1.1.4. pll_locked for DDR3

PLL locked signal

Table 15. Interface: pll_locked

Interface type: Conduit

Port Name	Direction	Description
pll_locked	Output	PLL lock signal to indicate whether the PLL has locked

4.1.1.5. pll_extra_clk_0 for DDR3

Additional core clock 0

Table 16. Interface: pll_extra_clk_0

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_0	Output	PLL extra core clock signal output 0. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.1.6. pll_extra_clk_1 for DDR3

Additional core clock 1

Table 17. Interface: pll_extra_clk_1

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_1	Output	PLL extra core clock signal output 1. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.1.7. pll_extra_clk_2 for DDR3

Additional core clock 2

Table 18. Interface: pll_extra_clk_2

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_2	Output	PLL extra core clock signal output 2. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.1.8. pll_extra_clk_3 for DDR3

Additional core clock 3

Table 19. Interface: pll_extra_clk_3

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_3	Output	PLL extra core clock signal output 3. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.1.9. oct for DDR3

On-Chip Termination (OCT) interface

Table 20. Interface: oct

Interface type: Conduit

Port Name	Direction	Description
oct_rzqin	Input	Calibrated On-Chip Termination (OCT) RZQ input pin

4.1.1.10. mem for DDR3

Interface between FPGA and external memory

Table 21. Interface: mem

Interface type: Conduit

Port Name	Direction	Description
mem_ck	Output	CK clock
mem_ck_n	Output	CK clock (negative leg)
mem_a	Output	Address
mem_ba	Output	Bank address
mem_cke	Output	Clock enable
mem_cs_n	Output	Chip select
mem_rm	Output	Rank multiplication for LRDIMM. Typically, mem_rm[0] and mem_rm[1] connect to CS2# and CS3# of the memory buffer of all LRDIMM slots.
mem_odt	Output	On-die termination
mem_ras_n	Output	RAS command
mem_cas_n	Output	CAS command
mem_we_n	Output	WE command
mem_reset_n	Output	Asynchronous reset
mem_par	Output	Command and address parity
mem_dm	Output	Write data mask
mem_dq	Bidirectional	Read/write data
mem_dqs	Bidirectional	Data strobe
mem_dqs_n	Bidirectional	Data strobe (negative leg)
mem_alert_n	Input	Alert flag

4.1.1.11. status for DDR3

PHY calibration status interface

Table 22. Interface: status

Interface type: Conduit

Port Name	Direction	Description
local_cal_success	Output	When high, indicates that PHY calibration was successful
local_cal_fail	Output	When high, indicates that PHY calibration failed

4.1.1.12. afi_reset_n for DDR3

AFI reset interface

Table 23. Interface: afi_reset_n

Interface type: Reset Output

Port Name	Direction	Description
afi_reset_n	Output	Reset for the AFI clock domain. Asynchronous assertion and synchronous deassertion

4.1.1.13. afi_clk for DDR3

AFI clock interface

Table 24. Interface: afi_clk

Interface type: Clock Output

Port Name	Direction	Description
afi_clk	Output	Clock for the Altera PHY Interface (AFI)

4.1.1.14. afi_half_clk for DDR3

AFI half-rate clock interface

Table 25. Interface: afi_half_clk

Interface type: Clock Output

Port Name	Direction	Description
afi_half_clk	Output	Clock running at half the frequency of the AFI clock afi_clk

4.1.1.15. afi for DDR3

Altera PHY Interface (AFI)

Table 26. Interface: afi

Interface type: Conduit

Port Name	Direction	Description
afi_cal_success	Output	Signals calibration successful completion
afi_cal_fail	Output	Signals calibration failure
afi_cal_req	Input	When asserted, the interface is recalibrated
afi_rlat	Output	Latency in afi_clk cycles between read command and read data valid
afi_wlat	Output	Latency in afi_clk cycles between write command and write data valid
afi_addr	Input	Address
afi_ba	Input	Bank address
afi_cke	Input	Clock enable
afi_cs_n	Input	Chip select
afi_rm	Input	Rank multiplication for LRDIMM

continued...

Port Name	Direction	Description
afi_odt	Input	On-die termination
afi_ras_n	Input	RAS command
afi_cas_n	Input	CAS command
afi_we_n	Input	WE command
afi_rst_n	Input	Asynchronous reset
afi_dm	Input	Write data mask
afi_dqs_burst	Input	Asserted by the controller to enable the output DQS signal
afi_wdata_valid	Input	Asserted by the controller to indicate that afi_wdata contains valid write data
afi_wdata	Input	Write data
afi_rdata_en_full	Input	Asserted by the controller to indicate the amount of relevant read data expected
afi_rdata	Output	Read data
afi_rdata_valid	Output	Asserted by the PHY to indicate that afi_rdata contains valid read data
afi_rrank	Input	Asserted by the controller to indicate which rank is being read from, to control shadow register switching
afi_wrank	Input	Asserted by the controller to indicate which rank is being written to, to control shadow register switching

4.1.1.16. emif_usr_reset_n for DDR3

User clock domain reset interface

Table 27. Interface: emif_usr_reset_n

Interface type: Reset Output

Port Name	Direction	Description
emif_usr_reset_n	Output	Reset for the user clock domain. Asynchronous assertion and synchronous deassertion

4.1.1.17. emif_usr_clk for DDR3

User clock interface

Table 28. Interface: emif_usr_clk

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk	Output	User clock domain

4.1.1.18. emif_usr_reset_n_sec for DDR3

User clock domain reset interface (for the secondary interface in ping-pong configuration)

Table 29. Interface: emif_usr_reset_n_sec

Interface type: Reset Output

Port Name	Direction	Description
emif_usr_reset_n_sec	Output	Reset for the user clock domain. Asynchronous assertion and synchronous deassertion. Intended for the secondary interface in a ping-pong configuration.

4.1.1.19. emif_usr_clk_sec for DDR3

User clock interface (for the secondary interface in ping-pong configuration)

Table 30. Interface: emif_usr_clk_sec

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk_sec	Output	User clock domain. Intended for the secondary interface in a ping-pong configuration.

4.1.1.20. cal_debug_reset_n for DDR3

User calibration debug clock domain reset interface

Table 31. Interface: cal_debug_reset_n

Interface type: Reset Input

Port Name	Direction	Description
cal_debug_reset_n	Input	Reset for the user clock connecting to the Avalon calibration debug bus. Asynchronous assertion and synchronous deassertion

4.1.1.21. cal_debug_clk for DDR3

User calibration debug clock interface

Table 32. Interface: cal_debug_clk

Interface type: Clock Input

Port Name	Direction	Description
cal_debug_clk	Input	User clock domain

4.1.1.22. cal_debug_out_reset_n for DDR3

User calibration debug clock domain reset interface

Table 33. Interface: cal_debug_out_reset_n

Interface type: Reset Output

Port Name	Direction	Description
cal_debug_out_reset_n	Output	Reset for the user clock connecting to the Avalon calibration debug_out bus. Asynchronous assertion and synchronous deassertion

4.1.1.23. cal_debug_out_clk for DDR3

User calibration debug clock interface

Table 34. Interface: cal_debug_out_clk

Interface type: Clock Output

Port Name	Direction	Description
cal_debug_out_clk	Output	User clock domain

4.1.1.24. clks_sharing_master_out for DDR3

Core clocks sharing master interface

Table 35. Interface: clks_sharing_master_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_master_out	Output	This port should fanout to all the core clocks sharing slaves.

4.1.1.25. clks_sharing_slave_in for DDR3

Core clocks sharing slave input interface

Table 36. Interface: clks_sharing_slave_in

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_in	Input	This port should be connected to the core clocks sharing master.

4.1.1.26. clks_sharing_slave_out for DDR3

Core clocks sharing slave output interface

Table 37. Interface: clks_sharing_slave_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_out	Output	This port may be used to fanout to another core clocks sharing slave. Alternatively, the master can fanout to all slaves.

4.1.1.27. ctrl_amm for DDR3

Controller Avalon Memory-Mapped interface

Table 38. Interface: ctrl_amm

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
amm_ready	Output	Wait-request is asserted when controller is busy
amm_read	Input	Read request signal
amm_write	Input	Write request signal
amm_address	Input	Address for the read/write request
amm_readdata	Output	Read data
amm_writedata	Input	Write data
amm_burstcount	Input	Number of transfers in each read/write burst
amm_byteenable	Input	Byte-enable for write data
amm_beginbursttransfer	Input	Indicates when a burst is starting
amm_readdatavalid	Output	Indicates whether read data is valid

4.1.1.28. ctrl_auto_precharge for DDR3

Controller auto-precharge interface

Table 39. Interface: ctrl_auto_precharge

Interface type: Conduit

Port Name	Direction	Description
ctrl_auto_precharge_req	Input	When asserted high along with a read or write request to the memory controller, indicates that the controller should close the currently opened page after the read or write burst.

4.1.1.29. ctrl_user_priority for DDR3

Controller user-requested priority interface

Table 40. Interface: ctrl_user_priority

Interface type: Conduit

Port Name	Direction	Description
ctrl_user_priority_hi	Input	When asserted high along with a read or write request to the memory controller, indicates that the request is high priority and should be fulfilled before other low priority requests.

4.1.1.30. ctrl_ecc_user_interrupt for DDR3

Controller ECC user interrupt interface

Table 41. Interface: ctrl_ecc_user_interrupt

Interface type: Conduit

Port Name	Direction	Description
ctrl_ecc_user_interrupt	Output	Controller ECC user interrupt signal to determine whether there is a bit error

4.1.1.31. ctrl_ecc_readdataerror for DDR3

Controller ECC read data error indication interface

Table 42. Interface: ctrl_ecc_readdataerror

Interface type: Conduit

Port Name	Direction	Description
ctrl_ecc_readdataerror	Output	Signal is asserted high by the controller ECC logic to indicate that the read data has an uncorrectable error. The signal has the same timing as the read data valid signal of the Controller Avalon Memory-Mapped interface.

4.1.1.32. ctrl_ecc_status for DDR3

Controller ECC status interface

Table 43. Interface: ctrl_ecc_status

Interface type: Conduit

Port Name	Direction	Description
ctrl_ecc_sts_intr	Output	ECC interrupt status - '1' indicates interrupt occurred; in case of ping-pong PHY, status from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_sbe_error	Output	'1' indicates SBE occurred; in case of ping-pong PHY, status from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_dbe_error	Output	'1' indicates DBE occurred; in case of ping-pong PHY, status from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_corr_dropped	Output	Correction command dropped status, '1' indicates correction command dropped; in case of ping-pong PHY, status from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_sbe_count	Output	Number of times SBE error occurred; in case of ping-pong PHY, results from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_dbe_count	Output	Number of times DBE error occurred; in case of ping-pong PHY, results from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB

continued...

Port Name	Direction	Description
ctrl_ecc_sts_corr_dropped_count	Output	Number of times correction command dropped; in case of ping-pong PHY, results from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_err_addr	Output	Address of the most recent SBE or DBE; in case of ping-pong PHY, addresses from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_corr_dropped_addr	Output	Address of the most recent correction command dropped; in case of ping-pong PHY, addresses from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB

4.1.1.33. ctrl_mmr_slave for DDR3

Controller MMR slave interface

Table 44. Interface: ctrl_mmr_slave

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
mmr_slave_waitrequest	Output	Wait-request is asserted when controller MMR interface is busy
mmr_slave_read	Input	MMR read request signal
mmr_slave_write	Input	MMR write request signal
mmr_slave_address	Input	Word address for MMR interface of memory controller
mmr_slave_readdata	Output	MMR read data
mmr_slave_writedata	Input	MMR write data
mmr_slave_burstcount	Input	Number of transfers in each read/write burst
mmr_slave_beginbursttransfer	Input	Indicates when a burst is starting
mmr_slave_readdatavalid	Output	Indicates whether MMR read data is valid

4.1.1.34. hps_emif for DDR3

Conduit between Hard Processor Subsystem and memory interface

Table 45. Interface: hps_emif

Interface type: Conduit

Port Name	Direction	Description
hps_to_emif	Input	Signals coming from Hard Processor Subsystem to the memory interface
emif_to_hps	Output	Signals going to Hard Processor Subsystem from the memory interface
hps_to_emif_gp	Input	Signals coming from Hard Processor Subsystem GPIO to the memory interface
emif_to_hps_gp	Output	Signals going to Hard Processor Subsystem GPIO from the memory interface

4.1.1.35. cal_debug for DDR3

Calibration debug interface

Table 46. Interface: cal_debug

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
cal_debug_waitrequest	Output	Wait-request is asserted when controller is busy
cal_debug_read	Input	Read request signal
cal_debug_write	Input	Write request signal
cal_debug_addr	Input	Address for the read/write request
cal_debug_read_data	Output	Read data
cal_debug_write_data	Input	Write data
cal_debug_byteenable	Input	Byte-enable for write data
cal_debug_read_data_valid	Output	Indicates whether read data is valid

4.1.1.36. cal_debug_out for DDR3

Calibration debug interface

Table 47. Interface: cal_debug_out

Interface type: Avalon Memory-Mapped Master

Port Name	Direction	Description
cal_debug_out_waitrequest	Input	Wait-request is asserted when controller is busy
cal_debug_out_read	Output	Read request signal
cal_debug_out_write	Output	Write request signal
cal_debug_out_addr	Output	Address for the read/write request
cal_debug_out_read_data	Input	Read data
cal_debug_out_write_data	Output	Write data
cal_debug_out_byteenable	Output	Byte-enable for write data
cal_debug_out_read_data_valid	Input	Indicates whether read data is valid

4.1.2. Intel Stratix 10 EMIF IP Interfaces for DDR4

The interfaces in the Intel Stratix 10 External Memory Interface IP each have signals that can be connected in Platform Designer. The following table lists the interfaces and corresponding interface types for DDR4.

Table 48. Interfaces for DDR4

Interface Name	Interface Type	Description
local_reset_req	Conduit	Local reset request. Output signal from local_reset_combiner
local_reset_status	Conduit	Local reset status. Input signal to the local_reset_combiner
pll_ref_clk	Clock Input	PLL reference clock input
pll_locked	Conduit	PLL locked signal
pll_extra_clk_0	Clock Output	Additional core clock 0
pll_extra_clk_1	Clock Output	Additional core clock 1
pll_extra_clk_2	Clock Output	Additional core clock 2
pll_extra_clk_3	Clock Output	Additional core clock 3
ac_parity_err	Output	PORT_AC_PARITY_STATE_DESC
oct	Conduit	On-Chip Termination (OCT) interface
mem	Conduit	Interface between FPGA and external memory
status	Conduit	PHY calibration status interface
afi_reset_n	Reset Output	AFI reset interface
afi_clk	Clock Output	AFI clock interface
afi_half_clk	Clock Output	AFI half-rate clock interface
afi	Conduit	Altera PHY Interface (AFI)
emif_usr_reset_n	Reset Output	User clock domain reset interface
emif_usr_clk	Clock Output	User clock interface
emif_usr_reset_n_sec	Reset Output	User clock domain reset interface (for the secondary interface in ping-pong configuration)
emif_usr_clk_sec	Clock Output	User clock interface (for the secondary interface in ping-pong configuration)
cal_debug_reset_n	Reset Input	User calibration debug clock domain reset interface
cal_debug_clk	Clock Input	User calibration debug clock interface
cal_debug_out_reset_n	Reset Output	User calibration debug clock domain reset interface
cal_debug_out_clk	Clock Output	User calibration debug clock interface
clks_sharing_master_out	Conduit	Core clocks sharing master interface
clks_sharing_slave_in	Conduit	Core clocks sharing slave input interface
clks_sharing_slave_out	Conduit	Core clocks sharing slave output interface
ctrl_amm	Avalon Memory-Mapped Slave	Controller Avalon Memory-Mapped interface
ctrl_auto_precharge	Conduit	Controller auto-precharge interface
ctrl_user_priority	Conduit	Controller user-requested priority interface
ctrl_ecc_user_interrupt	Conduit	Controller ECC user interrupt interface
ctrl_ecc_readdataerror	Conduit	Controller ECC read data error indication interface
ctrl_ecc_status	Conduit	Controller ECC status interface

continued...

Interface Name	Interface Type	Description
ctrl_mmr_slave	Avalon Memory-Mapped Slave	Controller MMR slave interface
hps_emif	Conduit	Conduit between Hard Processor Subsystem and memory interface
cal_debug	Avalon Memory-Mapped Slave	Calibration debug interface
cal_debug_out	Avalon Memory-Mapped Master	Calibration debug interface

4.1.2.1. local_reset_req for DDR4

Local reset request. Output signal from local_reset_combiner

Table 49. Interface: local_reset_req

Interface type: Conduit

Port Name	Direction	Description
local_reset_req	Input	Signal from user logic to request the memory interface to be reset and recalibrated. Reset request is sent by transitioning the local_reset_req signal from low to high, then keeping the signal at the high state for a minimum of 2 EMIF core clock cycles, then transitioning the signal from high to low. local_reset_req is asynchronous in that there is no setup/hold timing to meet, but it must meet the minimum pulse width requirement of 2 EMIF core clock cycles.

4.1.2.2. local_reset_status for DDR4

Local reset status. Input signal to the local_reset_combiner

Table 50. Interface: local_reset_status

Interface type: Conduit

Port Name	Direction	Description
local_reset_done	Output	Signal from memory interface to indicate whether it has completed a reset sequence, is currently out of reset, and is ready for a new reset request. When local_reset_done is low, the memory interface is in reset.

4.1.2.3. pll_ref_clk for DDR4

PLL reference clock input

Table 51. Interface: pll_ref_clk

Interface type: Clock Input

Port Name	Direction	Description
pll_ref_clk	Input	PLL reference clock input

4.1.2.4. pll_locked for DDR4

PLL locked signal

Table 52. Interface: pll_locked

Interface type: Conduit

Port Name	Direction	Description
pll_locked	Output	PLL lock signal to indicate whether the PLL has locked

4.1.2.5. pll_extra_clk_0 for DDR4

Additional core clock 0

Table 53. Interface: pll_extra_clk_0

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_0	Output	PLL extra core clock signal output 0. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.2.6. pll_extra_clk_1 for DDR4

Additional core clock 1

Table 54. Interface: pll_extra_clk_1

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_1	Output	PLL extra core clock signal output 1. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.2.7. pll_extra_clk_2 for DDR4

Additional core clock 2

Table 55. Interface: pll_extra_clk_2

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_2	Output	PLL extra core clock signal output 2. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock

Port Name	Direction	Description
		domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.2.8. pll_extra_clk_3 for DDR4

Additional core clock 3

Table 56. Interface: pll_extra_clk_3

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_3	Output	PLL extra core clock signal output 3. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.2.9. ac_parity_err for DDR4

Table 57. Interface: ac_parity_err

Interface type: Conduit

Port Name	Direction	Description
ac_parity_err	Output	PORT_AC_PARITY_STATE_DESC

4.1.2.10. oct for DDR4

On-Chip Termination (OCT) interface

Table 58. Interface: oct

Interface type: Conduit

Port Name	Direction	Description
oct_rzqin	Input	Calibrated On-Chip Termination (OCT) RZQ input pin

4.1.2.11. mem for DDR4

Interface between FPGA and external memory

Table 59. Interface: mem

Interface type: Conduit

Port Name	Direction	Description
mem_ck	Output	CK clock
mem_ck_n	Output	CK clock (negative leg)
mem_a	Output	Address. Address bit A17 is defined only for the x4 configuration of 16 Gb SDRAM.
mem_ba	Output	Bank address
mem_bg	Output	Bank group
mem_cke	Output	Clock enable
mem_cs_n	Output	Chip select
mem_odt	Output	On-die termination
mem_reset_n	Output	Asynchronous reset
mem_act_n	Output	Activation command
mem_par	Output	Command and address parity
mem_dq	Bidirectional	Read/write data
mem_dbi_n	Bidirectional	Acts as either the data bus inversion pin, or the data mask pin, depending on configuration.
mem_dqs	Bidirectional	Data strobe
mem_dqs_n	Bidirectional	Data strobe (negative leg)
mem_alert_n	Input	Alert flag

4.1.2.12. status for DDR4

PHY calibration status interface

Table 60. Interface: status

Interface type: Conduit

Port Name	Direction	Description
local_cal_success	Output	When high, indicates that PHY calibration was successful
local_cal_fail	Output	When high, indicates that PHY calibration failed

4.1.2.13. afi_reset_n for DDR4

AFI reset interface

Table 61. Interface: afi_reset_n

Interface type: Reset Output

Port Name	Direction	Description
afi_reset_n	Output	Reset for the AFI clock domain. Asynchronous assertion and synchronous deassertion

4.1.2.14. afi_clk for DDR4

AFI clock interface

Table 62. Interface: afi_clk

Interface type: Clock Output

Port Name	Direction	Description
afi_clk	Output	Clock for the Altera PHY Interface (AFI)

4.1.2.15. afi_half_clk for DDR4

AFI half-rate clock interface

Table 63. Interface: afi_half_clk

Interface type: Clock Output

Port Name	Direction	Description
afi_half_clk	Output	Clock running at half the frequency of the AFI clock afi_clk

4.1.2.16. afi for DDR4

Altera PHY Interface (AFI)

Table 64. Interface: afi

Interface type: Conduit

Port Name	Direction	Description
afi_cal_success	Output	Signals calibration successful completion
afi_cal_fail	Output	Signals calibration failure
afi_cal_req	Input	When asserted, the interface is recalibrated
afi_rlat	Output	Latency in afi_clk cycles between read command and read data valid
afi_wlat	Output	Latency in afi_clk cycles between write command and write data valid
afi_addr	Input	Address
afi_ba	Input	Bank address
afi_bg	Input	Bank group
afi_cke	Input	Clock enable
afi_cs_n	Input	Chip select
afi_odt	Input	On-die termination
afi_rst_n	Input	Asynchronous reset
afi_act_n	Input	Activation command
afi_par	Input	Command and address parity
afi_dm_n	Input	Write data mask

continued...

Port Name	Direction	Description
afi_dqs_burst	Input	Asserted by the controller to enable the output DQS signal
afi_wdata_valid	Input	Asserted by the controller to indicate that afi_wdata contains valid write data
afi_wdata	Input	Write data
afi_rdata_en_full	Input	Asserted by the controller to indicate the amount of relevant read data expected
afi_rdata	Output	Read data
afi_rdata_valid	Output	Asserted by the PHY to indicate that afi_rdata contains valid read data
afi_rrank	Input	Asserted by the controller to indicate which rank is being read from, to control shadow register switching
afi_wrank	Input	Asserted by the controller to indicate which rank is being written to, to control shadow register switching

4.1.2.17. emif_usr_reset_n for DDR4

User clock domain reset interface

Table 65. Interface: emif_usr_reset_n

Interface type: Reset Output

Port Name	Direction	Description
emif_usr_reset_n	Output	Reset for the user clock domain. Asynchronous assertion and synchronous deassertion

4.1.2.18. emif_usr_clk for DDR4

User clock interface

Table 66. Interface: emif_usr_clk

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk	Output	User clock domain

4.1.2.19. emif_usr_reset_n_sec for DDR4

User clock domain reset interface (for the secondary interface in ping-pong configuration)

Table 67. Interface: emif_usr_reset_n_sec

Interface type: Reset Output

Port Name	Direction	Description
emif_usr_reset_n_sec	Output	Reset for the user clock domain. Asynchronous assertion and synchronous deassertion. Intended for the secondary interface in a ping-pong configuration.

4.1.2.20. emif_usr_clk_sec for DDR4

User clock interface (for the secondary interface in ping-pong configuration)

Table 68. Interface: emif_usr_clk_sec

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk_sec	Output	User clock domain. Intended for the secondary interface in a ping-pong configuration.

4.1.2.21. cal_debug_reset_n for DDR4

User calibration debug clock domain reset interface

Table 69. Interface: cal_debug_reset_n

Interface type: Reset Input

Port Name	Direction	Description
cal_debug_reset_n	Input	Reset for the user clock connecting to the Avalon calibration debug bus. Asynchronous assertion and synchronous deassertion

4.1.2.22. cal_debug_clk for DDR4

User calibration debug clock interface

Table 70. Interface: cal_debug_clk

Interface type: Clock Input

Port Name	Direction	Description
cal_debug_clk	Input	User clock domain

4.1.2.23. cal_debug_out_reset_n for DDR4

User calibration debug clock domain reset interface

Table 71. Interface: cal_debug_out_reset_n

Interface type: Reset Output

Port Name	Direction	Description
cal_debug_out_reset_n	Output	Reset for the user clock connecting to the Avalon calibration debug_out bus. Asynchronous assertion and synchronous deassertion

4.1.2.24. cal_debug_out_clk for DDR4

User calibration debug clock interface

Table 72. Interface: cal_debug_out_clk

Interface type: Clock Output

Port Name	Direction	Description
cal_debug_out_clk	Output	User clock domain

4.1.2.25. clks_sharing_master_out for DDR4

Core clocks sharing master interface

Table 73. Interface: clks_sharing_master_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_master_out	Output	This port should fanout to all the core clocks sharing slaves.

4.1.2.26. clks_sharing_slave_in for DDR4

Core clocks sharing slave input interface

Table 74. Interface: clks_sharing_slave_in

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_in	Input	This port should be connected to the core clocks sharing master.

4.1.2.27. clks_sharing_slave_out for DDR4

Core clocks sharing slave output interface

Table 75. Interface: clks_sharing_slave_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_out	Output	This port may be used to fanout to another core clocks sharing slave. Alternatively, the master can fanout to all slaves.

4.1.2.28. ctrl_amm for DDR4

Controller Avalon Memory-Mapped interface

Table 76. Interface: ctrl_amm

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
amm_ready	Output	Wait-request is asserted when controller is busy
amm_read	Input	Read request signal

continued...

Port Name	Direction	Description
amm_write	Input	Write request signal
amm_address	Input	Address for the read/write request
amm_readdata	Output	Read data
amm_writedata	Input	Write data
amm_burstcount	Input	Number of transfers in each read/write burst
amm_byteenable	Input	Byte-enable for write data
amm_beginbursttransfer	Input	Indicates when a burst is starting
amm_readdatavalid	Output	Indicates whether read data is valid

4.1.2.29. ctrl_auto_precharge for DDR4

Controller auto-precharge interface

Table 77. Interface: ctrl_auto_precharge

Interface type: Conduit

Port Name	Direction	Description
ctrl_auto_precharge_req	Input	When asserted high along with a read or write request to the memory controller, indicates that the controller should close the currently opened page after the read or write burst.

4.1.2.30. ctrl_user_priority for DDR4

Controller user-requested priority interface

Table 78. Interface: ctrl_user_priority

Interface type: Conduit

Port Name	Direction	Description
ctrl_user_priority_hi	Input	When asserted high along with a read or write request to the memory controller, indicates that the request is high priority and should be fulfilled before other low priority requests.

4.1.2.31. ctrl_ecc_user_interrupt for DDR4

Controller ECC user interrupt interface

Table 79. Interface: ctrl_ecc_user_interrupt

Interface type: Conduit

Port Name	Direction	Description
ctrl_ecc_user_interrupt	Output	Controller ECC user interrupt signal to determine whether there is a bit error

4.1.2.32. ctrl_ecc_readdataerror for DDR4

Controller ECC read data error indication interface

Table 80. Interface: ctrl_ecc_readdataerror

Interface type: Conduit

Port Name	Direction	Description
ctrl_ecc_readdataerror	Output	Signal is asserted high by the controller ECC logic to indicate that the read data has an uncorrectable error. The signal has the same timing as the read data valid signal of the Controller Avalon Memory-Mapped interface.

4.1.2.33. ctrl_ecc_status for DDR4

Controller ECC status interface

Table 81. Interface: ctrl_ecc_status

Interface type: Conduit

Port Name	Direction	Description
ctrl_ecc_sts_intr	Output	ECC interrupt status - '1' indicates interrupt occurred; in case of ping-pong PHY, status from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_sbe_error	Output	'1' indicates SBE occurred; in case of ping-pong PHY, status from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_dbe_error	Output	'1' indicates DBE occurred; in case of ping-pong PHY, status from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_corr_dropped	Output	Correction command dropped status, '1' indicates correction command dropped; in case of ping-pong PHY, status from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_sbe_count	Output	Number of times SBE error occurred; in case of ping-pong PHY, results from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_dbe_count	Output	Number of times DBE error occurred; in case of ping-pong PHY, results from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_corr_dropped_count	Output	Number of times correction command dropped; in case of ping-pong PHY, results from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_err_addr	Output	Address of the most recent SBE or DBE; in case of ping-pong PHY, addresses from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB
ctrl_ecc_sts_corr_dropped_addr	Output	Address of the most recent correction command dropped; in case of ping-pong PHY, addresses from two interfaces are concatenated as a double-width port - interface 0 at LSB, interface 1 at MSB

4.1.2.34. ctrl_mmr_slave for DDR4

Controller MMR slave interface

Table 82. Interface: ctrl_mmr_slave

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
mmr_slave_waitrequest	Output	Wait-request is asserted when controller MMR interface is busy
mmr_slave_read	Input	MMR read request signal
mmr_slave_write	Input	MMR write request signal
mmr_slave_address	Input	Word address for MMR interface of memory controller
mmr_slave_readdata	Output	MMR read data
mmr_slave_writedata	Input	MMR write data
mmr_slave_burstcount	Input	Number of transfers in each read/write burst
mmr_slave_beginbursttransfer	Input	Indicates when a burst is starting
mmr_slave_readdatavalid	Output	Indicates whether MMR read data is valid

4.1.2.35. hps_emif for DDR4

Conduit between Hard Processor Subsystem and memory interface

Table 83. Interface: hps_emif

Interface type: Conduit

Port Name	Direction	Description
hps_to_emif	Input	Signals coming from Hard Processor Subsystem to the memory interface
emif_to_hps	Output	Signals going to Hard Processor Subsystem from the memory interface
hps_to_emif_gp	Input	Signals coming from Hard Processor Subsystem GPIO to the memory interface
emif_to_hps_gp	Output	Signals going to Hard Processor Subsystem GPIO from the memory interface

4.1.2.36. cal_debug for DDR4

Calibration debug interface

Table 84. Interface: cal_debug

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
cal_debug_waitrequest	Output	Wait-request is asserted when controller is busy
cal_debug_read	Input	Read request signal

continued...

Port Name	Direction	Description
cal_debug_write	Input	Write request signal
cal_debug_addr	Input	Address for the read/write request
cal_debug_read_data	Output	Read data
cal_debug_write_data	Input	Write data
cal_debug_byteenable	Input	Byte-enable for write data
cal_debug_read_data_valid	Output	Indicates whether read data is valid

4.1.2.37. cal_debug_out for DDR4

Calibration debug interface

Table 85. Interface: cal_debug_out

Interface type: Avalon Memory-Mapped Master

Port Name	Direction	Description
cal_debug_out_waitrequest	Input	Wait-request is asserted when controller is busy
cal_debug_out_read	Output	Read request signal
cal_debug_out_write	Output	Write request signal
cal_debug_out_addr	Output	Address for the read/write request
cal_debug_out_read_data	Input	Read data
cal_debug_out_write_data	Output	Write data
cal_debug_out_byteenable	Output	Byte-enable for write data
cal_debug_out_read_data_valid	Input	Indicates whether read data is valid

4.1.3. Intel Stratix 10 EMIF IP Interfaces for QDR II/II+/II+ Xtreme

The interfaces in the Intel Stratix 10 External Memory Interface IP each have signals that can be connected in Platform Designer. The following table lists the interfaces and corresponding interface types for QDR II/II+/II+ Xtreme.

Table 86. Interfaces for QDR II/II+/II+ Xtreme

Interface Name	Interface Type	Description
local_reset_req	Conduit	Local reset request. Output signal from local_reset_combiner
local_reset_status	Conduit	Local reset status. Input signal to the local_reset_combiner
pll_ref_clk	Clock Input	PLL reference clock input
pll_locked	Conduit	PLL locked signal
pll_extra_clk_0	Clock Output	Additional core clock 0
pll_extra_clk_1	Clock Output	Additional core clock 1
pll_extra_clk_2	Clock Output	Additional core clock 2

continued...

Interface Name	Interface Type	Description
pll_extra_clk_3	Clock Output	Additional core clock 3
oct	Conduit	On-Chip Termination (OCT) interface
mem	Conduit	Interface between FPGA and external memory
status	Conduit	PHY calibration status interface
emif_usr_reset_n	Reset Output	User clock domain reset interface
emif_usr_clk	Clock Output	User clock interface
cal_debug_reset_n	Reset Input	User calibration debug clock domain reset interface
cal_debug_clk	Clock Input	User calibration debug clock interface
cal_debug_out_reset_n	Reset Output	User calibration debug clock domain reset interface
cal_debug_out_clk	Clock Output	User calibration debug clock interface
clks_sharing_master_out	Conduit	Core clocks sharing master interface
clks_sharing_slave_in	Conduit	Core clocks sharing slave input interface
clks_sharing_slave_out	Conduit	Core clocks sharing slave output interface
ctrl_amm	Avalon Memory-Mapped Slave	Controller Avalon Memory-Mapped interface
cal_debug	Avalon Memory-Mapped Slave	Calibration debug interface
cal_debug_out	Avalon Memory-Mapped Master	Calibration debug interface

4.1.3.1. local_reset_req for QDR II/II+/II+ Xtreme

Local reset request. Output signal from local_reset_combiner

Table 87. Interface: local_reset_req

Interface type: Conduit

Port Name	Direction	Description
local_reset_req	Input	Signal from user logic to request the memory interface to be reset and recalibrated. Reset request is sent by transitioning the local_reset_req signal from low to high, then keeping the signal at the high state for a minimum of 2 EMIF core clock cycles, then transitioning the signal from high to low. local_reset_req is asynchronous in that there is no setup/hold timing to meet, but it must meet the minimum pulse width requirement of 2 EMIF core clock cycles.

4.1.3.2. local_reset_status for QDR II/II+/II+ Xtreme

Local reset status. Input signal to the local_reset_combiner

Table 88. Interface: local_reset_status

Interface type: Conduit

Port Name	Direction	Description
local_reset_done	Output	Signal from memory interface to indicate whether it has completed a reset sequence, is currently out of reset, and is ready for a new reset request. When local_reset_done is low, the memory interface is in reset.

4.1.3.3. pll_ref_clk for QDR II/II+/II+ Xtreme

PLL reference clock input

Table 89. Interface: pll_ref_clk

Interface type: Clock Input

Port Name	Direction	Description
pll_ref_clk	Input	PLL reference clock input

4.1.3.4. pll_locked for QDR II/II+/II+ Xtreme

PLL locked signal

Table 90. Interface: pll_locked

Interface type: Conduit

Port Name	Direction	Description
pll_locked	Output	PLL lock signal to indicate whether the PLL has locked

4.1.3.5. pll_extra_clk_0 for QDR II/II+/II+ Xtreme

Additional core clock 0

Table 91. Interface: pll_extra_clk_0

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_0	Output	PLL extra core clock signal output 0. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.3.6. pll_extra_clk_1 for QDR II/II+/II+ Xtreme

Additional core clock 1

Table 92. Interface: pll_extra_clk_1

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_1	Output	PLL extra core clock signal output 1. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.3.7. pll_extra_clk_2 for QDR II/II+/II+ Xtreme

Additional core clock 2

Table 93. Interface: pll_extra_clk_2

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_2	Output	PLL extra core clock signal output 2. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.3.8. pll_extra_clk_3 for QDR II/II+/II+ Xtreme

Additional core clock 3

Table 94. Interface: pll_extra_clk_3

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_3	Output	PLL extra core clock signal output 3. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.3.9. oct for QDR II/II+/II+ Xtreme

On-Chip Termination (OCT) interface

Table 95. Interface: oct

Interface type: Conduit

Port Name	Direction	Description
oct_rzqin	Input	Calibrated On-Chip Termination (OCT) RZQ input pin

4.1.3.10. mem for QDR II/II+/II+ Xtreme

Interface between FPGA and external memory

Table 96. Interface: mem

Interface type: Conduit

Port Name	Direction	Description
mem_k	Output	K clock
mem_k_n	Output	K clock (negative leg)
mem_a	Output	Address
mem_wps_n	Output	Write port select
mem_rps_n	Output	Read port select
mem_doff_n	Output	DLL turn off
mem_bws_n	Output	Byte write select
mem_d	Output	Write data
mem_q	Input	Read data
mem_cq	Input	Echo clock
mem_cq_n	Input	Echo clock (negative leg)

4.1.3.11. status for QDR II/II+/II+ Xtreme

PHY calibration status interface

Table 97. Interface: status

Interface type: Conduit

Port Name	Direction	Description
local_cal_success	Output	When high, indicates that PHY calibration was successful
local_cal_fail	Output	When high, indicates that PHY calibration failed

4.1.3.12. emif_usr_reset_n for QDR II/II+/II+ Xtreme

User clock domain reset interface

Table 98. Interface: emif_usr_reset_n

Interface type: Reset Output

Port Name	Direction	Description
emif_usr_reset_n	Output	Reset for the user clock domain. Asynchronous assertion and synchronous deassertion

4.1.3.13. emif_usr_clk for QDR II/II+/II+ Xtreme

User clock interface

Table 99. Interface: emif_usr_clk

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk	Output	User clock domain

4.1.3.14. cal_debug_reset_n for QDR II/II+/II+ Xtreme

User calibration debug clock domain reset interface

Table 100. Interface: cal_debug_reset_n

Interface type: Reset Input

Port Name	Direction	Description
cal_debug_reset_n	Input	Reset for the user clock connecting to the Avalon calibration debug bus. Asynchronous assertion and synchronous deassertion

4.1.3.15. cal_debug_clk for QDR II/II+/II+ Xtreme

User calibration debug clock interface

Table 101. Interface: cal_debug_clk

Interface type: Clock Input

Port Name	Direction	Description
cal_debug_clk	Input	User clock domain

4.1.3.16. cal_debug_out_reset_n for QDR II/II+/II+ Xtreme

User calibration debug clock domain reset interface

Table 102. Interface: cal_debug_out_reset_n

Interface type: Reset Output

Port Name	Direction	Description
cal_debug_out_reset_n	Output	Reset for the user clock connecting to the Avalon calibration debug_out bus. Asynchronous assertion and synchronous deassertion

4.1.3.17. cal_debug_out_clk for QDR II/II+/II+ Xtreme

User calibration debug clock interface

Table 103. Interface: cal_debug_out_clk

Interface type: Clock Output

Port Name	Direction	Description
cal_debug_out_clk	Output	User clock domain

4.1.3.18. clks_sharing_master_out for QDR II/II+/II+ Xtreme

Core clocks sharing master interface

Table 104. Interface: clks_sharing_master_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_master_out	Output	This port should fanout to all the core clocks sharing slaves.

4.1.3.19. clks_sharing_slave_in for QDR II/II+/II+ Xtreme

Core clocks sharing slave input interface

Table 105. Interface: clks_sharing_slave_in

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_in	Input	This port should be connected to the core clocks sharing master.

4.1.3.20. clks_sharing_slave_out for QDR II/II+/II+ Xtreme

Core clocks sharing slave output interface

Table 106. Interface: clks_sharing_slave_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_out	Output	This port may be used to fanout to another core clocks sharing slave. Alternatively, the master can fanout to all slaves.

4.1.3.21. ctrl_amm for QDR II/II+/II+ Xtreme

Controller Avalon Memory-Mapped interface

Table 107. Interface: ctrl_amm

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
amm_ready	Output	Wait-request is asserted when controller is busy
amm_read	Input	Read request signal
amm_write	Input	Write request signal
amm_address	Input	Address for the read/write request
amm_readdata	Output	Read data
amm_writedata	Input	Write data
amm_burstcount	Input	Number of transfers in each read/write burst

continued...

Port Name	Direction	Description
amm_byteenable	Input	Byte-enable for write data
amm_beginbursttransfer	Input	Indicates when a burst is starting
amm_readdatavalid	Output	Indicates whether read data is valid

4.1.3.22. cal_debug for QDR II/II+/II+ Xtreme

Calibration debug interface

Table 108. Interface: cal_debug

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
cal_debug_waitrequest	Output	Wait-request is asserted when controller is busy
cal_debug_read	Input	Read request signal
cal_debug_write	Input	Write request signal
cal_debug_addr	Input	Address for the read/write request
cal_debug_read_data	Output	Read data
cal_debug_write_data	Input	Write data
cal_debug_byteenable	Input	Byte-enable for write data
cal_debug_read_data_valid	Output	Indicates whether read data is valid

4.1.3.23. cal_debug_out for QDR II/II+/II+ Xtreme

Calibration debug interface

Table 109. Interface: cal_debug_out

Interface type: Avalon Memory-Mapped Master

Port Name	Direction	Description
cal_debug_out_waitrequest	Input	Wait-request is asserted when controller is busy
cal_debug_out_read	Output	Read request signal
cal_debug_out_write	Output	Write request signal
cal_debug_out_addr	Output	Address for the read/write request
cal_debug_out_read_data	Input	Read data
cal_debug_out_write_data	Output	Write data
cal_debug_out_byteenable	Output	Byte-enable for write data
cal_debug_out_read_data_valid	Input	Indicates whether read data is valid

4.1.4. Intel Stratix 10 EMIF IP Interfaces for QDR-IV

The interfaces in the Intel Stratix 10 External Memory Interface IP each have signals that can be connected in Platform Designer. The following table lists the interfaces and corresponding interface types for QDR-IV.

Table 110. Interfaces for QDR-IV

Interface Name	Interface Type	Description
local_reset_req	Conduit	Local reset request. Output signal from local_reset_combiner
local_reset_status	Conduit	Local reset status. Input signal to the local_reset_combiner
pll_ref_clk	Clock Input	PLL reference clock input
pll_locked	Conduit	PLL locked signal
pll_extra_clk_0	Clock Output	Additional core clock 0
pll_extra_clk_1	Clock Output	Additional core clock 1
pll_extra_clk_2	Clock Output	Additional core clock 2
pll_extra_clk_3	Clock Output	Additional core clock 3
oct	Conduit	On-Chip Termination (OCT) interface
mem	Conduit	Interface between FPGA and external memory
status	Conduit	PHY calibration status interface
afi_reset_n	Reset Output	AFI reset interface
afi_clk	Clock Output	AFI clock interface
afi_half_clk	Clock Output	AFI half-rate clock interface
afi	Conduit	Altera PHY Interface (AFI)
emif_usr_reset_n	Reset Output	User clock domain reset interface
emif_usr_clk	Clock Output	User clock interface
cal_debug_reset_n	Reset Input	User calibration debug clock domain reset interface
cal_debug_clk	Clock Input	User calibration debug clock interface
cal_debug_out_reset_n	Reset Output	User calibration debug clock domain reset interface
cal_debug_out_clk	Clock Output	User calibration debug clock interface
clks_sharing_master_out	Conduit	Core clocks sharing master interface
clks_sharing_slave_in	Conduit	Core clocks sharing slave input interface
clks_sharing_slave_out	Conduit	Core clocks sharing slave output interface
ctrl_amm	Avalon Memory-Mapped Slave	Controller Avalon Memory-Mapped interface
cal_debug	Avalon Memory-Mapped Slave	Calibration debug interface
cal_debug_out	Avalon Memory-Mapped Master	Calibration debug interface

4.1.4.1. local_reset_req for QDR-IV

Local reset request. Output signal from local_reset_combiner

Table 111. Interface: local_reset_req

Interface type: Conduit

Port Name	Direction	Description
local_reset_req	Input	Signal from user logic to request the memory interface to be reset and recalibrated. Reset request is sent by transitioning the local_reset_req signal from low to high, then keeping the signal at the high state for a minimum of 2 EMIF core clock cycles, then transitioning the signal from high to low. local_reset_req is asynchronous in that there is no setup/hold timing to meet, but it must meet the minimum pulse width requirement of 2 EMIF core clock cycles.

4.1.4.2. local_reset_status for QDR-IV

Local reset status. Input signal to the local_reset_combiner

Table 112. Interface: local_reset_status

Interface type: Conduit

Port Name	Direction	Description
local_reset_done	Output	Signal from memory interface to indicate whether it has completed a reset sequence, is currently out of reset, and is ready for a new reset request. When local_reset_done is low, the memory interface is in reset.

4.1.4.3. pll_ref_clk for QDR-IV

PLL reference clock input

Table 113. Interface: pll_ref_clk

Interface type: Clock Input

Port Name	Direction	Description
pll_ref_clk	Input	PLL reference clock input

4.1.4.4. pll_locked for QDR-IV

PLL locked signal

Table 114. Interface: pll_locked

Interface type: Conduit

Port Name	Direction	Description
pll_locked	Output	PLL lock signal to indicate whether the PLL has locked

4.1.4.5. pll_extra_clk_0 for QDR-IV

Additional core clock 0

Table 115. Interface: pll_extra_clk_0

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_0	Output	PLL extra core clock signal output 0. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.4.6. pll_extra_clk_1 for QDR-IV

Additional core clock 1

Table 116. Interface: pll_extra_clk_1

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_1	Output	PLL extra core clock signal output 1. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.4.7. pll_extra_clk_2 for QDR-IV

Additional core clock 2

Table 117. Interface: pll_extra_clk_2

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_2	Output	PLL extra core clock signal output 2. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.4.8. pll_extra_clk_3 for QDR-IV

Additional core clock 3

Table 118. Interface: pll_extra_clk_3

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_3	Output	PLL extra core clock signal output 3. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock

Port Name	Direction	Description
		domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.4.9. oct for QDR-IV

On-Chip Termination (OCT) interface

Table 119. Interface: oct

Interface type: Conduit

Port Name	Direction	Description
oct_rzqin	Input	Calibrated On-Chip Termination (OCT) RZQ input pin

4.1.4.10. mem for QDR-IV

Interface between FPGA and external memory

Table 120. Interface: mem

Interface type: Conduit

Port Name	Direction	Description
mem_ck	Output	CK clock
mem_ck_n	Output	CK clock (negative leg)
mem_dka	Output	DK clock for port A
mem_dka_n	Output	DK clock for port A (negative leg)
mem_dkb	Output	DK clock for port B
mem_dkb_n	Output	DK clock for port B (negative leg)
mem_a	Output	Address
mem_reset_n	Output	Asynchronous reset
mem_lda_n	Output	Synchronous load for port A
mem_ldb_n	Output	Synchronous load for port B
mem_rwa_n	Output	Synchronous read/write for port A
mem_rwb_n	Output	Synchronous read/write for port B
mem_lbk0_n	Output	Loopback mode
mem_lbk1_n	Output	Loopback mode
mem_cfg_n	Output	Configuration bit
mem_ap	Output	Address parity
mem_ainv	Output	Address inversion
mem_dqa	Bidirectional	Read/write data for port A
mem_dqb	Bidirectional	Read/write data for port B
mem_dinva	Bidirectional	Read/write data inversion for port A

continued...

Port Name	Direction	Description
mem_dinvb	Bidirectional	Read/write data inversion for port B
mem_qka	Input	Read data clock for port A
mem_qka_n	Input	Read data clock for port A (negative leg)
mem_qkb	Input	Read data clock for port B
mem_qkb_n	Input	Read data clock for port B (negative leg)
mem_pe_n	Input	Address parity error flag

4.1.4.11. status for QDR-IV

PHY calibration status interface

Table 121. Interface: status

Interface type: Conduit

Port Name	Direction	Description
local_cal_success	Output	When high, indicates that PHY calibration was successful
local_cal_fail	Output	When high, indicates that PHY calibration failed

4.1.4.12. afi_reset_n for QDR-IV

AFI reset interface

Table 122. Interface: afi_reset_n

Interface type: Reset Output

Port Name	Direction	Description
afi_reset_n	Output	Reset for the AFI clock domain. Asynchronous assertion and synchronous deassertion

4.1.4.13. afi_clk for QDR-IV

AFI clock interface

Table 123. Interface: afi_clk

Interface type: Clock Output

Port Name	Direction	Description
afi_clk	Output	Clock for the Altera PHY Interface (AFI)

4.1.4.14. afi_half_clk for QDR-IV

AFI half-rate clock interface

Table 124. Interface: afi_half_clk

Interface type: Clock Output

Port Name	Direction	Description
afi_half_clk	Output	Clock running at half the frequency of the AFI clock afi_clk

4.1.4.15. afi for QDR-IV

Altera PHY Interface (AFI)

Table 125. Interface: afi

Interface type: Conduit

Port Name	Direction	Description
afi_ld_n	Input	Synchronous load for port A and B
afi_rw_n	Input	Synchronous read/write for port A and B
afi_lbk0_n	Input	Loopback mode
afi_lbk1_n	Input	Loopback mode
afi_cfg_n	Input	Configuration bit
afi_ap	Input	Address parity
afi_ainv	Input	Address inversion
afi_rdata_dinv	Output	Data inversion for read data
afi_wdata_dinv	Input	Data inversion for write data
afi_pe_n	Output	Address parity error flag

4.1.4.16. emif_usr_reset_n for QDR-IV

User clock domain reset interface

Table 126. Interface: emif_usr_reset_n

Interface type: Reset Output

Port Name	Direction	Description
emif_usr_reset_n	Output	Reset for the user clock domain. Asynchronous assertion and synchronous deassertion

4.1.4.17. emif_usr_clk for QDR-IV

User clock interface

Table 127. Interface: emif_usr_clk

Interface type: Clock Output

Port Name	Direction	Description
emif_usr_clk	Output	User clock domain

4.1.4.18. cal_debug_reset_n for QDR-IV

User calibration debug clock domain reset interface

Table 128. Interface: cal_debug_reset_n

Interface type: Reset Input

Port Name	Direction	Description
cal_debug_reset_n	Input	Reset for the user clock connecting to the Avalon calibration debug bus. Asynchronous assertion and synchronous deassertion

4.1.4.19. cal_debug_clk for QDR-IV

User calibration debug clock interface

Table 129. Interface: cal_debug_clk

Interface type: Clock Input

Port Name	Direction	Description
cal_debug_clk	Input	User clock domain

4.1.4.20. cal_debug_out_reset_n for QDR-IV

User calibration debug clock domain reset interface

Table 130. Interface: cal_debug_out_reset_n

Interface type: Reset Output

Port Name	Direction	Description
cal_debug_out_reset_n	Output	Reset for the user clock connecting to the Avalon calibration debug_out bus. Asynchronous assertion and synchronous deassertion

4.1.4.21. cal_debug_out_clk for QDR-IV

User calibration debug clock interface

Table 131. Interface: cal_debug_out_clk

Interface type: Clock Output

Port Name	Direction	Description
cal_debug_out_clk	Output	User clock domain

4.1.4.22. clks_sharing_master_out for QDR-IV

Core clocks sharing master interface

Table 132. Interface: clks_sharing_master_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_master_out	Output	This port should fanout to all the core clocks sharing slaves.

4.1.4.23. clks_sharing_slave_in for QDR-IV

Core clocks sharing slave input interface

Table 133. Interface: clks_sharing_slave_in

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_in	Input	This port should be connected to the core clocks sharing master.

4.1.4.24. clks_sharing_slave_out for QDR-IV

Core clocks sharing slave output interface

Table 134. Interface: clks_sharing_slave_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_out	Output	This port may be used to fanout to another core clocks sharing slave. Alternatively, the master can fanout to all slaves.

4.1.4.25. ctrl_amm for QDR-IV

Controller Avalon Memory-Mapped interface

Table 135. Interface: ctrl_amm

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
amm_ready	Output	Wait-request is asserted when controller is busy
amm_read	Input	Read request signal
amm_write	Input	Write request signal
amm_address	Input	Address for the read/write request
amm_readdata	Output	Read data
amm_writedata	Input	Write data
amm_burstcount	Input	Number of transfers in each read/write burst
amm_beginbursttransfer	Input	Indicates when a burst is starting
amm_readdatavalid	Output	Indicates whether read data is valid

4.1.4.26. cal_debug for QDR-IV

Calibration debug interface

Table 136. Interface: cal_debug

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
cal_debug_waitrequest	Output	Wait-request is asserted when controller is busy
cal_debug_read	Input	Read request signal
cal_debug_write	Input	Write request signal
cal_debug_addr	Input	Address for the read/write request
cal_debug_read_data	Output	Read data
cal_debug_write_data	Input	Write data
cal_debug_byteenable	Input	Byte-enable for write data
cal_debug_read_data_valid	Output	Indicates whether read data is valid

4.1.4.27. cal_debug_out for QDR-IV

Calibration debug interface

Table 137. Interface: cal_debug_out

Interface type: Avalon Memory-Mapped Master

Port Name	Direction	Description
cal_debug_out_waitrequest	Input	Wait-request is asserted when controller is busy
cal_debug_out_read	Output	Read request signal
cal_debug_out_write	Output	Write request signal
cal_debug_out_addr	Output	Address for the read/write request
cal_debug_out_read_data	Input	Read data
cal_debug_out_write_data	Output	Write data
cal_debug_out_byteenable	Output	Byte-enable for write data
cal_debug_out_read_data_valid	Input	Indicates whether read data is valid

4.1.5. Intel Stratix 10 EMIF IP Interfaces for RLDRAM 3

The interfaces in the Intel Stratix 10 External Memory Interface IP each have signals that can be connected in Platform Designer. The following table lists the interfaces and corresponding interface types for RLDRAM 3.

Table 138. Interfaces for RLDRAM 3

Interface Name	Interface Type	Description
local_reset_req	Conduit	Local reset request. Output signal from local_reset_combiner
local_reset_status	Conduit	Local reset status. Input signal to the local_reset_combiner
pll_ref_clk	Clock Input	PLL reference clock input
pll_locked	Conduit	PLL locked signal
pll_extra_clk_0	Clock Output	Additional core clock 0
pll_extra_clk_1	Clock Output	Additional core clock 1
pll_extra_clk_2	Clock Output	Additional core clock 2
pll_extra_clk_3	Clock Output	Additional core clock 3
oct	Conduit	On-Chip Termination (OCT) interface
mem	Conduit	Interface between FPGA and external memory
status	Conduit	PHY calibration status interface
afi_reset_n	Reset Output	AFI reset interface
afi_clk	Clock Output	AFI clock interface
afi_half_clk	Clock Output	AFI half-rate clock interface
afi	Conduit	Altera PHY Interface (AFI)
cal_debug_reset_n	Reset Input	User calibration debug clock domain reset interface
cal_debug_clk	Clock Input	User calibration debug clock interface
cal_debug_out_reset_n	Reset Output	User calibration debug clock domain reset interface
cal_debug_out_clk	Clock Output	User calibration debug clock interface
clks_sharing_master_out	Conduit	Core clocks sharing master interface
clks_sharing_slave_in	Conduit	Core clocks sharing slave input interface
clks_sharing_slave_out	Conduit	Core clocks sharing slave output interface
cal_debug	Avalon Memory-Mapped Slave	Calibration debug interface
cal_debug_out	Avalon Memory-Mapped Master	Calibration debug interface

4.1.5.1. local_reset_req for RLDRAM 3

Local reset request. Output signal from local_reset_combiner

Table 139. Interface: local_reset_req

Interface type: Conduit

Port Name	Direction	Description
local_reset_req	Input	Signal from user logic to request the memory interface to be reset and recalibrated. Reset request is sent by transitioning the local_reset_req signal from low to high, then keeping the signal at the high state for a minimum of 2 EMIF core clock cycles, then transitioning the signal from high to low. local_reset_req is asynchronous in that there is

Port Name	Direction	Description
		no setup/hold timing to meet, but it must meet the minimum pulse width requirement of 2 EMIF core clock cycles.

4.1.5.2. local_reset_status for RLDRAM 3

Local reset status. Input signal to the local_reset_combiner

Table 140. Interface: local_reset_status

Interface type: Conduit

Port Name	Direction	Description
local_reset_done	Output	Signal from memory interface to indicate whether it has completed a reset sequence, is currently out of reset, and is ready for a new reset request. When local_reset_done is low, the memory interface is in reset.

4.1.5.3. pll_ref_clk for RLDRAM 3

PLL reference clock input

Table 141. Interface: pll_ref_clk

Interface type: Clock Input

Port Name	Direction	Description
pll_ref_clk	Input	PLL reference clock input

4.1.5.4. pll_locked for RLDRAM 3

PLL locked signal

Table 142. Interface: pll_locked

Interface type: Conduit

Port Name	Direction	Description
pll_locked	Output	PLL lock signal to indicate whether the PLL has locked

4.1.5.5. pll_extra_clk_0 for RLDRAM 3

Additional core clock 0

Table 143. Interface: pll_extra_clk_0

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_0	Output	PLL extra core clock signal output 0. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock

Port Name	Direction	Description
		domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.5.6. pll_extra_clk_1 for RLDRAM 3

Additional core clock 1

Table 144. Interface: pll_extra_clk_1

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_1	Output	PLL extra core clock signal output 1. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.5.7. pll_extra_clk_2 for RLDRAM 3

Additional core clock 2

Table 145. Interface: pll_extra_clk_2

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_2	Output	PLL extra core clock signal output 2. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.5.8. pll_extra_clk_3 for RLDRAM 3

Additional core clock 3

Table 146. Interface: pll_extra_clk_3

Interface type: Clock Output

Port Name	Direction	Description
pll_extra_clk_3	Output	PLL extra core clock signal output 3. This signal exists if you specify the EMIF PLL to generate additional output clock signals (up to 4) that can be used by user logic. This clock signal is asynchronous to the memory interface core clock domains (such as emif_usr_clk or afi_clk). You must follow proper clock-domain-crossing techniques when transferring data between clock domains.

4.1.5.9. oct for RLDRAM 3

On-Chip Termination (OCT) interface

Table 147. Interface: oct

Interface type: Conduit

Port Name	Direction	Description
oct_rzqin	Input	Calibrated On-Chip Termination (OCT) RZQ input pin

4.1.5.10. mem for RLDRAM 3

Interface between FPGA and external memory

Table 148. Interface: mem

Interface type: Conduit

Port Name	Direction	Description
mem_ck	Output	CK clock
mem_ck_n	Output	CK clock (negative leg)
mem_dk	Output	DK clock
mem_dk_n	Output	DK clock (negative leg)
mem_a	Output	Address
mem_ba	Output	Bank address
mem_cs_n	Output	Chip select
mem_rm	Output	Rank multiplication for LRDIMM. Typically, mem_rm[0] and mem_rm[1] connect to CS2# and CS3# of the memory buffer of all LRDIMM slots.
mem_we_n	Output	WE command
mem_reset_n	Output	Asynchronous reset
mem_ref_n	Output	REF command
mem_dm	Output	Write data mask
mem_dq	Bidirectional	Read/write data
mem_qk	Input	Read data clock
mem_qk_n	Input	Read data clock (negative leg)

4.1.5.11. status for RLDRAM 3

PHY calibration status interface

Table 149. Interface: status

Interface type: Conduit

Port Name	Direction	Description
local_cal_success	Output	When high, indicates that PHY calibration was successful
local_cal_fail	Output	When high, indicates that PHY calibration failed

4.1.5.12. afi_reset_n for RLDRAM 3

AFI reset interface

Table 150. Interface: afi_reset_n

Interface type: Reset Output

Port Name	Direction	Description
afi_reset_n	Output	Reset for the AFI clock domain. Asynchronous assertion and synchronous deassertion

4.1.5.13. afi_clk for RLDRAM 3

AFI clock interface

Table 151. Interface: afi_clk

Interface type: Clock Output

Port Name	Direction	Description
afi_clk	Output	Clock for the Altera PHY Interface (AFI)

4.1.5.14. afi_half_clk for RLDRAM 3

AFI half-rate clock interface

Table 152. Interface: afi_half_clk

Interface type: Clock Output

Port Name	Direction	Description
afi_half_clk	Output	Clock running at half the frequency of the AFI clock afi_clk

4.1.5.15. afi for RLDRAM 3

Altera PHY Interface (AFI)

Table 153. Interface: afi

Interface type: Conduit

Port Name	Direction	Description
afi_cal_success	Output	Signals calibration successful completion
afi_cal_fail	Output	Signals calibration failure

continued...

Port Name	Direction	Description
afi_cal_req	Input	When asserted, the interface is recalibrated
afi_rlat	Output	Latency in afi_clk cycles between read command and read data valid
afi_wlat	Output	Latency in afi_clk cycles between write command and write data valid
afi_addr	Input	Address
afi_ba	Input	Bank address
afi_cs_n	Input	Chip select
afi_we_n	Input	WE command
afi_rst_n	Input	Asynchronous reset
afi_ref_n	Input	REF command
afi_dm	Input	Write data mask
afi_wdata_valid	Input	Asserted by the controller to indicate that afi_wdata contains valid write data
afi_wdata	Input	Write data
afi_rdata_en_full	Input	Asserted by the controller to indicate the amount of relevant read data expected
afi_rdata	Output	Read data
afi_rdata_valid	Output	Asserted by the PHY to indicate that afi_rdata contains valid read data
afi_rrank	Input	Asserted by the controller to indicate which rank is being read from, to control shadow register switching
afi_wrank	Input	Asserted by the controller to indicate which rank is being written to, to control shadow register switching

4.1.5.16. cal_debug_reset_n for RLDRAM 3

User calibration debug clock domain reset interface

Table 154. Interface: cal_debug_reset_n

Interface type: Reset Input

Port Name	Direction	Description
cal_debug_reset_n	Input	Reset for the user clock connecting to the Avalon calibration debug bus. Asynchronous assertion and synchronous deassertion

4.1.5.17. cal_debug_clk for RLDRAM 3

User calibration debug clock interface

Table 155. Interface: cal_debug_clk

Interface type: Clock Input

Port Name	Direction	Description
cal_debug_clk	Input	User clock domain

4.1.5.18. cal_debug_out_reset_n for RLDRAM 3

User calibration debug clock domain reset interface

Table 156. Interface: cal_debug_out_reset_n

Interface type: Reset Output

Port Name	Direction	Description
cal_debug_out_reset_n	Output	Reset for the user clock connecting to the Avalon calibration debug_out bus. Asynchronous assertion and synchronous deassertion

4.1.5.19. cal_debug_out_clk for RLDRAM 3

User calibration debug clock interface

Table 157. Interface: cal_debug_out_clk

Interface type: Clock Output

Port Name	Direction	Description
cal_debug_out_clk	Output	User clock domain

4.1.5.20. clks_sharing_master_out for RLDRAM 3

Core clocks sharing master interface

Table 158. Interface: clks_sharing_master_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_master_out	Output	This port should fanout to all the core clocks sharing slaves.

4.1.5.21. clks_sharing_slave_in for RLDRAM 3

Core clocks sharing slave input interface

Table 159. Interface: clks_sharing_slave_in

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_in	Input	This port should be connected to the core clocks sharing master.

4.1.5.22. clks_sharing_slave_out for RLDRAM 3

Core clocks sharing slave output interface

Table 160. Interface: clks_sharing_slave_out

Interface type: Conduit

Port Name	Direction	Description
clks_sharing_slave_out	Output	This port may be used to fanout to another core clocks sharing slave. Alternatively, the master can fanout to all slaves.

4.1.5.23. cal_debug for RLDRAM 3

Calibration debug interface

Table 161. Interface: cal_debug

Interface type: Avalon Memory-Mapped Slave

Port Name	Direction	Description
cal_debug_waitrequest	Output	Wait-request is asserted when controller is busy
cal_debug_read	Input	Read request signal
cal_debug_write	Input	Write request signal
cal_debug_addr	Input	Address for the read/write request
cal_debug_read_data	Output	Read data
cal_debug_write_data	Input	Write data
cal_debug_byteenable	Input	Byte-enable for write data
cal_debug_read_data_valid	Output	Indicates whether read data is valid

4.1.5.24. cal_debug_out for RLDRAM 3

Calibration debug interface

Table 162. Interface: cal_debug_out

Interface type: Avalon Memory-Mapped Master

Port Name	Direction	Description
cal_debug_out_waitrequest	Input	Wait-request is asserted when controller is busy
cal_debug_out_read	Output	Read request signal
cal_debug_out_write	Output	Write request signal
cal_debug_out_addr	Output	Address for the read/write request
cal_debug_out_read_data	Input	Read data
cal_debug_out_write_data	Output	Write data
cal_debug_out_byteenable	Output	Byte-enable for write data
cal_debug_out_read_data_valid	Input	Indicates whether read data is valid

4.2. AFI Signals

The following tables list Altera PHY interface (AFI) signals grouped according to their functions.

In each table, the **Direction** column denotes the direction of the signal relative to the PHY. For example, a signal defined as an output passes out of the PHY to the controller. The AFI specification does not include any bidirectional signals.

Note: Not all signals listed apply to every device family or every memory protocol.

4.2.1. AFI Clock and Reset Signals

The AFI interface provides up to two clock signals and an asynchronous reset signal.

Table 163. Clock and Reset Signals

Signal Name	Direction	Width	Description
afi_clk	Output	1	Clock with which all data exchanged on the AFI bus is synchronized. In general, this clock is referred to as full-rate, half-rate, or quarter-rate, depending on the ratio between the frequency of this clock and the frequency of the memory device clock.
afi_half_clk	Output	1	Clock signal that runs at half the speed of the afi_clk. The controller uses this signal when the half-rate bridge feature is in use. This signal is optional.
afi_reset_n	Output	1	Asynchronous reset output signal. You must synchronize this signal to the clock domain in which you use it.

4.2.2. AFI Address and Command Signals

The address and command signals for AFI 4.0 encode read/write/configuration commands to send to the memory device. The address and command signals are single-data rate signals.

Table 164. Address and Command Signals

Signal Name	Direction	Width	Description
afi_addr	Input	AFI_ADDR_WIDTH	Address.
afi_bg	Input	AFI_BANKGROUP_WIDTH	Bank group (DDR4 only).
afi_ba	Input	AFI_BANKADDR_WIDTH	Bank address.
afi_cke	Input	AFI_CLK_EN_WIDTH	Clock enable.
afi_cs_n	Input	AFI_CS_WIDTH	Chip select signal. (The number of chip selects may not match the number of ranks; for example, RDIMMs and LRDIMMs require a minimum of 2 chip select signals for both single-rank and dual-rank configurations. Consult your memory device data sheet for information about chip select signal width.)
afi_ras_n	Input	AFI_CONTROL_WIDTH	RAS# (for DDR3 memory devices.)
afi_we_n	Input	AFI_CONTROL_WIDTH	WE# (for DDR3 memory devices.)
afi_rw_n	Input	AFI_CONTROL_WIDTH * 2	RWA/B# (QDR-IV).

continued...

Signal Name	Direction	Width	Description
afi_cas_n	Input	AFI_CONTROL_WIDTH	CAS# (for DDR3 memory devices.)
afi_act_n	Input	AFI_CONTROL_WIDTH	ACT# (DDR4).
afi_rst_n	Input	AFI_CONTROL_WIDTH	RESET# (for DDR3 and DDR4 memory devices.)
afi_odt	Input	AFI_CLK_EN_WIDTH	On-die termination signal for DDR3 memory devices. (Do not confuse this memory device signal with the FPGA's internal on-chip termination signal.)
afi_par	Input	AFI_CS_WIDTH	Address and command parity input. (DDR4) Address parity input. (QDR-IV)
afi_ainv	Input	AFI_CONTROL_WIDTH	Address inversion. (QDR-IV)
afi_mem_clk_disable	Input	AFI_CLK_PAIR_COUNT	When this signal is asserted, mem_clk and mem_clk_n are disabled. This signal is used in low-power mode.
afi_wps_n	Output	AFI_CS_WIDTH	WPS (for QDR II/II+ memory devices.)
afi_rps_n	Output	AFI_CS_WIDTH	RPS (for QDR II/II+ memory devices.)

4.2.3. AFI Write Data Signals

Write Data Signals for AFI 4.0 control the data, data mask, and strobe signals passed to the memory device during write operations.

Table 165. Write Data Signals

Signal Name	Direction	Width	Description
afi_dqs_burst	Input	AFI_RATE_RATIO	Controls the enable on the strobe (DQS) pins for DDR3 memory devices. When this signal is asserted, mem_dqs and mem_dqsn are driven. This signal must be asserted before afi_wdata_valid to implement the write preamble, and must be driven for the correct duration to generate a correctly timed mem_dqs signal.
afi_wdata_valid	Input	AFI_RATE_RATIO	Write data valid signal. This signal controls the output enable on the data and data mask pins.
afi_wdata	Input	AFI_DQ_WIDTH	Write data signal to send to the memory device at double-data rate. This signal controls the PHY's mem_dq output.
afi_dm	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_dm signal for DDR3 memory devices. Also directly controls the PHY's mem_dbi signal for DDR4.

continued...

Signal Name	Direction	Width	Description
			The mem_dm and mem_dbi features share the same port on the memory device.
afi_bws_n	Input	AFI_DM_WIDTH	Data mask. This signal controls the PHY's mem_bws_n signal for QDR II/II+ memory devices.
afi_dinv	Input	AFI_WRITE_DQS_WIDTH * 2	Data inversion. It directly controls the PHY's mem_dinva/b signal for QDR-IV devices.

4.2.4. AFI Read Data Signals

Read Data Signals for AFI 4.0 control the data sent from the memory device during read operations.

Table 166. Read Data Signals

Signal Name	Direction	Width	Description
afi_rdata_en_full	Input	AFI_RATE_RATIO	Read data enable full. Indicates that the memory controller is currently performing a read operation. This signal is held high for the entire read burst.If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).
afi_rdata	Output	AFI_DQ_WIDTH	Read data from the memory device. This data is considered valid only when afi_rdata_valid is asserted by the PHY.
afi_rdata_valid	Output	AFI_RATE_RATIO	Read data valid. When asserted, this signal indicates that the afi_rdata bus is valid.If this signal is aligned to even clock cycles, it is possible to use 1-bit even in half-rate mode (i.e., AFI_RATE=2).

4.2.5. AFI Calibration Status Signals

The PHY instantiates a sequencer which calibrates the memory interface with the memory device and some internal components such as read FIFOs and valid FIFOs. The sequencer reports the results of the calibration process to the controller through the Calibration Status Signals in the AFI interface.

Table 167. Calibration Status Signals

Signal Name	Direction	Width	Description
afi_cal_success	Output	1	Asserted to indicate that calibration has completed successfully.
afi_cal_fail	Output	1	Asserted to indicate that calibration has failed.
afi_cal_req	Input	1	Effectively a synchronous reset for the sequencer. When this signal is asserted, the sequencer returns to the reset state; when this signal is released, a new calibration sequence begins.

continued...

Signal Name	Direction	Width	Description
afi_wlat	Output	AFI_WLAT_WIDTH	The required write latency in afi_clk cycles, between address/command and write data being issued at the PHY/controller interface. The afi_wlat value can be different for different groups; each group's write latency can range from 0 to 63. If write latency is the same for all groups, only the lowest 6 bits are required.
afi_rlat *(1)*	Output	AFI_RLAT_WIDTH	The required read latency in afi_clk cycles between address/command and read data being returned to the PHY/controller interface. Values can range from 0 to 63.

Note to Table:

1. The `afi_rlat` signal is not supported for PHY-only designs. Instead, you can sample the `afi_rdata_valid` signal to determine when valid read data is available.

4.2.6. AFI Shadow Register Management Signals

Shadow registers are a feature that enables high-speed multi-rank support. Shadow registers allow the sequencer to calibrate each rank separately, and save the calibrated settings—such as deskew delay-chain configurations—of each rank in its own set of shadow registers.

During a rank-to-rank switch, the correct set of calibrated settings is restored just in time to optimize the data valid window. The PHY relies on additional AFI signals to control which set of shadow registers to activate.

Table 168. Shadow Register Management Signals

Signal Name	Direction	Width	Description
afi_wrank	Input	AFI_WRANK_WIDTH	Signal from controller specifying which rank the write data is going to. The signal timing is identical to that of afi_dqs_burst. That is, afi_wrank must be asserted at the same time and must last the same duration as the afi_dqs_burst signal.
afi_rrank	Output	AFI_RRANK_WIDTH	Signal from controller specifying which rank is being read. The signal must be asserted at the same time as the afi_rdata_en signal when issuing a read command, but unlike afi_rdata_en, afi_rrank is stateful. That is, once asserted, the signal value must remain unchanged until the controller issues a new read command to a different rank.

Both the afi_wrank and afi_rrank signals encode the rank being accessed using the one-hot scheme (e.g. in a quad-rank interface, 0001, 0010, 0100, 1000 refer to the 1st, 2nd, 3rd, 4th rank respectively). The ordering within the bus is the same as other AFI signals. Specifically the bus is ordered by time slots, for example:

```
Half-rate afi_w/rrank = {T1, T0}
```

```
Quarter-rate afi_w/rrank = {T3, T2, T1, T0}
```

Where T*x* is a number of rank-bit words that one-hot encodes the rank being accessed at the y^{th} full-rate cycle.

Additional Requirements for Shadow Register Support

To ensure that the hardware has enough time to switch from one shadow register to another, the controller must satisfy the following minimum rank-to-rank-switch delays (tRTRS):

- Two read commands going to different ranks must be separated by a minimum of 3 full-rate cycles (in addition to the burst length delay needed to avoid collision of data bursts).
- Two write commands going to different rank must be separated by a minimum of 4 full-rate cycles (in addition to the burst length delay needed to avoid collision of data bursts).

The FPGA device supports a maximum of 4 sets of shadow registers, each for an independent set of timings. More than 4 ranks are supported if those ranks have four or fewer sets of independent timing. For example, the rank multiplication mode of an LRDIMM allows more than one physical rank to share a set of timing data as a single logical rank. Therefore the device can support up to 4 logical ranks, though that means more than 4 physical ranks.

4.3. AFI 4.0 Timing Diagrams

4.3.1. AFI Address and Command Timing Diagrams

Depending on the ratio between the memory clock and the PHY clock, different numbers of bits must be provided per PHY clock on the AFI interface. The following figures illustrate the AFI address/command waveforms in full, half and quarter rate respectively.

The waveforms show how the AFI command phase corresponds to the memory command output. AFI command 0 corresponds to the first memory command slot, AFI command 1 corresponds to the second memory command slot, and so on.

Figure 24. AFI Address and Command Full-Rate

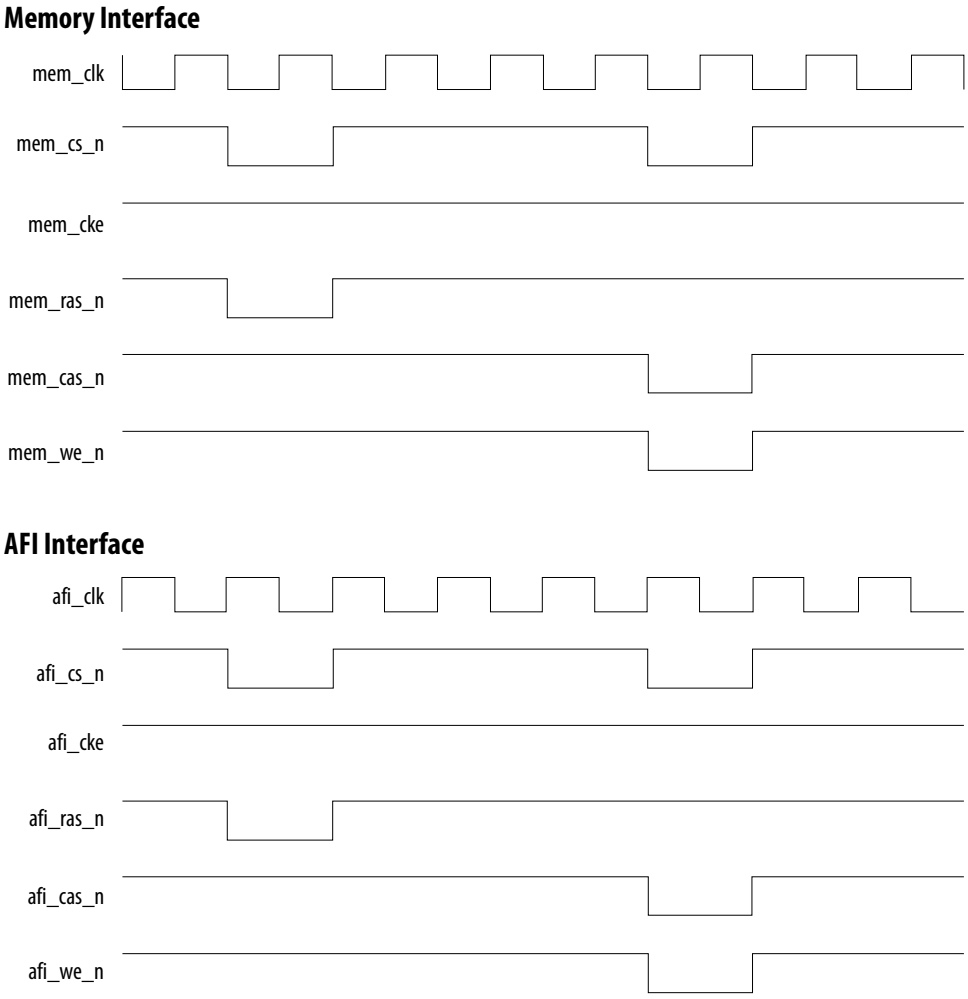


Figure 25. AFI Address and Command Half-Rate

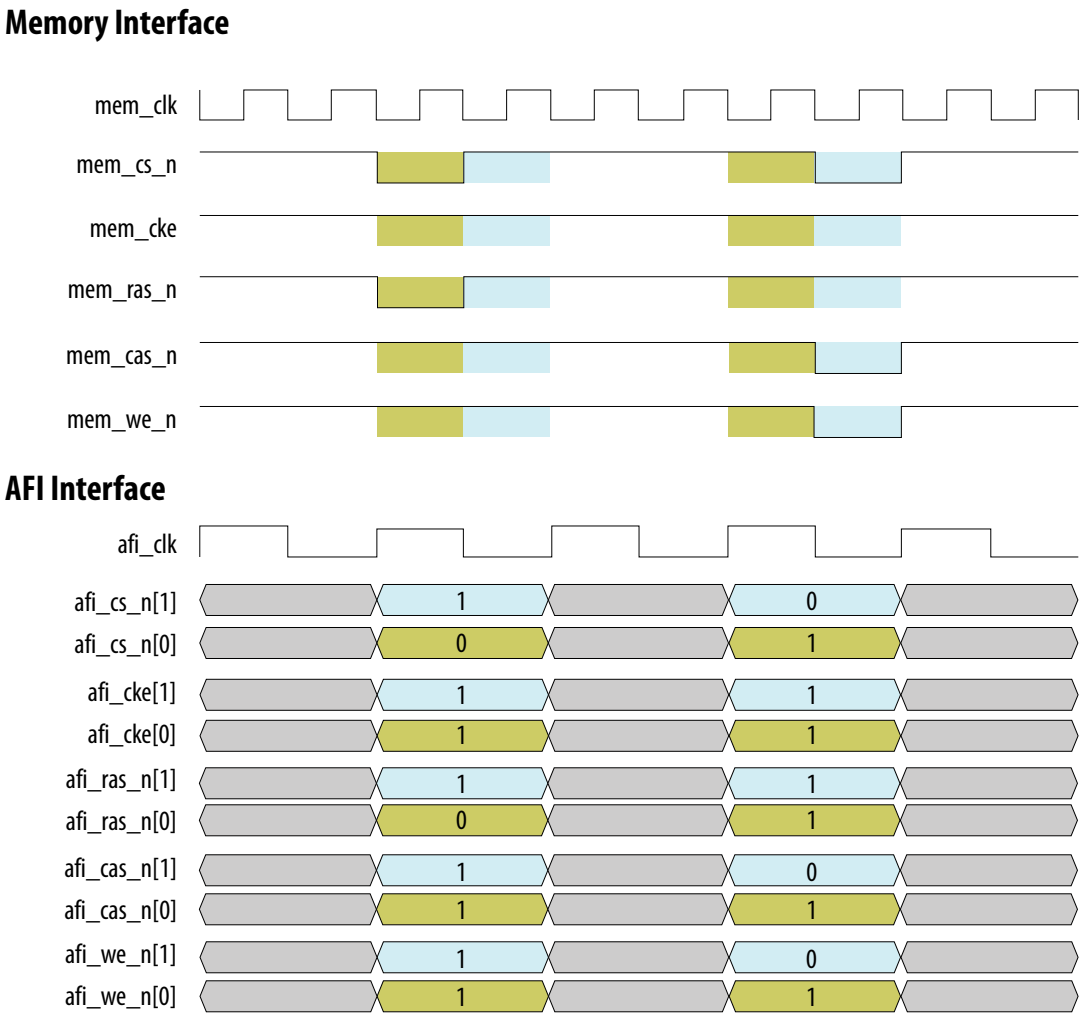
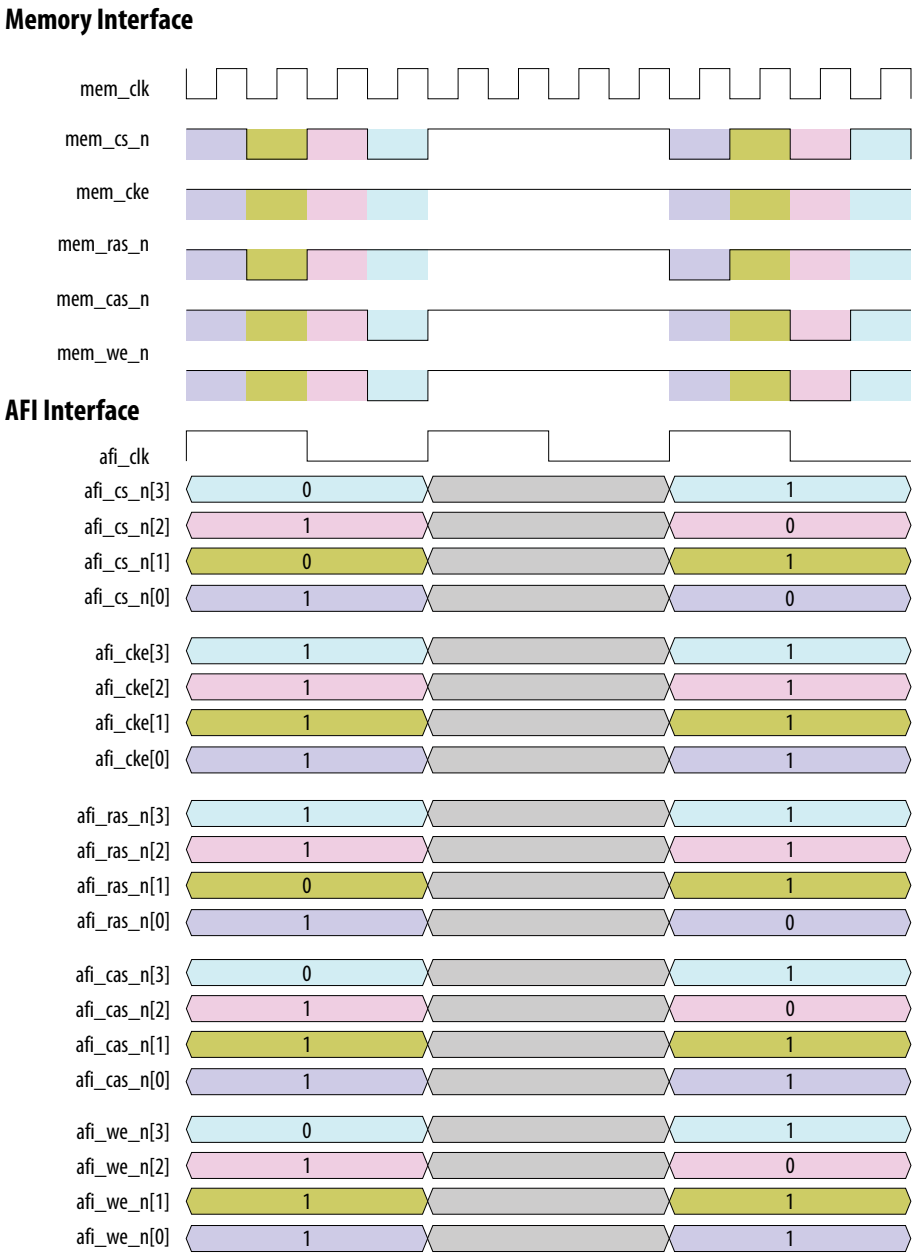


Figure 26. AFI Address and Command Quarter-Rate



4.3.2. AFI Write Sequence Timing Diagrams

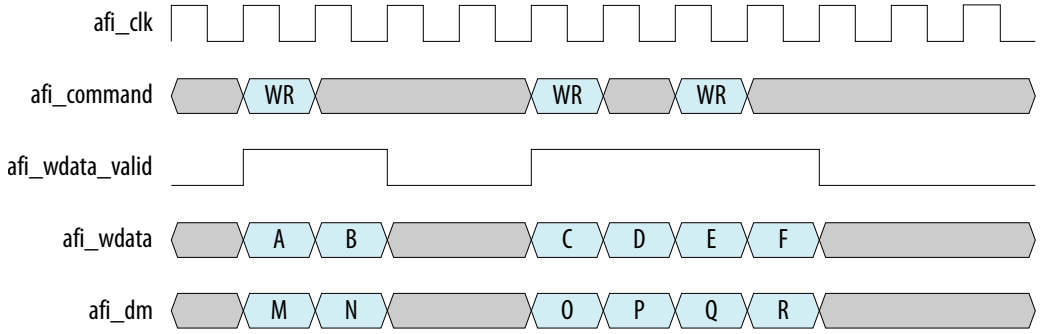
The following timing diagrams illustrate the relationships between the write command and corresponding write data and write enable signals, in full, half, and quarter rate.

For half rate and quarter rate, when the `write` command is sent on the first memory clock in a PHY clock (for example, `afi_cs_n[0] = 0`), that access is called *aligned access*; otherwise it is called *unaligned access*. You may use either aligned or unaligned access, or you may use both, but you must ensure that the distance

between the `write` command and the corresponding write data are constant on the AFI interface. For example, if a command is sent on the second memory clock in a PHY clock, the write data must also start at the second memory clock in a PHY clock.

Write sequences with wlat=0

Figure 27. AFI Write Data Full-Rate, wlat=0



The following diagrams illustrate both aligned and unaligned access. The first three write commands are aligned accesses where they were issued on LSB of `afi_command`. The fourth write command is unaligned access where it was issued on a different command slot. AFI signals must be shifted accordingly, based on the command slot.

Figure 28. AFI Write Data Half-Rate, wlat=0

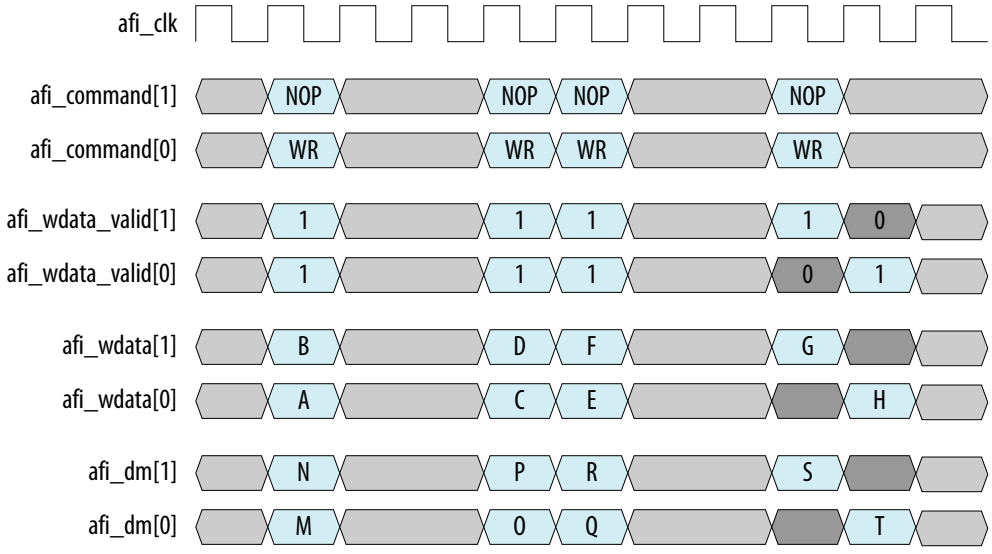
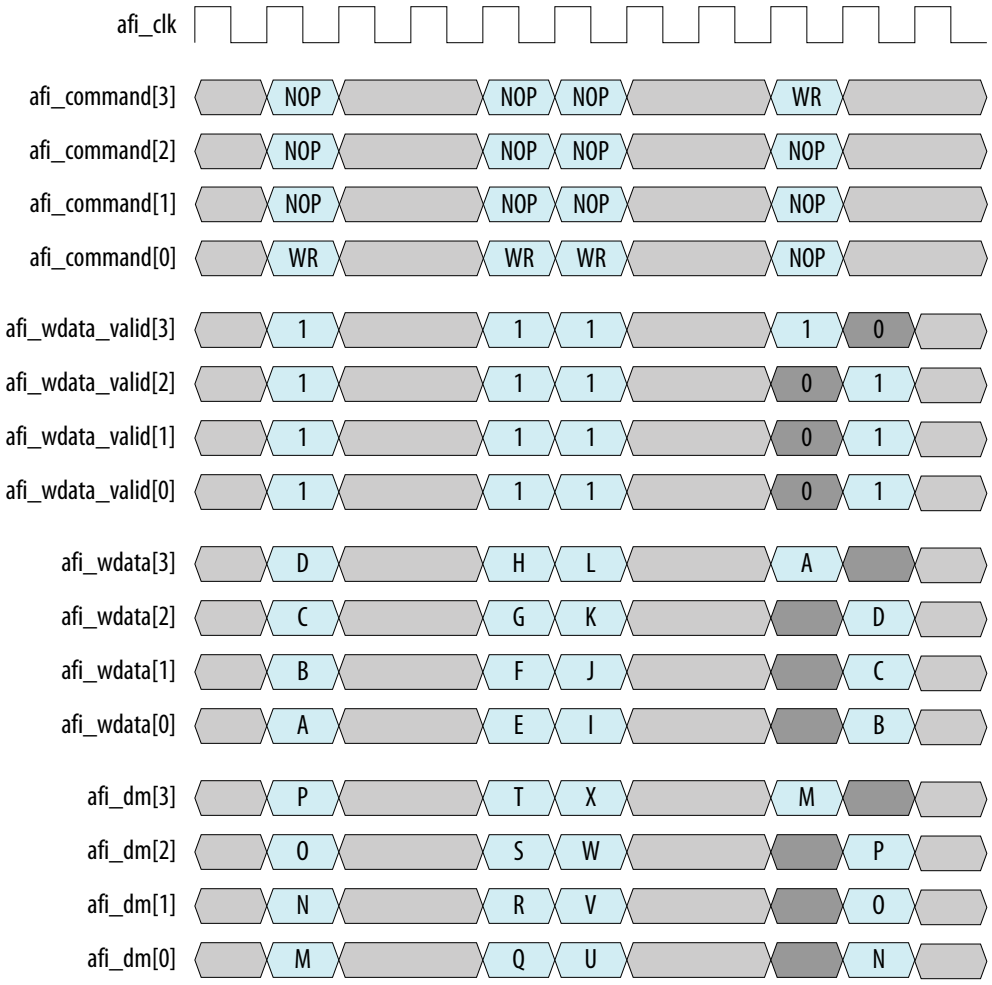


Figure 29. AFI Write Data Quarter-Rate, wlat=0



Write sequences with wlat=non-zero

The `afi_wlat` is a signal from the PHY. The controller must delay `afi_dqs_burst`, `afi_wdata_valid`, `afi_wdata` and `afi_dm` signals by a number of PHY clock cycles equal to `afi_wlat`, which is a static value determined by calibration before the PHY asserts `cal_success` to the controller. The following figures illustrate the cases when wlat=1. Note that wlat is in the number of PHY clocks and therefore wlat=1 equals 1, 2, and 4 memory clocks delay, respectively, on full, half and quarter rate.

Figure 30. AFI Write Data Full-Rate, wlat=1

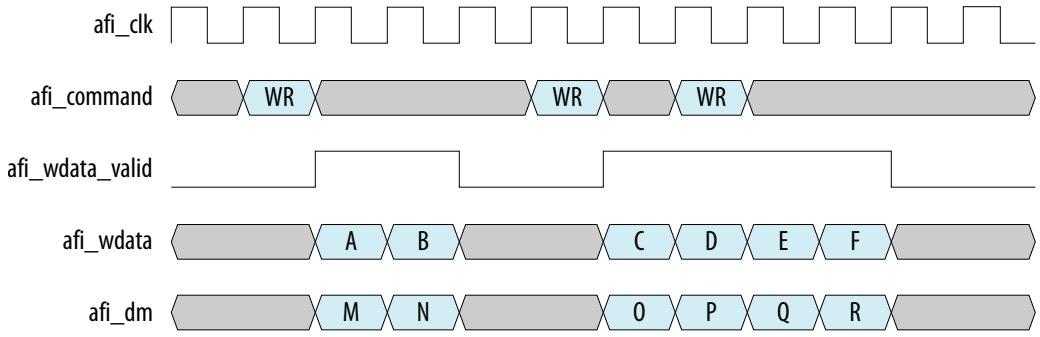


Figure 31. AFI Write Data Half-Rate, wlat=1

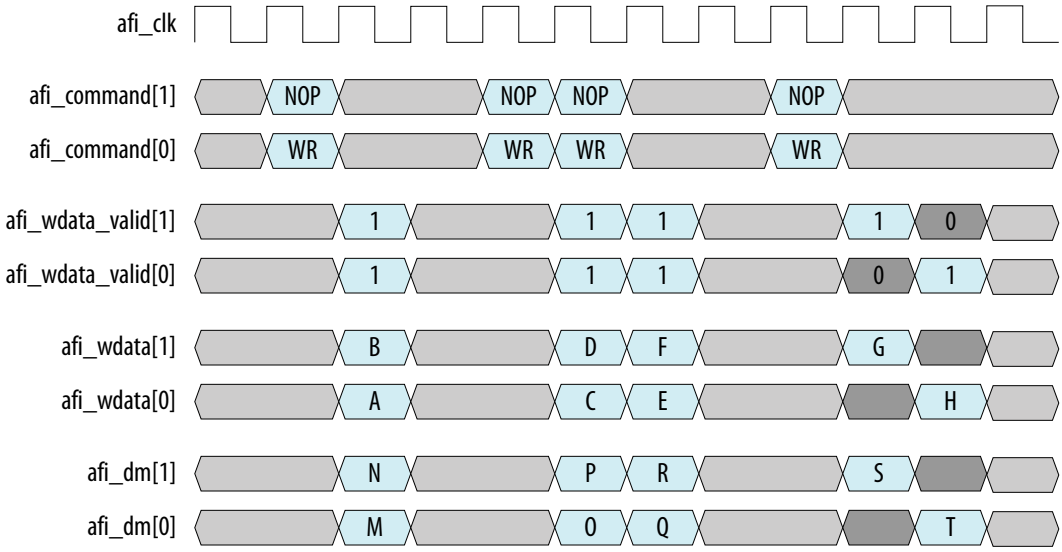
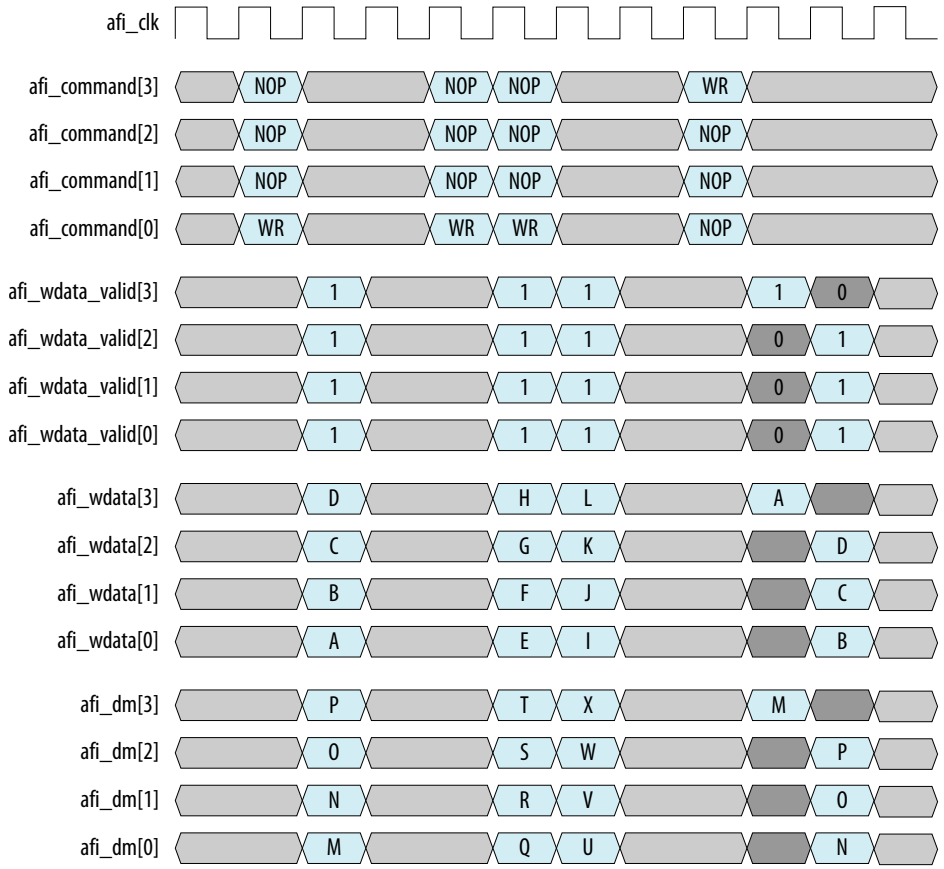


Figure 32. AFI Write Data Quarter-Rate, wlat=1



DQS burst

The `afi_dqs_burst` signal must be asserted one or two complete memory clock cycles earlier to generate DQS preamble. DQS preamble is equal to one-half and one-quarter AFI clock cycles in half and quarter rate, respectively.

A DQS preamble of two is required in DDR4, when the write preamble is set to two clock cycles.

The following diagrams illustrate how afi_dqs_burst must be asserted in full, half, and quarter-rate configurations.

Figure 33. AFI DQS Burst Full-Rate, wlat=1

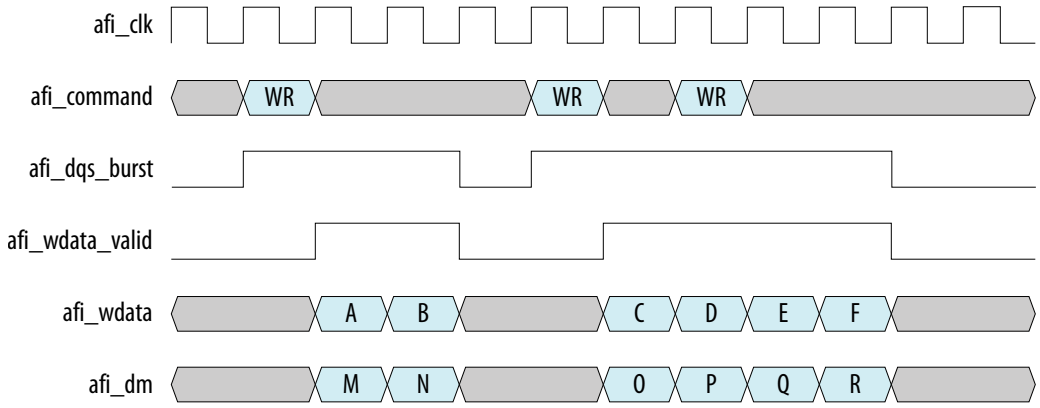


Figure 34. AFI DQS Burst Half-Rate, wlat=1

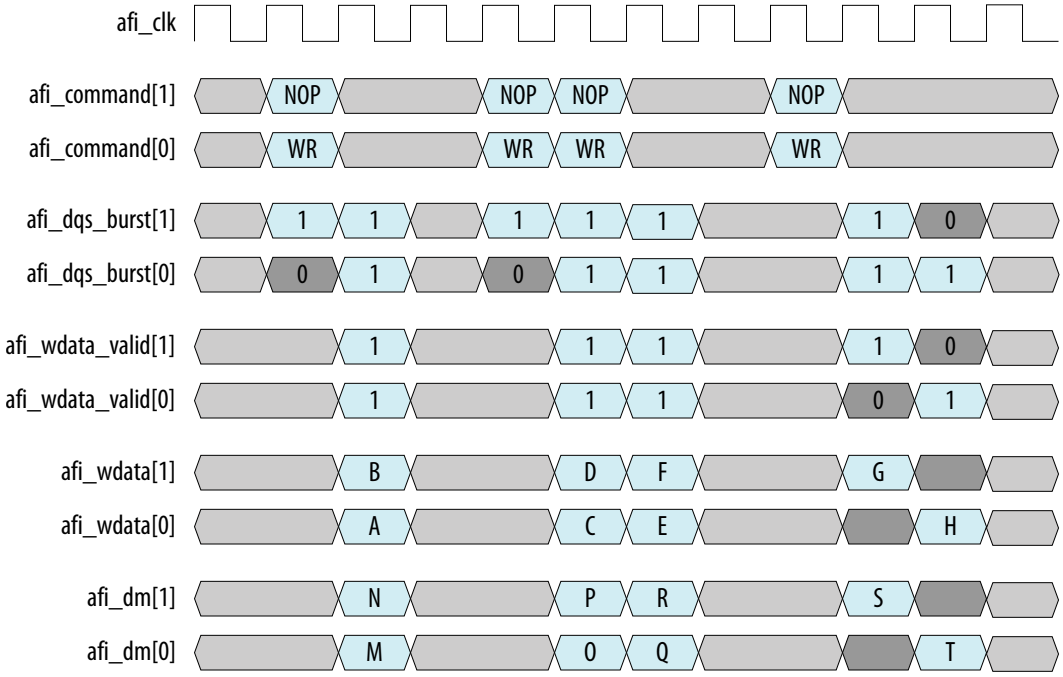
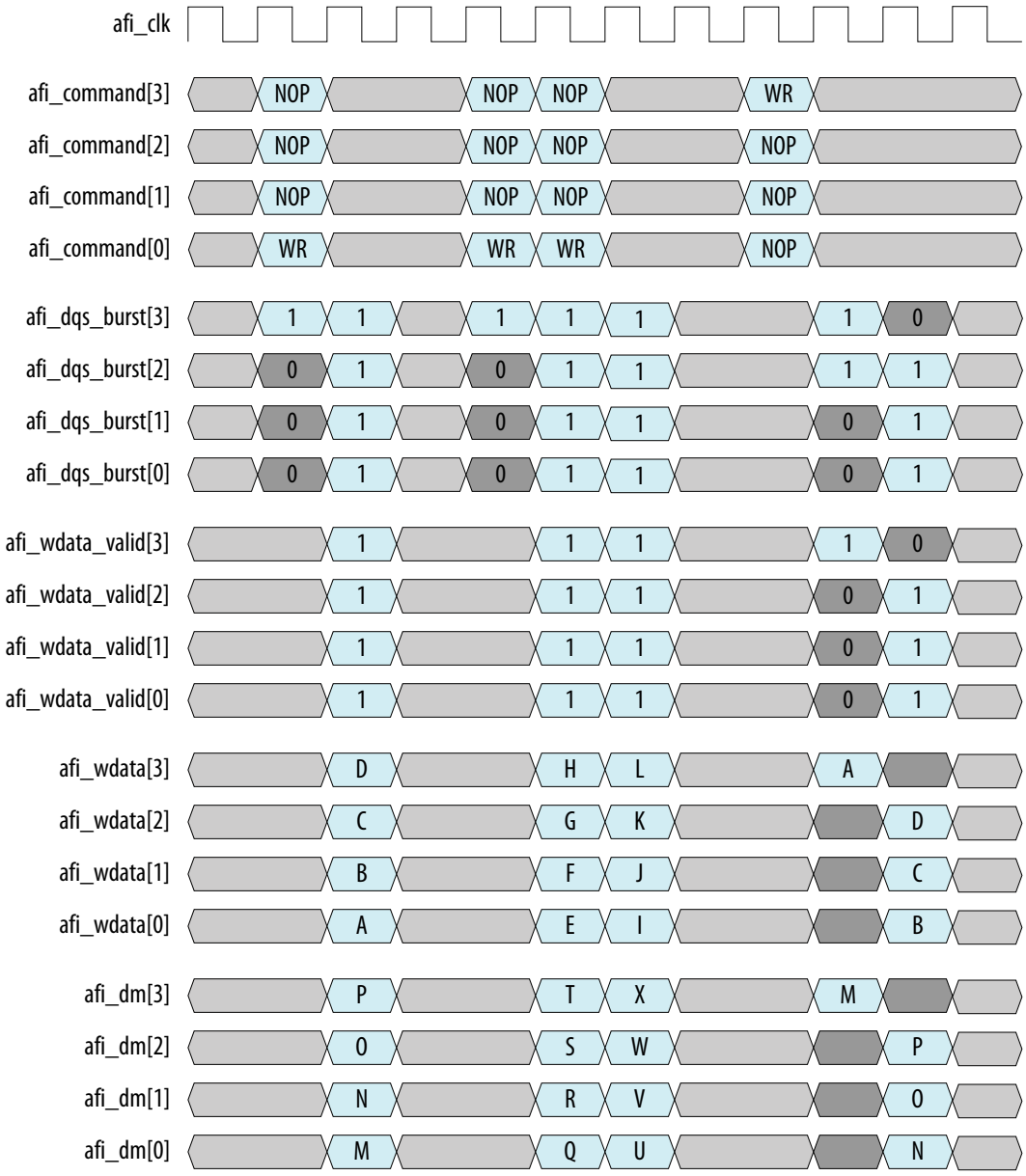


Figure 35. AFI DQS Burst Quarter-Rate, wlat=1



Write data sequence with DBI (DDR4 and QDRIV only)

The DDR4 write DBI feature is supported in the PHY, and when it is enabled, the PHY sends and receives the DBI signal without any controller involvement. The sequence is identical to non-DBI scenarios on the AFI interface.

Write data sequence with CRC (DDR4 only)

When the CRC feature of the PHY is enabled and used, the controller ensures at least one memory clock cycle between `write` commands, during which the PHY inserts the CRC data. Sending back to back `write` command would cause functional failure. The following figures show the legal sequences in CRC mode.

Entries marked as *0* and *RESERVE* must be observed by the controller; no information is allowed on those entries.

Figure 36. AFI Write Data with CRC Half-Rate, wlat=2

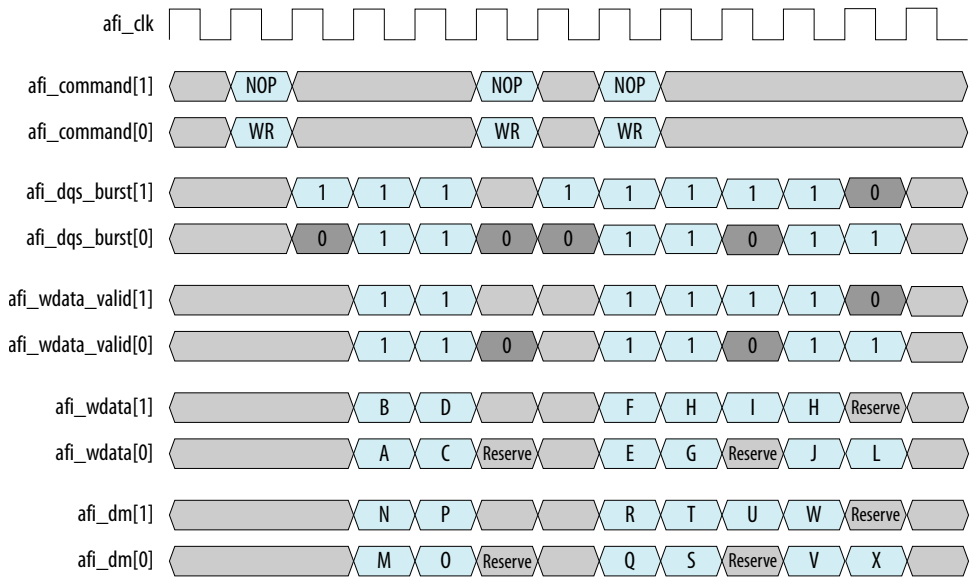
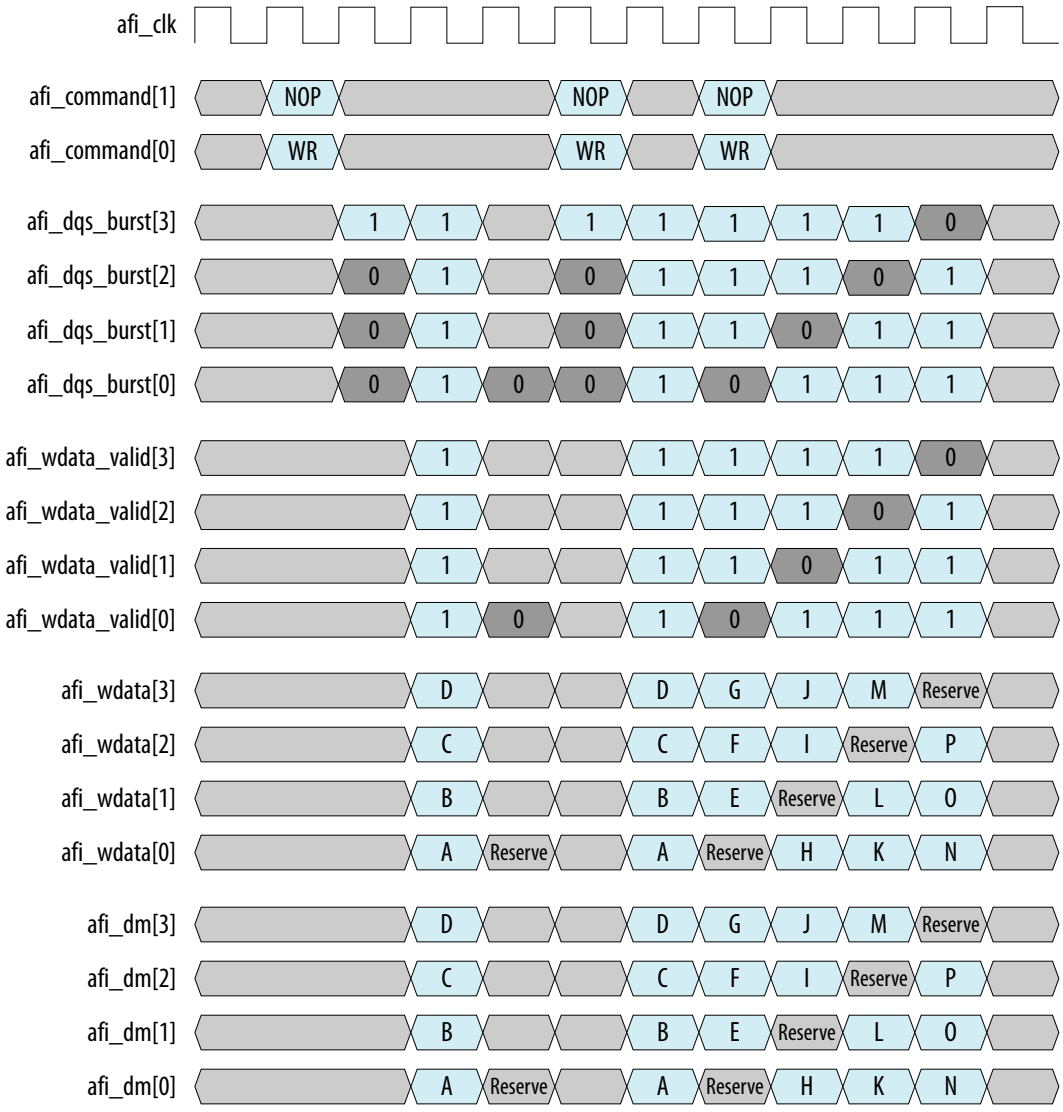


Figure 37. AFI Write Data with CRC Quarter-Rate, wlat=2



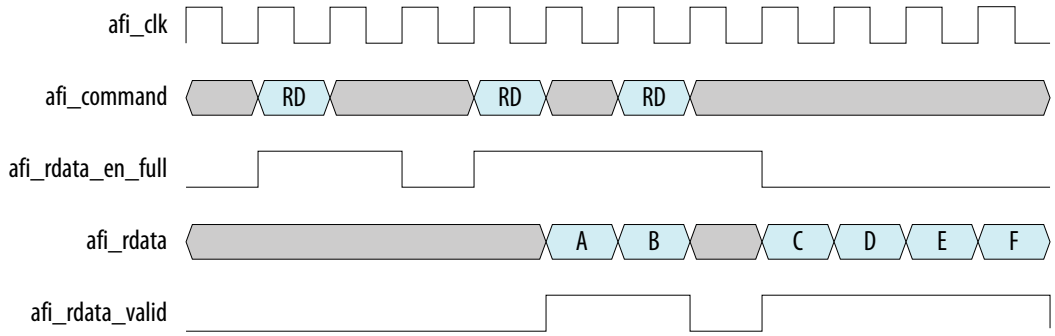
4.3.3. AFI Read Sequence Timing Diagrams

The following waveforms illustrate the AFI read data waveform in full, half, and quarter-rate, respectively.

The `afi_rdata_en_full` signal must be asserted for the entire read burst operation. The `afi_rdata_en` signal need only be asserted for the intended read data.

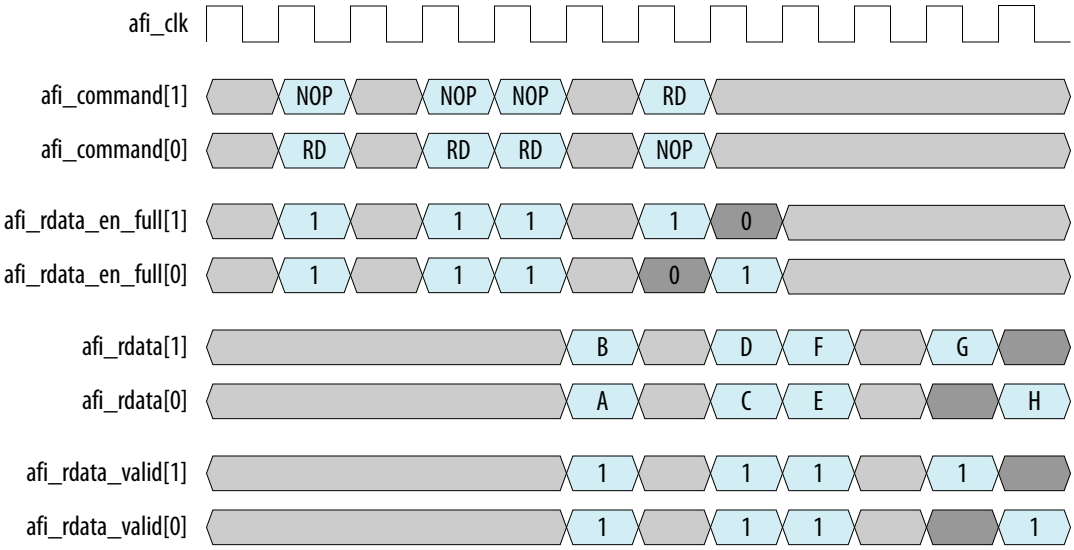
Aligned and unaligned access for read commands is similar to write commands; however, the `afi_rdata_en_full` signal must be sent on the same memory clock in a PHY clock as the read command. That is, if a read command is sent on the second memory clock in a PHY clock, `afi_rdata_en_full` must also be asserted, starting from the second memory clock in a PHY clock.

Figure 38. AFI Read Data Full-Rate



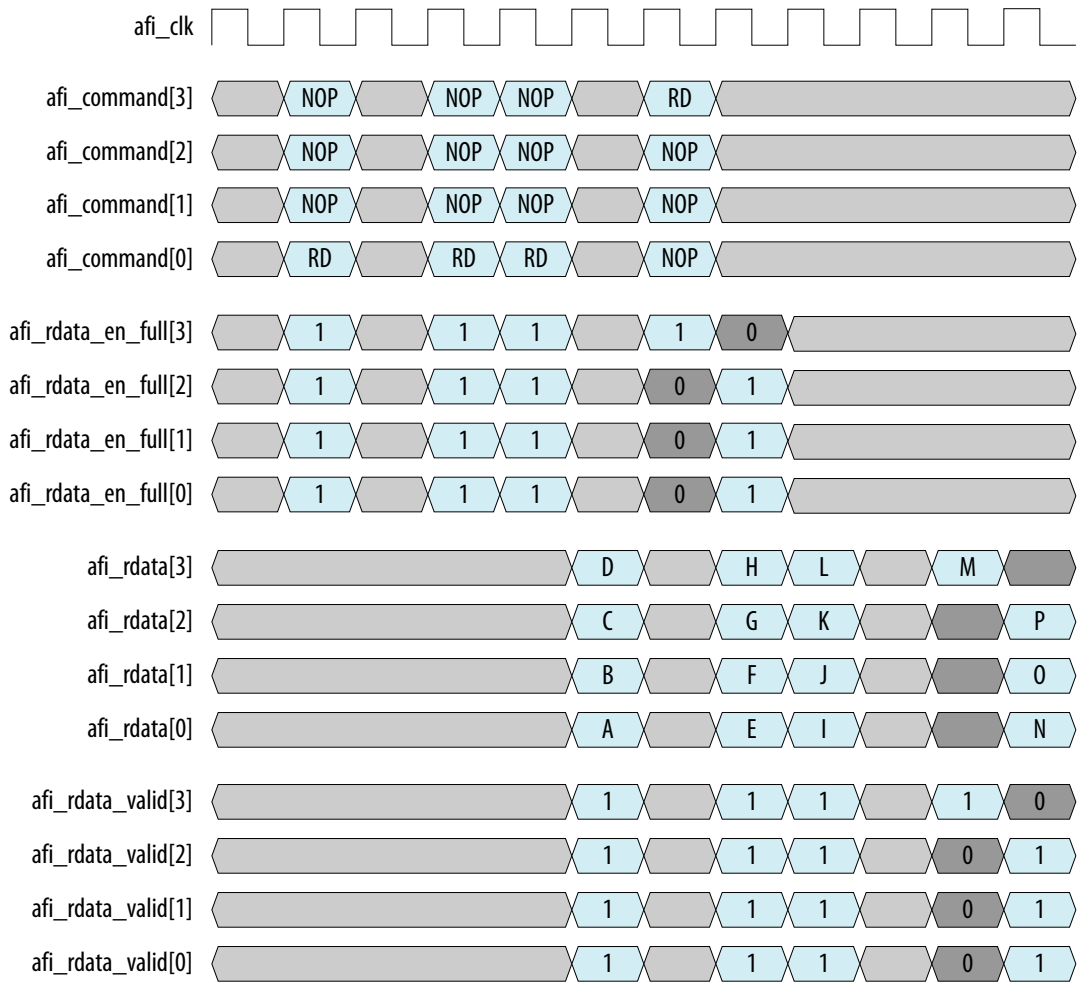
The following figure illustrates that the second and third reads require only the first and second half of data, respectively. The first three `read` commands are aligned accesses where they are issued on the LSB of `afi_command`. The fourth `read` command is unaligned access, where it is issued on a different command slot. AFI signals must be shifted accordingly, based on command slot.

Figure 39. AFI Read Data Half-Rate



In the following figure, the first three `read` commands are aligned accesses where they are issued on the LSB of `afi_command`. The fourth `read` command is unaligned access, where it is issued on a different command slot. AFI signals must be shifted accordingly, based on command slot.

Figure 40. AFI Read Data Quarter-Rate



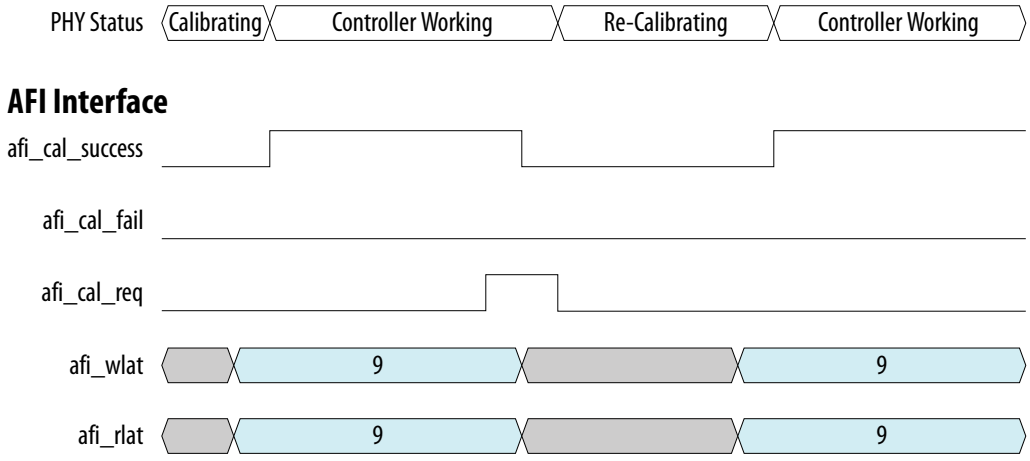
4.3.4. AFI Calibration Status Timing Diagram

The controller interacts with the PHY during calibration at power-up and at recalibration.

At power-up, the PHY holds `afi_cal_success` and `afi_cal_fail` 0 until calibration is done, when it asserts `afi_cal_success`, indicating to controller that the PHY is ready to use and `afi_wlat` and `afi_rlat` signals have valid values.

At recalibration, the controller asserts `afi_cal_req`, which triggers the same sequence as at power-up, and forces recalibration of the PHY.

Figure 41. Calibration



4.4. Intel Stratix 10 Memory Mapped Register (MMR) Tables

The address buses to read and write from the MMR registers are 10 bits wide, while the read and write data buses are configured to be 32 bits. The Bits Register Link column in the table below provides the mapping on the width of the data read within the 32-bit bus. The reads and writes are always performed using the 32-bit-wide bus.

Register Summary

Register	Address 32-bit Bus	Bits Register Link
ctrlcfg0	10	32
ctrlcfg1	11	32
dramtiming0	20	32
caltiming0	31	32
caltiming1	32	32
caltiming2	33	32
caltiming3	34	32
caltiming4	35	32
caltiming9	40	32
dramaddrw	42	32
sideband0	43	32
sideband1	44	32
sideband4	47	32
sideband6	49	32
sideband7	50	32
sideband9	52	32

continued...

Register	Address 32-bit Bus	Bits Register Link
sideband11	54	32
sideband12	55	32
sideband13	56	32
sideband14	57	32
dramsts	59	32
niosreserve0	68	32
niosreserve1	69	32
sideband16	79	32
ecc3	130	32
ecc4	144	32
ecc5	145	32
ecc6	146	32
ecc7	147	32
ecc8	148	32

Note: Addresses are in decimal format.

4.4.1. ctrlcfg0

address=10(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_mem_type	3	0	Indicates memory type. "0000" for DDR3 SDRAM, and "0001" for DDR4 SDRAM.	Read
cfg_dimm_type	6	4	Indicates dimm type.	Read
cfg_ac_pos	8	7	Indicates Command Address pin position.	Read
Reserved	31	9	Reserved.	Read

4.4.2. ctrlcfg1

address=11(32 bit)

Field	Bit High	Bit Low	Description	Access
Reserved	4	0	Reserved.	Read
cfg_addr_order	6	5	Indicates the order for address interleaving. This is related to mappings between Avalon-MM address and the SDRAM address. "00"	Read

continued...

Field	Bit High	Bit Low	Description	Access
			- chip, row, bank(BG, BA), column; "01" - chip, bank(BG, BA), row, column; "10"-row, chip, bank(BG, BA), column.	
cfg_ctrl_enable_ecc	7	7	Enable the generation and checking of ECC.	Read
cfg_dbc0_enable_ecc	8	8	Enable the generation and checking of ECC.	Read
cfg_dbc1_enable_ecc	9	9	Enable the generation and checking of ECC.	Read
cfg_dbc2_enable_ecc	10	10	Enable the generation and checking of ECC.	Read
cfg_dbc3_enable_ecc	11	11	Enable the generation and checking of ECC.	Read
cfg_reorder_data	12	12	This bit controls whether the controller can reorder operations to optimize SDRAM bandwidth. It should generally be set to a one.	Read
cfg_ctrl_reorder_rdata	13	13	This bit controls whether the controller needs to reorder the read return data.	Read
cfg_dbc0_reorder_rdata	14	14	This bit controls whether the controller needs to reorder the read return data.	Read
cfg_dbc1_reorder_rdata	15	15	This bit controls whether the controller needs to reorder the read return data.	Read
cfg_dbc2_reorder_rdata	16	16	This bit controls whether the controller needs to reorder the read return data.	Read
cfg_dbc3_reorder_rdata	17	17	This bit controls whether the controller needs to reorder the read return data.	Read
cfg_reorder_read	18	18	This bit controls whether the controller can reorder read command.	Read
cfg_starve_limit	24	19	Specifies the number of DRAM burst transactions that an individual transaction allows to reorder ahead of it before its priority is raised in the memory controller.	Read
Reserved	25	25	Reserved.	Read
cfg_ctrl_enable_dm	26	26	Set to 1 to enable DRAM operation if DM pins are connected.	Read
cfg_dbc0_enable_dm	27	27	Set to 1 to enable DRAM operation if DM pins are connected.	Read
cfg_dbc1_enable_dm	28	28	Set to 1 to enable DRAM operation if DM pins are connected.	Read
cfg_dbc2_enable_dm	29	29	Set to 1 to enable DRAM operation if DM pins are connected.	Read
cfg_dbc3_enable_dm	30	30	Set to 1 to enable DRAM operation if DM pins are connected.	Read

4.4.3. dramtiming0

address=20(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_tcl	6	0	Memory read latency.	Read
Reserved	31	7	Reserved.	Read

4.4.4. caltiming0

address=31(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_act_to_rdwr	5	0	Activate to Read/Write command timing.	Read
cfg_t_param_act_to_pch	11	6	Active to precharge.	Read
cfg_t_param_act_to_act	17	12	Active to activate timing on same bank.	Read
cfg_t_param_act_to_act_diff_bank	23	18	Active to activate timing on different banks, for DDR4 same bank group.	Read
cfg_t_param_act_to_act_diff_bg	29	24	Active to activate timing on different bank groups, DDR4 only.	Read

4.4.5. caltiming1

address=32(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_rd_to_rd	5	0	Read to read command timing on same bank.	Read
cfg_t_param_rd_to_rd_diff_chip	11	6	Read to read command timing on different chips.	Read
cfg_t_param_rd_to_rd_diff_bg	17	12	Read to read command timing on different chips.	Read
cfg_t_param_rd_to_wr	23	18	Write to read command timing on same bank.	Read
cfg_t_param_rd_to_wr_diff_chip	29	24	Read to write command timing on different chips	Read

4.4.6. caltiming2

address=33(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_rd_to_wr_diff_bg	5	0	Read to write command timing on different bank groups.	Read
cfg_t_param_rd_to_pch	11	6	Read to precharge command timing.	Read
cfg_t_param_rd_ap_to_valid	17	12	Read command with autoprecharge to data valid timing.	Read
cfg_t_param_wr_to_wr	23	18	Write to write command timing on same bank.	Read
cfg_t_param_wr_to_wr_diff_chip	29	24	Write to write command timing on different chips.	Read

4.4.7. caltiming3

address=34(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_wr_to_wr_diff_bg	5	0	Write to write command timing on different bank groups.	Read
cfg_t_param_wr_to_rd	11	6	Write to read command timing.	Read
cfg_t_param_wr_to_rd_diff_chip	17	12	Write to read command timing on different chips.	Read
cfg_t_param_wr_to_rd_diff_bg	23	18	Write to read command timing on different bank groups.	Read
cfg_t_param_wr_to_pch	29	24	Write to precharge command timing.	Read

4.4.8. caltiming4

address=35(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_wr_ap_to_valid	5	0	Write with autoprecharge to valid command timing.	Read
cfg_t_param_pch_to_valid	11	6	Precharge to valid command timing.	Read
cfg_t_param_pch_all_to_valid	17	12	Precharge all to banks being ready for bank activation command.	Read
cfg_t_param_arf_to_valid	25	18	Auto Refresh to valid DRAM command window.	Read
cfg_t_param_pdn_to_valid	31	26	Power down to valid bank command window.	Read

4.4.9. caltiming9

address=40(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_t_param_4_act_to_act	7	0	The four-activate window timing parameter.	Read

4.4.10. dramaddrw

address=42(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_col_addr_width	4	0	The number of column address bits for the memory devices in your memory interface.	Read
cfg_row_addr_width	9	5	The number of row address bits for the memory devices in your memory interface.	Read
cfg_bank_addr_width	13	10	The number of bank address bits for the memory devices in your memory interface.	Read
cfg_bank_group_addr_width	15	14	The number of bank group address bits for the memory devices in your memory interface.	Read
cfg_cs_addr_width	18	16	The number of chip select address bits for the memory devices in your memory interface.	Read

4.4.11. sideband0

address=43(32 bit)

Field	Bit High	Bit Low	Description	Access
mr_cmd_trigger	0	0	Mode Register Command Request. When asserted, indicates user request to execute mode register command. Controller clears bit to 0 when operation is completed. Register offset 37h and 38h must be properly configured before requesting Mode Register Command. Read offset 31h for Mode Register Command Status.	Read/Write

4.4.12. sideband1

address=44(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_refresh_req	3	0	Rank Refresh Request. When asserted, indicates a refresh request to the specific rank. Controller clears this bit to 0 when the refresh is executed.	Read/Write

4.4.13. sideband4

address=47(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_self_rfsh_req	3	0	Self-refresh request. When asserted, indicates a self-refresh request to DRAM. All 4 bits must be asserted or de-asserted at the same time. User clear to exit self refresh.	Read/Write

4.4.14. sideband6

address=49(32 bit)

Field	Bit High	Bit Low	Description	Access
mr_cmd_ack	0	0	Register Command In Progress. When asserted, indicates Mode Register Command in progress.	Read

4.4.15. sideband7

address=50(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_refresh_ack	0	0	Refresh In Progress. Acknowledgement signal for refresh request. Indicates that refresh is in progress. Asserts when refresh request is sent out to PHY until `tRFC/t_param_arf_to_valid` is fulfilled.	Read

4.4.16. sideband9

address=52(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_self_rfsh_ack	0	0	Self-refresh In Progress. Acknowledgement signal for the self-refresh request. A value of 1 indicates that memory is in self refresh mode.	Read

4.4.17. sideband11

address=54(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_auto_pd_ack	0	0	Auto Power Down In Progress. Acknowledgement signal for auto power down. A value of 1 indicates that the memory is in auto power down mode.	Read

4.4.18. sideband12

address=55(32 bit)

Field	Bit High	Bit Low	Description	Access
mr_cmd_type	2	0	Register command type. Indicates the type of register command.	Read/Write
			000 - Mode Register Set (DDR3 and DDR4)	
			Others - Reserved	
mr_cmd_rank	6	3	Register command rank. Indicates the rank targeted by the register command.	Read/Write
			0001 - Chip select 0	
			0010 - Chip select 1	
			0011 - Chip select 0 and chip select 1	
			1111 - all chip selects	
			Mode Register Set - Any combination of chip selects.	

4.4.19. sideband13

address=56(32 bit)

Field	Bit High	Bit Low	Description	Access
mr_cmd_opcode	31	0	Register Command Opcode. Information used for register command.	Read/Write
			DDR4	
			[26:24] C2:C0	
			[23] ACT	
			[22:21] BG1:BG0	
			[20] Reserved	
			[19:18] BA1:BA0	
			[17] A17	
			[16] RAS#	
			[15] CAS#	
			[14] WE#	
			[13:0] A13:A0	
			MRS: [22:21] is BG1:BG0, [19:18] is BA1:BA0, [13:0] is Opcode[13:0]	
			DDR3	
			[26:21] Reserved	
			[20:18] BA2:BA0	
			[17] A14	
			[16] RAS#	
			[15] CAS#	
			[14] WE#	
			[13:0] A13:A0	
			MRS: [19:18] is BA1:BA0, [13:0] is Opcode[13:0]	

4.4.20. sideband14

address=57(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_refresh_cid	3	1	DDR4 3DS Chip ID Refresh. When asserted, indicates the logical rank chip ID for 3DS refresh. (This field is not applicable for DDR3.)	Read

4.4.21. dramsts

address=59(32 bit)

Field	Bit High	Bit Low	Description	Access
phy_cal_success	0	0	This bit is set to 1 if the PHY calibrates successfully.	Read
phy_cal_fail	1	1	This bit is set to 1 if the PHY does not calibrate successfully.	Read

4.4.22. niosreserve0

address=68(32 bit)

Field	Bit High	Bit Low	Description	Access
nios_reserve0	15	0	Indicates interface width.	Read

4.4.23. niosreserve1

address=69(32 bit)

Field	Bit High	Bit Low	Description	Access
nios_reserve1	15	0	Indicates ACDS version.	Read

4.4.24. sideband16

address=79(32 bit)

Field	Bit High	Bit Low	Description	Access
mmr_3ds_refresh_ack	31	0	DDR4 3DS Refresh Acknowledge. When asserted, indicates acknowledgement for the DDR4 3DS refresh.	Read
			[7:0] Refresh acknowledgement for logical rank [7:0] for physical rank 0.	
			[15:8] Refresh acknowledgement for logical rank [7:0] for physical rank 1.	
			[23:16] Refresh acknowledgement for logical rank [7:0] for physical rank 2.	
			[31:24] Refresh acknowledgement for logical rank [7:0] for physical rank 3.	

4.4.25. ecc3: ECC Error and Interrupt Configuration

address=130(32 bit)

Field	Bit High	Bit Low	Description	Access
cfg_gen_sbe	0	0	A value of 1 enables the generate SBE feature. Generates a single bit error during the write process.	Read/Write
cfg_gen_dbe	1	1	A value of 1 enables the generate DBE feature. Generates a double bit error during the write process.	Read/Write
cfg_enable_intr	2	2	A value of 1 enables the interrupt feature. The interrupt signal notifies if an error condition occurs. The condition is configurable.	Read/Write
cfg_mask_sbe_intr	3	3	A value of 1 masks the interrupt signal when SBE occurs.	Read/Write
cfg_mask_dbe_intr	4	4	A value of 1 masks the interrupt signal when DBE occurs.	Read/Write
cfg_mask_corr_dropped_intr	5	5	A value of 1 masks the interrupt signal when the auto correction command can't be scheduled, due to back-pressure (FIFO full).	Read/Write
cfg_mask_hmi_intr	6	6	A value of 1 masks the interrupt signal when the hard memory interface asserts an interrupt signal via the `hmi_interrupt` port.	Read/Write
cfg_clr_intr	7	7	Writing a vale of 1 to this self-clearing bit clears the interrupt signal, error status, and address.	Read/Write
Reserved	31	8		Read

4.4.26. ecc4: Status and Error Information

address=144(32 bit)

Field	Bit High	Bit Low	Description	Access
sts_ecc_intr	0	0	Indicates the interrupt status; a value of 1 indicates an interrupt occurred.	Read
sts_sbe_error	1	1	Indicates the SBE status; a value of 1 indicates SBE occurred.	Read
sts_dbe_error	2	2	Indicates the DBE status; a value of 1 indicates DBE occurred.	Read
sts_corr_dropped	3	3	Indicates the status of correction command dropped; a value of 1 indicates correction command dropped.	Read
sts_sbe_count	7	4	Indicates the number of times SBE error has occurred. The counter can overflow.	Read

continued...

Field	Bit High	Bit Low	Description	Access
sts_dbe_count	11	8	Indicates the number of times DBE error has occurred. The counter can overflow.	Read
sts_corr_dropped_count	15	12	Indicates the number of times correction command has dropped. The counter can overflow.	Read
Reserved	31	16		Read

4.4.27. ecc5: Address of Most Recent SBE/DBE

address=145(32 bit)

Field	Bit High	Bit Low	Description	Access
sts_err_addr*	31	0	Address of the most recent single-bit error or double-bit error.	Read

4.4.28. ecc6: Address of Most Recent Correction Command Dropped

address=146(32 bit)

Field	Bit High	Bit Low	Description	Access
sts_corr_dropped_addr	31	0	Address of the most recent correction command dropped.	Read

About ECC Errors in DDR3 and DDR4 Interfaces

ECC errors are categorized as either single-bit errors (which are correctable by ECC code), or double-bit errors (which are not correctable). You can determine whether an ECC error has occurred, by checking the values of the ecc4 register fields `sts_ecc_intr`, `sts_sbe_error`, and `sts_dbe_error`.

- If a double-bit error has occurred, it indicates that the memory is corrupted and cannot be corrected by ECC code. You can choose to restart your system.
- If a single-bit error has occurred, the controller attempts to correct the error by performing a write-back to memory using the fixed data plus an ECC code. The write-back is enabled when you have selected **Enable Auto Error Correction to External Memory** on the *Controller* tab in the IP parameter. The write-back requires space in the command queue and in the data FIFO buffer; because Intel Stratix 10 FPGAs have only 8 command queues, it is possible that the controller may not be able to schedule the write-back, in which case the write-back may be dropped. You can determine whether a write-back has been dropped, by reading the status of the ecc4 register fields `ctrl_ecc_sts_corr_dropped_count`, `ctrl_ecc_sts_corr_dropped_addr`, `ctrl_ecc_sts_corr_dropped`, and `ctrl_ecc_sts_intr` registers.

  If you discover that a write-back has been dropped, you can do either of two things:

  — You can ignore the dropped write-back, because it is a single-bit error that the controller may be able to detect and correct on the next memory read without any intervention – provided the condition does not further deteriorate into a double-bit error.

  — You can read the address from the ecc6 register field `ctrl_ecc_sts_corr_dropped_addr`, and perform a memory write with `byte_enable=0`, thereby causing the controller to access the memory location again and schedule a new write-back.

4.4.29. ecc7: Extension for Address of Most Recent SBE/DBE

address=147(32 bit)

Field	Bit High	Bit Low	Description	Access
sts_err_addr_ext	2	0	Extension for address of the most recent single-bit error or double-bit error.	Read

4.4.30. ecc8: Extension for Address of Most Recent Correction Command Dropped

address=148(32 bit)

Field	Bit High	Bit Low	Description	Access
sts_corr_dropped_addr_ext	2	0	Extension for address of the most recent correction command dropped.	Read

5. Intel Stratix 10 EMIF – Simulating Memory IP

To simulate your design you require the following components:

- A simulator—The simulator must be an Intel-supported VHDL or Verilog HDL simulator:
  - Aldec Riviera-Pro
  - Cadence Xcelium
  - Siemens EDA* ModelSim
  - Siemens EDA Questa*
  - Synopsys* VCS/VCS-MX
- A design using Intel's External Memory Interface (EMIF) IP
- An example driver or traffic generator (to initiate read and write transactions)
- A testbench and a suitable memory simulation model

The Intel External Memory Interface IP is not compatible with the Platform Designer Testbench System. Instead, use the simulation design example from your generated IP to validate memory interface operation, or as a reference for creating a full simulatable design. The provided simulation design example contains the generated memory interface, a memory model, and a traffic generator. For more information about the EMIF simulation design example, refer to the *Intel Stratix 10 EMIF IP Design Example User Guide*.

Memory Simulation Models

There are two types of memory simulation models that you can use:

- Intel-provided generic memory model
- Vendor-specific memory model

The Intel Quartus Prime software generates the generic memory simulation model with the simulation design example. The model adheres to all the memory protocol specifications, and can be parameterized.

Vendor-specific memory models are simulation models for specific memory components from memory vendors such as Micron and Samsung. You can obtain these simulation models from the memory vendor's website.

Note: Intel does not provide support for vendor-specific memory models.

Related Information

Modifying the Example Driver to Replicate the Failure on page 382

5.1. Simulation Options

The following simulation options are available with the example testbench to improve simulation speed:

- Full calibration—Calibrates the same way as in hardware, and includes all phase sweeps, delay adjustments, and data centering.
- Skip calibration—Loads memory configuration settings and enters user mode, providing the fastest simulation time.

Note: For proper simulation of DQS Tracking, you must enable full calibration.

Both simulation options represent accurate controller efficiency and do not take into account board skew. This may cause a discrepancy in the simulated interface latency numbers. For more information regarding simulation assumptions and differences between RTL simulation and post-fit implementation, refer to the *Simulation Versus Hardware Implementation* chapter in the *Intel Stratix 10 EMIF IP Design Example User Guide*.

Table 169. Typical Simulation Times Using Intel Stratix 10 EMIF IP

Calibration Mode/Run Time (1)	Estimated Simulation Time	
	Small Interface (×8 Single Rank)	Large Interface (×72 Quad Rank)
Full • Full calibration • Includes all phase/delay sweeps and centering	20 minutes	~ 1 day
Skip • Skip calibration • Preloads calculated settings	10 minutes	25 minutes
Abstract PHY • Replace PHY and external memory model with a single abstract PHY model. • IMPORTANT: *External memory model is NOT used in this mode. No I/O switching occurs to the external memory model*.	1 minute	5 minutes
Note to Table: 1. Uses one loop of driver test. One loop of driver is approximately 600 read or write requests, with burst length up to 64. 2. Simulation times shown in this table are approximate measurements made using Synopsys VCS. Simulation times can vary considerably, depending on the IP configuration, the simulator used, and the computer or server used.		

Related Information

Simulation Walkthrough on page 135

5.2. HPS EMIF Simulation

Simulation of a design containing Intel Stratix 10 HPS EMIF is not supported.

5.3. Simulation Walkthrough

Simulation is a good way to determine the latency of your system. However, the latency reflected 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 Intel Stratix 10 EMIF IP supports functional simulation only. Functional simulation is supported at the RTL level after generating a post-fit functional simulation netlist. The post-fit netlist for designs that contain Intel Stratix 10 EMIF IP is a hybrid of the gate level (for FPGA core) and RTL level (for the external memory interface IP). You should validate the functional operation of your design using RTL simulation, and the timing of your design using timing analysis.

The Intel Stratix 10 EMIF IP supports functional simulation through the design example using the Traffic Generator (TG1) or the Configurable Traffic Generator 2.0 (TG2). (For information on TG2, refer to Using the Configurable Traffic Generator (TG2). Functional simulation using TG2 is allowed only with *default traffic pattern*, where TG2 runs a default traffic pattern after reset instead of waiting for user configuration for TG2, as in *user mode*. Do not select *Bypass the default traffic mode* when creating a design example for functional simulation using TG2.

To perform functional simulation for an Intel Stratix 10 EMIF IP design example, locate the design example files in the design example directory.

You can use the IP functional simulation model with any supported VHDL or Verilog HDL simulator.

After you have generated the memory IP, you can locate multiple file sets for various supported simulations in the `sim/ed_sim` subdirectory. For more information about the EMIF simulation design example, refer to the *Intel Stratix 10 External Memory Interfaces IP Design Example User Guide*.

Related Information

Simulation Options on page 134

5.3.1. Calibration Modes

Calibration occurs shortly after the memory device is initialized, to compensate for uncertainties in the hardware system, including silicon PVT variation, circuit board trace delays, and skewed arrival times. Such variations are usually not present in an RTL simulation environment, resulting in two simulatable calibration modes: Skip Calibration mode (which is the default), and Full Calibration mode.

Skip Calibration Mode

In Skip Calibration mode, the calibration processor assumes an ideal hardware environment, where PVT variations, board delays, and trace skews are all zero. Instead of running the actual calibration routine, the calibration processor calculates the expected arrival time of read data based on the memory latency values entered during EMIF IP generation, resulting in reduced simulation time. Skip calibration mode

is recommended for use during system development, because it allows you to focus on interacting with the controller and optimizing your memory access patterns, thus facilitating rapid RTL development.

If you enable Skip Calibration Mode, the interface still performs some memory initialization, sending DRAM Mode Register Set (MRS) commands, or commands to program register code words for RDIMM/LRDIMM, before starting normal operation. These initialization commands are necessary to set up the memory model operation and latencies.

Full Calibration Mode

Full Calibration mode simulates every stage of the calibration algorithm immediately after memory device initialization. The calibration algorithm processes each data group sequentially and each pin in each group individually, causing simulation time to increase with the number of data pins in your interface. You can observe how the calibration algorithm compensates for various delays in the system by incorporating your own board delay model based on trace delays from your PCB design tools. Due to the large simulation overhead, Full Calibration simulation mode is not recommended for rapid development of IP cores.

VHDL Support

VHDL support for mixed-language simulators is implemented by generating the top-level wrapper for the core in VHDL, while all submodules are provided as clear text SystemVerilog files.

A set of precompiled device libraries is provided for use with the Questa - Intel FPGA Edition simulator, which is supplied with the Intel Quartus Prime software. Submodules normally provided as cleartext SystemVerilog files are encrypted using IEEE Verilog HDL encryption for Questa - Intel FPGA Edition.

5.3.2. Abstract PHY Simulation

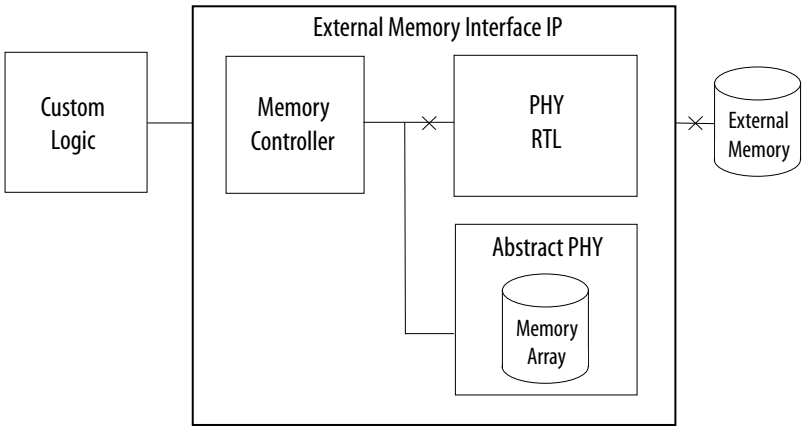
The Abstract PHY is a simulation model of the EMIF PHY that can decrease simulation time by 3-10 times. The Abstract PHY replaces the lane and the external memory model with a single model containing an internal memory array. No switching of the I/Os to the external memory model occurs when simulating with the Abstract PHY.

Abstract PHY reduces simulation time by two mechanisms:

- The Nios processor has been disabled and is replaced by HDL forces that are applied at the beginning of simulation. The HDL forces are a minimum set of registers that configures the memory interface for simulation. The write and read latency values applied by the HDL forces are not representative of the post-calibration values applied to the memory interface running on hardware. However, as long as the customer logic is Avalon and AFI-compliant, these values allow for successful RTL simulation.
- The abstract PHY eliminates the need for full-speed clocks and therefore simulation of the Abstract PHY does not require full-speed clock simulation events.

To use the Abstract PHY, enable **Simulation Options ➤ Abstract PHY for fast simulation** on the **Diagnostic** tab during EMIF IP generation. When you enable Abstract PHY, the EMIF IP is configured as shown below. The PHY RTL and external memory model are disconnected from the data path and in their place is the abstract PHY containing an internal memory array.

Figure 42. Abstract PHY



Note:

- You cannot observe the external memory device signals when you are using Abstract PHY.
- Abstract PHY does not reflect accurate latency numbers.
- Abstract PHY Simulation does not support user-initiated resets using the reset sequence described in User-requested Reset in Intel Stratix 10 EMIF IP.

5.3.3. Simulation Scripts

The Intel Quartus Prime software generates simulation scripts during project generation for four different third party simulation tools—Cadence, Synopsys, Aldec, and Siemens EDA.

The simulation scripts are located under the `sim/ed_sim` directory, in separate folders named after each supported simulator.

5.3.4. Functional Simulation with Verilog HDL

Simulation scripts for the Synopsys, Cadence, Aldec, and Siemens EDA simulators are provided for you to run the example design.

The simulation scripts are located in the following main folder locations:

Simulation scripts in the simulation folders are located as follows:

- `sim\ed_sim\mentor\msim_setup.tcl`
- `sim\ed_sim\synopsys\vcs\vcs_setup.sh`
- `sim\ed_sim\synopsys\vcsmx\vcsmx_setup.sh`
- `sim\ed_sim\aldec\rivierapro_setup.tcl`
- `sim\ed_sim\cadence\xcelium_setup.sh`

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Intel Quartus Prime Pro Edition User Guide, Third-party Simulation*.

Related Information

Intel Quartus Prime Pro Edition User Guide: Third-party Simulation

5.3.5. Functional Simulation with VHDL

The EMIF VHDL fileset is provided for customers that wish to generate the top-level RTL instance of their EMIF IP cores in VHDL.

Prior to Intel Quartus Prime version 15.1, the VHDL fileset was comprised entirely of VHDL files. Beginning with Intel Quartus Prime version 15.1, only the top-level IP instance file is guaranteed to be written in VHDL; submodules can still be deployed as Verilog/SystemVerilog (encrypted or plain text) files, or VHDL files. Note that the Questa - Intel FPGA Edition is no longer restricted to a single HDL language as of Intel Quartus Prime 15.1; however, some files may still be encrypted in order to be excluded from the maximum unencrypted module limit of this tool.

Because the VHDL fileset consists of both VHDL and Verilog files, you must follow certain mixed-language simulation guidelines. The general guideline for mixed-language simulation is that you must always link the Verilog files (whether encrypted or not) against the Verilog version of the libraries, and the VHDL files (whether SimGen-generated or pure VHDL) against the VHDL libraries.

Simulation scripts for the Synopsys, Cadence, Aldec, and Siemens EDA simulators are provided for you to run the example design. These simulation scripts are located in the following main folder locations:

Simulation scripts in the simulation folders are located as follows:

- `sim\ed_sim\mentor\msim_setup.tcl`
- `sim\ed_sim\synopsys\vcsmx\vcsmx_setup.sh`
- `sim\ed_sim\synopsys\vcs\vcs_setup.sh`
- `sim\ed_sim\cadence\xcelium_setup.sh`
- `sim\ed_sim\aldec\rivierapro_setup.tcl`

For more information about simulating Verilog HDL or VHDL designs using command lines, refer to the *Intel Quartus Prime Pro Edition User Guide, Third-party Simulation*.

Related Information

Intel Quartus Prime Pro Edition User Guide: Third-party Simulation

5.3.6. Simulating the Design Example

This topic describes how to simulate the design example in Cadence, Synopsys, Siemens EDA, and Aldec simulators.

To simulate the design example in the Intel Quartus Prime software using the Cadence simulator, follow these steps:

1. At the Linux* shell command prompt, change directory to `sim\ed_sim\cadence`
2. Run the simulation by typing the following command at the command prompt:

```
sh xcelium_setup.sh
```

To simulate the example design in the Intel Quartus Prime software using the Synopsys simulator, follow these steps:

1. At the Linux shell command prompt, change directory to `sim\ed_sim\synopsys\vcsmx`
2. Run the simulation by typing the following command at the command prompt:

```
sh vcsmx_setup.sh
```

To simulate the example design in the Intel Quartus Prime software using the Siemens EDA simulator, follow these steps:

1. At the Linux or Windows shell command prompt, change directory to `sim\ed_sim\mentor`
2. Execute the **msim_setup.tcl** script that automatically compiles and runs the simulation by typing the following command at the Linux or Windows command prompt:

```
vsim -do msim_setup.tcl
```

or

Type the following command at the ModelSim* command prompt:

```
do msim_setup.tcl
```

For more information about simulating the external memory interface using the Siemens EDA simulator, refer to the *Simulating External Memory Interface IP With ModelSim* chapter in the *Intel Stratix 10 External Memory Interfaces IP Design Example User Guide*.

Note: Intel does not provide the `run.do` file for the example design with the EMIF interface.

To simulate the example design in the Intel Quartus Prime software using the Aldec simulator, follow these steps:

1. At the Linux or Windows shell command prompt, change directory to `sim\ed_sim\aldec`
2. Execute the **rivierapro_setup.tcl** script that automatically compiles and runs the simulation by typing the following command at the Linux or Windows command prompt: `vsim -do rivierapro.tcl`
3. To compile and elaborate the design after the script loads, type `ld_debug`.
4. Type `run -all` to run the simulation.

For more information about simulation, refer to the *Simulating Designs* chapter in Volume 3 of the Intel Quartus Prime Handbook.

If your Intel Quartus Prime project appears to be configured correctly but the example testbench still fails, check the known issues on the Intel FPGA Knowledge Base before filing a service request.

Related Information

- Calibration Modes on page 135
- Abstract PHY Simulation on page 136

- Simulation Scripts on page 137
- Functional Simulation with Verilog HDL on page 137
- Functional Simulation with VHDL on page 138
- Simulating Intel FPGA Designs
- Intel FPGA Knowledge Base

5.3.6.1. User-requested Reset in Intel Stratix 10 EMIF IP

The following table summarizes information about the user-requested reset mechanism in the Intel Stratix 10 EMIF IP.

Table 170.

	Description
Reset-related signals	`local_reset_req` (input) `local_reset_done` (output)
When can user logic request a reset?	`local_reset_req` has effect only when `local_reset_done` is high. After device power-on, the `local_reset_done` signal transitions high after the completion of the first calibration, whether the calibration is successful or not. In subsequent calibration in user mode, the `local_reset_done` signal transitions high once the calibration is completed. The `local_reset_done` signal takes more time to transition high in first calibration after device power-on as more operations are required put the PHY into working state.
Is user-requested reset a requirement?	A user-requested reset is optional. The I/O SSM automatically ensures that the memory interface begins from a known state as part of the device power-on sequence. A user-requested reset is necessarily only if the user logic must explicitly reset a memory interface after the device power-on sequence.
When does a user-requested reset actually happen?	A reset request is handled by the I/O SSM. If the I/O SSM receives a reset request from multiple interfaces within the same I/O column, it must serialize the reset sequence of the individual interfaces. You should not make assumptions about when the reset sequence will begin after a request is issued.
Timing requirement and triggering mechanism.	Reset request is sent by transitioning the `local_reset_req` signal from low to high, then keeping the signal at the high state for a minimum of 2 EMIF core clock cycles, then transitioning the signal from high to low. `local_reset_req` is asynchronous in that there is no setup/hold timing to meet, but it must meet the minimum pulse width requirement of 2 EMIF core clock cycles.
How long can an external memory interface be kept in reset?	It is not possible to keep an external memory interface in reset indefinitely. Asserting `local_reset_req` high continuously has no effect as a reset request is completed by a full 0->1->0 pulse.
Delaying initial calibration.	Initial calibration cannot be skipped. The `local_reset_done` signal is driven high only after initial calibration has completed.
Reset scope (within an external memory interface).	Only circuits that are required to restore EMIF to power-up state are reset. Excluded from the reset sequence are the IOSSM, the IOPLL(s), the DLL(s), and the CPA.
Reset scope (within an I/O column).	`local_reset_req` is a per-interface reset.

Method for Initiating a User-requested Reset

Step 1 - Precondition

Before asserting `local_reset_req`, user logic must ensure that the `local_reset_done` signal is high.

As part of the device power-on sequence, the `local_reset_done` signal automatically transitions to high upon the completion of the interface calibration sequence, regardless of whether calibration is successful or not.

Note: When targeting a group of interfaces that share the same core clocks, user logic must ensure that the `local_reset_done` signal of every interface is high.

Step 2 - Reset Request

After the pre-condition is satisfied, user logic can send a reset request by driving the `local_cal_req` signal from low to high and then low again (that is, by sending a pulse of 1).

- The low-to-high and high-to-low transitions can occur asychronously; that is, they need not happen in relation to any clock edges. However, the pulse must meet a minimum pulse width of at least 2 EMIF core clock cycles. For example, if the `emif_usr_clk` has a period of 4ns, then the `local_reset_req` pulse must last at least 8ns (that is, two `emif_usr_clk` periods).
- The reset request is considered complete only after the high-to-low transition. The EMIF IP does not initiate the reset sequence when the `local_reset_req` is simply held high.
- Additional pulses to `local_reset_req` are ignored until the reset sequence is completed.

Optional - Detecting local_reset_done deassertion and assertion

If you want, you can monitor the status of the `local_reset_done` signal to explicitly detect the status of the reset sequence.

- After the EMIF IP receives a reset request, it deasserts the `local_reset_done` signal. After initial power-up calibration, `local_reset_done` is de-asserted only in response to a user-requested reset. The reset sequence is imminent when `local_reset_done` has transitioned to low, although the exact timing depends on the current state of the I/O SSM. As part of the EMIF reset sequence, the core reset signal (`emif_usr_reset_n`, `afi_reset_n`) is driven low. Do not use a register reset by the core reset signal to sample `local_reset_done`.
- After the reset sequence has completed, `local_reset_done` is driven high again. `local_reset_done` being driven high indicates the completion of the reset sequence and the readiness to accept a new reset request; however, it does not imply that calibration was successful or that the hard memory controller is ready to accept requests. For these purposes, user logic must check signals such as `afi_cal_success`, `afi_cal_fail`, and `amm_ready`.

6. Intel Stratix 10 EMIF IP for DDR3

This chapter contains IP parameter descriptions, board skew equations, pin planning information, and board design guidance for Intel Stratix 10 external memory interfaces for DDR3.

6.1. Parameter Descriptions

The following topics describe the parameters available on each tab of the IP parameter editor, which you can use to configure your IP.

6.1.1. Intel Stratix 10 EMIF IP DDR3 Parameters: General

Table 171. Group: General / Interface

Display Name	Description
Configuration	Specifies the configuration of the memory interface. The available options depend on the protocol and the targeted FPGA product. (Identifier: PHY_DDR3_CONFIG_ENUM)
Instantiate two controllers sharing a Ping Pong PHY	Specifies the instantiation of two identical memory controllers that share an address/command bus through the use of Ping Pong PHY. This parameter is available only if you specify the **Hard PHY and Hard Controller** option. When this parameter is enabled, the IP exposes two independent Avalon interfaces to the user logic, and a single external memory interface with double width for the data bus and the CS#, CKE, ODT, and CK/CK# signals. (Identifier: PHY_DDR3_USER_PING_PONG_EN)

Table 172. Group: General / Clocks

Display Name	Description
Memory clock frequency	Specifies the **operating frequency** of the memory interface in MHz. If you change the memory frequency, you should update the memory latency parameters on the **Memory** tab and the memory timing parameters on the **Mem Timing** tab. (Identifier: PHY_DDR3_MEM_CLK_FREQ_MHZ)
Use recommended PLL reference clock frequency	Specifies that the PLL reference clock frequency is automatically calculated for best performance. *If you want to specify a different PLL reference clock frequency, uncheck the check box for this parameter.* (Identifier: PHY_DDR3_DEFAULT_REF_CLK_FREQ)
PLL reference clock frequency	This parameter tells the IP what PLL reference clock frequency the user will supply. Users must select a valid PLL reference clock frequency from the list. The values in the list can change when the memory interface frequency changes and/or the clock rate of user logic changes. It is recommended to use the fastest possible PLL reference clock frequency because it leads to

continued...

Display Name	Description
	better jitter performance. Selection is required only if the user does not check the "Use recommended PLL reference clock frequency" option. (Identifier: PHY_DDR3_USER_REF_CLK_FREQ_MHZ)
PLL reference clock jitter	Specifies the **peak-to-peak jitter** on the PLL reference clock source. The clock source of the PLL reference clock must meet or exceed the following jitter requirements: 10ps peak to peak, or 1.42ps RMS at 1e-12 BER, 1.22ps at 1e-16 BER. (Identifier: PHY_DDR3_REF_CLK_JITTER_PS)
Clock rate of user logic	Specifies the relationship between the user logic clock frequency and the memory clock frequency. For example, if the memory clock sent from the FPGA to the memory device is toggling at 800MHz, a quarter-rate interface means that the user logic in the FPGA runs at 200MHz. The list of available options is dependent on the memory protocol and device family. (Identifier: PHY_DDR3_RATE_ENUM)
Core clocks sharing	When a design contains multiple interfaces of the same protocol, rate, frequency, and PLL reference clock source, they can share a common set of core clock domains. By sharing core clock domains, they **reduce clock network usage and avoid clock synchronization logic between the interfaces**. To share core clocks, denote one of the interfaces as **"Master"**, and the remaining interfaces as **"Slave"**. In the RTL, connect the `clks_sharing_master_out` signal from the master interface to the `clks_sharing_slave_in` signal of all the slave interfaces. Both master and slave interfaces still expose their own output clock ports in the RTL (for example, `emif_usr_clk`, `afi_clk`), but the physical signals are equivalent, hence it does not matter whether a clock port from a master or a slave is used. *As the combined width of all interfaces sharing the same core clock increases, you may encounter timing closure difficulty for transfers between the FPGA core and the periphery.* (Identifier: PHY_DDR3_CORE_CLKS_SHARING_ENUM)
Export clks_sharing_slave_out to facilitate multi-slave connectivity	When more than one slave exist, you can either connect the clks_sharing_master_out interface from the master to the clks_sharing_slave_in interface of all the slaves (i.e. one-to-many topology), OR, you can connect the clks_sharing_master_out interface to one slave, and connect the clks_sharing_slave_out interface of that slave to the next slave (i.e. daisy-chain topology). Both approaches produce the same result. The daisy-chain approach may be easier to achieve in the Platform Designer tool, whereas the one-to-many approach may be more intuitive. (Identifier: PHY_DDR3_CORE_CLKS_SHARING_EXPOSE_SLAVE_OUT)
Specify additional core clocks based on existing PLL	Displays additional parameters allowing you to create additional output clocks based on the existing PLL. This parameter **provides an alternative clock-generation mechanism for when your design exhausts available PLL resources**. The additional output clocks that you create can be fed into the core. Clock signals created with this parameter are synchronous to each other, but asynchronous to the memory interface core clock domains (such as `emif_usr_clk` or `afi_clk`). *You must follow proper clock-domain-crossing techniques when transferring data between clock domains.* (Identifier: PLL_ADD_EXTRA_CLKS)

Table 173. Group: General / Clocks / Additional Core Clocks

Display Name	Description
Number of additional core clocks	Specifies the number of additional output clocks to create from the PLL. (Identifier: PLL_USER_NUM_OF_EXTRA_CLKS)

Table 174. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_0

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_5)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_5)

Table 175. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_1

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_6)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_6)

Table 176. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_2

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_7)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_7)

Table 177. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_3

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_8)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_8)

6.1.2. Intel Stratix 10 EMIF IP DDR3 Parameters: FPGA I/O

You should use Hyperlynx* or similar simulators to determine the best settings for your board. Refer to the EMIF Simulation Guidance wiki page for additional information.

Table 178. Group: FPGA I/O / FPGA I/O Settings

Display Name	Description
Voltage	The voltage level for the I/O pins driving the signals between the memory device and the FPGA memory interface. (Identifier: PHY_DDR3_IO_VOLTAGE)
Use default I/O settings	Specifies that a legal set of I/O settings are automatically selected. The default I/O settings are not necessarily optimized for a specific board. *To achieve optimal signal integrity, perform I/O simulations with IBIS models and enter the I/O settings manually, based on simulation results*. (Identifier: PHY_DDR3_DEFAULT_IO)

Table 179. Group: FPGA I/O / FPGA I/O Settings / Address/Command

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the address/command pins of the memory interface. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR3_USER_AC_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR3_USER_AC_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_DDR3_USER_AC_SLEW_RATE_ENUM)

Table 180. Group: FPGA I/O / FPGA I/O Settings / Memory Clock

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the memory clock pins. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR3_USER_CK_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR3_USER_CK_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_DDR3_USER_CK_SLEW_RATE_ENUM)

Table 181. Group: FPGA I/O / FPGA I/O Settings / Data Bus

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the data and data clock/strobe pins of the memory interface. The selected I/O standard option configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR3_USER_DATA_IO_STD_ENUM)
Output mode	This parameter allows you to change the output current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR3_USER_DATA_OUT_MODE_ENUM)
Input mode	This parameter allows you to change the input termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR3_USER_DATA_IN_MODE_ENUM)

Table 182. Group: FPGA I/O / FPGA I/O Settings / PHY Inputs

Display Name	Description
PLL reference clock I/O standard	Specifies the I/O standard for the PLL reference clock of the memory interface. (Identifier: PHY_DDR3_USER_PLL_REF_CLK_IO_STD_ENUM)
RZQ I/O standard	Specifies the I/O standard for the RZQ pin used in the memory interface. (Identifier: PHY_DDR3_USER_RZQ_IO_STD_ENUM)

6.1.3. Intel Stratix 10 EMIF IP DDR3 Parameters: Memory

Table 183. Group: Memory / Topology

Display Name	Description
Memory format	Specifies the format of the external memory device. The following formats are supported: **Component** - a Discrete memory device; **UDIMM** - Unregistered/Unbuffered DIMM where address/control, clock, and data are unbuffered; **RDIMM** - Registered DIMM where address/control and clock are buffered; **SODIMM** - Small Outline DIMM is similar to UDIMM but smaller in size and is typically used for systems with limited space. Some memory protocols may not be available in all formats. (Identifier: MEM_DDR3_FORMAT_ENUM)
DQ width	Specifies the total number of data pins in the interface. (Identifier: MEM_DDR3_DQ_WIDTH)
DQ pins per DQS group	Specifies the total number of DQ pins per DQS group. (Identifier: MEM_DDR3_DQ_PER_DQS)
Number of clocks	Specifies the number of CK/CK# clock pairs exposed by the memory interface. Usually more than 1 pair is required for RDIMM/LRDIMM formats. The value of this parameter depends on the memory device selected; *refer to the data sheet for your memory device.* (Identifier: MEM_DDR3_CK_WIDTH)
Number of chip selects	Specifies the total number of chip selects in the interface, up to a maximum of 4. This parameter applies to **discrete components only**. (Identifier: MEM_DDR3_DISCRETE_CS_WIDTH)
Number of DIMMs	Total number of DIMMs. (Identifier: MEM_DDR3_NUM_OF_DIMMS)
Number of physical ranks per DIMM	Number of ranks per DIMM. For LRDIMM, this represents the number of physical ranks on the DIMM behind the memory buffer (Identifier: MEM_DDR3_RANKS_PER_DIMM)
Row address width	Specifies the number of row address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of address pins needed for access to all available rows. (Identifier: MEM_DDR3_ROW_ADDR_WIDTH)
Column address width	Specifies the number of column address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of address pins needed for access to all available columns. (Identifier: MEM_DDR3_COL_ADDR_WIDTH)
Bank address width	Specifies the number of bank address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of bank address pins needed for access to all available banks. (Identifier: MEM_DDR3_BANK_ADDR_WIDTH)
Enable DM pins	Indicates whether the interface uses data mask (DM) pins. This feature allows specified portions of the data bus to be written to memory (not available in x4 mode). **One DM pin exists per DQS group.** (Identifier: MEM_DDR3_DM_EN)
Enable address mirroring for odd chip-selects	Enabling address mirroring for multi-CS discrete components. Typically used when components are arranged in a clamshell layout. (Identifier: MEM_DDR3_DISCRETE_MIRROR_ADDRESSING_EN)
Enable address mirroring for odd ranks	Enabling address mirroring for dual-rank or quad-rank DIMM. (Identifier: MEM_DDR3_MIRROR_ADDRESSING_EN)
ALERT# pin placement	Specifies placement for the `mem_alert_n` signal. You can select "**I/O Lane with Address/Command Pins**" or "**I/O Lane with DQS Group**". If you select "**I/O Lane with DQS Group**", you can specify the DQS group with which to place the mem_alert_n pin. For optimum signal integrity, you

continued...

Display Name	Description
	should choose "**I/O Lane with Address/Command Pins**". For interfaces containing multiple memory devices, it is recommended to connect the ALERT# pins together to the ALERT# pin on the FPGA. (Identifier: MEM_DDR3_ALERT_N_PLACEMENT_ENUM)
DQS group of ALERT#	Select the DQS group with which the ALERT# pin is placed. (Identifier: MEM_DDR3_ALERT_N_DQS_GROUP)

Table 184. Group: Memory / Latency and Burst

Display Name	Description
Memory CAS latency setting	Specifies the number of clock cycles between the read command and the availability of the first bit of output data at the memory device. Overall read latency equals the additive latency (AL) + the CAS latency (CL). *Overall read latency depends on the memory device selected; refer to the datasheet for your device.* (Identifier: MEM_DDR3_TCL)
Memory write CAS latency setting	Specifies the number of clock cycles from the release of internal write to the latching of the first data in at the memory device. *This value depends on the memory device selected; refer to the datasheet for your device.* (Identifier: MEM_DDR3_WTCL)
Memory additive CAS latency setting	Determines the posted CAS additive latency of the memory device. Enable this feature to **improve command and bus efficiency, and increase system bandwidth**. (Identifier: MEM_DDR3_ATCL_ENUM)

Table 185. Group: Memory / Mode Register Settings

Display Name	Description
Hide advanced mode register settings	Show or hide advanced mode register settings. Changing advanced mode register settings to non-default values is strongly discouraged. (Identifier: MEM_DDR3_HIDE_ADV_MR_SETTINGS)
Burst Length	Specifies the DRAM burst length which determines how many consecutive addresses should be accessed for a given read/write command. (Identifier: MEM_DDR3_BL_ENUM)
Read Burst Type	Indicates whether accesses within a given burst are in sequential or interleaved order. Select sequential if you are using the Intel-provided memory controller. (Identifier: MEM_DDR3_BT_ENUM)
DLL precharge power down	Specifies whether the DLL in the memory device is off or on during precharge power-down (Identifier: MEM_DDR3_PD_ENUM)
Enable the DLL in memory device	Enable the DLL in memory device (Identifier: MEM_DDR3_DLL_EN)
Auto self-refresh method	Indicates whether to enable or disable auto self-refresh. Auto self-refresh allows the controller to issue self-refresh requests, rather than manually issuing self-refresh in order for memory to retain data. (Identifier: MEM_DDR3_ASR_ENUM)
Self-refresh temperature	Specifies the self-refresh temperature as "**Normal**" or "**Extended**" mode. *More information on Normal and Extended temperature modes can be found in the memory device datasheet.* (Identifier: MEM_DDR3_SRT_ENUM)
DDR3 RDIMM/LRDIMM control words	*Each 4-bit/8-bit setting can be obtained from the manufacturer's data sheet and should be entered in hexadecimal*, starting with the 8-bit setting RCBx on the left and continuing to RC1x followed by the 4-bit setting RCOF and ending with RC00 on the right (Identifier: MEM_DDR3_RDIMM_CONFIG)
DDR3 LRDIMM additional control words	Each 4-bit setting can be obtained from the manufacturer's data sheet and should be entered in hexadecimal, starting with BC0F on the left and ending with BC00 on the right (Identifier: MEM_DDR3_LRDIMM_EXTENDED_CONFIG)

6.1.4. Intel Stratix 10 EMIF IP DDR3 Parameters: Mem I/O

Table 186. Group: Mem I/O / Memory I/O Settings

Display Name	Description
Output drive strength setting	Specifies the output driver impedance setting at the memory device. *To obtain optimum signal integrity performance, select option based on board simulation results.* (Identifier: MEM_DDR3_DRV_STR_ENUM)
ODT Rtt nominal value	Determines the nominal on-die termination value applied to the DRAM. The termination is applied any time that ODT is asserted. If you specify a different value for RTT_WR, that value takes precedence over the values mentioned here. *For optimum signal integrity performance, select your option based on board simulation results.* (Identifier: MEM_DDR3_RTT_NOM_ENUM)
Dynamic ODT (Rtt_WR) value	Specifies the mode of the dynamic on-die termination (ODT) during writes to the memory device (used for multi-rank configurations). *For optimum signal integrity performance, select this option based on board simulation results.* (Identifier: MEM_DDR3_RTT_WR_ENUM)

Table 187. Group: Mem I/O / ODT Activation

Display Name	Description
Use Default ODT Assertion Tables	Enables the default ODT assertion pattern as determined from vendor guidelines. These settings are provided as a default only; *you should simulate your memory interface to determine the optimal ODT settings and assertion patterns.* (Identifier: MEM_DDR3_USE_DEFAULT_ODT)

6.1.5. Intel Stratix 10 EMIF IP DDR3 Parameters: Mem Timing

These parameters should be read from the table in the datasheet associated with the speed bin of the memory device (not necessarily the frequency at which the interface is running).

Table 188. Group: Mem Timing / Parameters dependent on Speed Bin

Display Name	Description
Speed bin	The speed grade of the memory device used. This parameter refers to the maximum rate at which the memory device is specified to run. (Identifier: MEM_DDR3_SPEEDBIN_ENUM)
tIS (base)	tIS (base) refers to the **setup time for the Address/Command/Control (A) bus** to the rising edge of CK. (Identifier: MEM_DDR3_TIS_PS)
tIS (base) AC level	tIS (base) AC level refers to the **voltage level which the address/ command signal must cross and remain above during the setup margin window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire setup period. (Identifier: MEM_DDR3_TIS_AC_MV)
tIH (base)	tIH (base) refers to the **hold time for the Address/Command (A) bus** after the rising edge of CK. Depending on what AC level the user has chosen for a design, the hold margin can vary (this variance will be automatically determined when the user chooses the "**tIH (base) AC level**"). (Identifier: MEM_DDR3_TIH_PS)

continued...

Display Name	Description
tIH (base) DC level	tIH (base) DC level refers to the **voltage level which the address/ command signal must not cross during the hold window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire hold period. (Identifier: MEM_DDR3_TIH_DC_MV)
tDS (base)	tDS(base) refers to the **setup time for the Data(DQ) bus** before the rising edge of the DQS strobe. (Identifier: MEM_DDR3_TDS_PS)
tDS (base) AC level	tDS (base) AC level refers to **the voltage level which the data bus must cross and remain above during the setup margin window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire setup period. (Identifier: MEM_DDR3_TDS_AC_MV)
tDH (base)	tDH (base) refers to the **hold time for the Data (DQ) bus** after the rising edge of CK. (Identifier: MEM_DDR3_TDH_PS)
tDH (base) DC level	tDH (base) DC level refers to the **voltage level which the data bus must not cross during the hold window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire hold period. (Identifier: MEM_DDR3_TDH_DC_MV)
tDQSQ	tDQSQ describes **the latest valid transition of the associated DQ pins for a READ**. tDQSQ specifically refers to the DQS, DQS# to DQ skew. It is the length of time between the DQS, DQS# crossing to the last valid transition of the slowest DQ pin in the DQ group associated with that DQS strobe. (Identifier: MEM_DDR3_TDQSQ_PS)
tQH	tQH specifies the **output hold time for the DQ in relation to DQS, DQS#**. It is the length of time between the DQS, DQS# crossing to the earliest invalid transition of the fastest DQ pin in the DQ group associated with that DQS strobe. (Identifier: MEM_DDR3_TQH_CYC)
tDQSCK	tDQSCK describes the **skew between the memory clock (CK) and the input data strobes (DQS) used for reads**. It is the time between the rising data strobe edge (DQS, DQS#) relative to the rising CK edge. (Identifier: MEM_DDR3_TDQSCK_PS)
tDQSS	tDQSS describes the **skew between the memory clock (CK) and the output data strobes used for writes**. It is the time between the rising data strobe edge (DQS, DQS#) relative to the rising CK edge. (Identifier: MEM_DDR3_TDQSS_CYC)
tQSH	tQSH refers to the differential High Pulse Width, which is measured as a percentage of tCK. It is **the time during which the DQS is high for a read**. (Identifier: MEM_DDR3_TQSH_CYC)
tDSH	tDSH specifies the **write DQS hold time**. This is the time difference between the rising CK edge and the falling edge of DQS, measured as a percentage of tCK. (Identifier: MEM_DDR3_TDSH_CYC)
tWLS	tWLS describes the **write leveling setup time**. It is measured from the rising edge of CK to the rising edge of DQS. (Identifier: MEM_DDR3_TWLS_PS)
tWLH	tWLH describes the **write leveling hold time**. It is measured from the rising edge of DQS to the rising edge of CK (Identifier: MEM_DDR3_TWLH_PS)
tDSS	tDSS describes the **time between the falling edge of DQS to the rising edge of the next CK transition**. (Identifier: MEM_DDR3_TDSS_CYC)
tINIT	tINIT describes the **time duration of the memory initialization after a device power-up**. After RESET_n is de-asserted, wait for another 500us until CKE becomes active. During this time, the DRAM starts internal initialization; this happens independently of external clocks. (Identifier: MEM_DDR3_TINIT_US)

continued...

Display Name	Description
tMRD	The mode register set command cycle time, tMRD is the **minimum time period required between two MRS commands**. (Identifier: MEM_DDR3_TMRD_CK_CYC)
tRAS	tRAS describes the **activate to precharge duration**. A row cannot be deactivated until the tRAS time has been met. Therefore tRAS determines how long the memory has to wait after a activate command before a precharge command can be issued to close the row. (Identifier: MEM_DDR3_TRAS_NS)
tRCD	tRCD, **row command delay**, describes **the active to read/write time**. It is the amount of delay between the activation of a row through the RAS command and the access to the data through the CAS command. (Identifier: MEM_DDR3_TRCD_NS)
tRP	tRP refers to the **Precharge (PRE) command period**. It describes how long it takes for the memory to disable access to a row by precharging and before it is ready to activate a different row. (Identifier: MEM_DDR3_TRP_NS)
tWR	tWR refers to the **Write Recovery time**. It specifies the amount of clock cycles needed to complete a write before a precharge command can be issued. (Identifier: MEM_DDR3_TWR_NS)

Table 189. Group: Mem Timing / Parameters dependent on Speed Bin, Operating Frequency, and Page Size

Display Name	Description
tRRD	tRRD refers to the **Row Active to Row Active Delay**. It is the minimum time interval (measured in memory clock cycles) between two activate commands to rows in different banks in the same rank (Identifier: MEM_DDR3_TRRD_CYC)
tFAW	tFAW refers to the **four activate window time**. It describes the period of time during which only four banks can be active. (Identifier: MEM_DDR3_TFAW_NS)
tWTR	tWTR or **Write Timing Parameter** describes the **delay from start of internal write transaction to internal read command, for accesses to the same bank**. The delay is measured from the first rising memory clock edge after the last write data is received to the rising memory clock edge when a read command is received. (Identifier: MEM_DDR3_TWTR_CYC)
tRTP	tRTP refers to the **internal READ Command to PRECHARGE Command delay**. It is the number of memory clock cycles that is needed between a read command and a precharge command to the same rank. (Identifier: MEM_DDR3_TRTP_CYC)

Table 190. Group: Mem Timing / Parameters dependent on Density and Temperature

Display Name	Description
tRFC	tRFC refers to the **Refresh Cycle Time**. It is the amount of delay after a refresh command before an activate command can be accepted by the memory. This parameter is dependent on the memory density and is necessary for proper hardware functionality. (Identifier: MEM_DDR3_TRFC_NS)
tREFI	tREFI refers to the **average periodic refresh interval**. It is the maximum amount of time the memory can tolerate in between each refresh command (Identifier: MEM_DDR3_TREFI_US)

6.1.6. Intel Stratix 10 EMIF IP DDR3 Parameters: Board

Table 191. Group: Board / Intersymbol Interference/Crosstalk

Display Name	Description
Use default ISI/crosstalk values	You can enable this option to use default intersymbol interference and crosstalk values for your topology. Note that the default values are not optimized for your board. *For optimal signal integrity, it is recommended that you do not enable this parameter, but instead perform I/O simulation using IBIS models and Hyperlynx*, and manually enter values based on your simulation results, instead of using the default values.* (Identifier: BOARD_DDR3_USE_DEFAULT_ISI_VALUES)
Address and command ISI/crosstalk	The address and command window reduction due to ISI and crosstalk effects. The number to be entered is the **total loss of margin on both the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR3_USER_AC_ISI_NS)
Read DQS/DQS# ISI/crosstalk	The reduction of the read data window due to ISI and crosstalk effects on the DQS/DQS# signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR3_USER_RCLK_ISI_NS)
Read DQ ISI/crosstalk	The reduction of the read data window due to ISI and crosstalk effects on the DQ signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on the setup and hold side (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR3_USER_RDATA_ISI_NS)
Write DQS/DQS# ISI/crosstalk	The reduction of the write data window due to ISI and crosstalk effects on the DQS/DQS# signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR3_USER_WCLK_ISI_NS)
Write DQ ISI/crosstalk	The reduction of the write data window due to ISI and crosstalk effects on the DQ signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR3_USER_WDATA_ISI_NS)

Table 192. Group: Board / Board and Package Skews

Display Name	Description
Package deskewed with board layout (DQS group)	Enable this parameter if you are compensating for package skew on the DQ, DQS, and DM buses in the board layout. **Include package skew in calculating the following board skew parameters.** (Identifier: BOARD_DDR3_IS_SKEW_WITHIN_DQS_DESKEWED)
Maximum board skew within DQS group	The largest skew between all DQ and DM pins in a DQS group. This value affects the read capture and write margins. (Identifier: BOARD_DDR3_BRD_SKEW_WITHIN_DQS_NS)
Maximum system skew within DQS group	The largest skew between all DQ and DM pins in a DQS group. Enter combined board and package skew. This value affects the read capture and write margins. (Identifier: BOARD_DDR3_PKG_BRD_SKEW_WITHIN_DQS_NS)
Package deskewed with board layout (address/command bus)	Enable this parameter if you are compensating for package skew on the address, command, control, and memory clock buses in the board layout. **Include package skew in calculating the following board skew parameters.** (Identifier: BOARD_DDR3_IS_SKEW_WITHIN_AC_DESKEWED)

continued...

Display Name	Description
Maximum board skew within address/ command bus	The largest skew between the address and command signals. Enter the board skew only; package skew is calculated automatically, based on the memory interface configuration, and added to this value. (Identifier: BOARD_DDR3_BRD_SKEW_WITHIN_AC_NS)
Maximum system skew within address/ command bus	Maximum system skew within address/command bus refers to the largest skew between the address and command signals. (Identifier: BOARD_DDR3_PKG_BRD_SKEW_WITHIN_AC_NS)
Average delay difference between DQS and CK	The average delay difference between the DQS signals and the CK signal, calculated by averaging the longest and smallest DQS trace delay minus the CK trace delay. Positive values represent DQS signals that are longer than CK signals and negative values represent DQS signals that are shorter than CK signals. (Identifier: BOARD_DDR3_DQS_TO_CK_SKEW_NS)
Maximum delay difference between DIMMs/devices	The largest propagation delay on DQ signals between ranks (*applicable only when there is more than one rank*). For example: when you configure two ranks using one DIMM there is a short distance between the ranks for the same DQ pin; when you implement two ranks using two DIMMs the distance is larger. (Identifier: BOARD_DDR3_SKEW_BETWEEN_DIMMS_NS)
Maximum skew between DQS groups	The largest skew between DQS signals. (Identifier: BOARD_DDR3_SKEW_BETWEEN_DQS_NS)
Average delay difference between address/command and CK	The average delay difference between the address/command signals and the CK signal, calculated by averaging the longest and smallest address/ command signal trace delay minus the maximum CK trace delay. Positive values represent address and command signals that are longer than CK signals and negative values represent address and command signals that are shorter than CK signals. (Identifier: BOARD_DDR3_AC_TO_CK_SKEW_NS)
Maximum CK delay to DIMM/device	The delay of the longest CK trace from the FPGA to any DIMM/device. (Identifier: BOARD_DDR3_MAX_CK_DELAY_NS)
Maximum DQS delay to DIMM/device	The delay of the longest DQS trace from the FPGA to any DIMM/device (Identifier: BOARD_DDR3_MAX_DQS_DELAY_NS)

6.1.7. Intel Stratix 10 EMIF IP DDR3 Parameters: Controller

Table 193. Group: Controller / Low Power Mode

Display Name	Description
Enable Auto Power-Down	Enable this parameter to have the controller automatically place the memory device into power-down mode after a specified number of idle controller clock cycles. The idle wait time is configurable. **All ranks must be idle to enter auto power-down.** (Identifier: CTRL_DDR3_AUTO_POWER_DOWN_EN)
Auto Power-Down Cycles	Specifies the number of idle controller cycles after which the memory device is placed into power-down mode. You can configure the idle waiting time. The supported range for number of cycles is from 1 to 65534. (Identifier: CTRL_DDR3_AUTO_POWER_DOWN_CYCS)

Table 194. Group: Controller / Efficiency

Display Name	Description
Enable User Refresh Control	When enabled, user logic has complete control and is responsible for issuing adaquate refresh commands to the memory devices, via the MMR interface. This feature provides increased control over worst-case read latency and enables you to issue refresh bursts during idle periods. (Identifier: CTRL_DDR3_USER_REFRESH_EN)
Enable Auto-Precharge Control	Select this parameter to enable the auto-precharge control on the controller top level. If you assert the auto-precharge control signal while requesting a read or write burst, you can specify whether the controller should close (auto-precharge) the currently open page at the end of the read or write burst, potentially making a future access to a different page of the same bank faster. (Identifier: CTRL_DDR3_AUTO_PRECHARGE_EN)
Address Ordering	Controls the mapping between Avalon addresses and memory device addresses. By changing the value of this parameter, you can change the mappings between the Avalon-MM address and the DRAM address. (Identifier: CTRL_DDR3_ADDR_ORDER_ENUM)
Enable Reordering	Enable this parameter to allow the controller to perform command and data reordering. **Reordering can improve efficiency by reducing bus turnaround time and row/bank switching time.** Data reordering allows the single-port memory controller to change the order of read and write commands to achieve highest efficiency. Command reordering allows the controller to issue bank management commands early based on incoming patterns, so that the desired row in memory is already open when the command reaches the memory interface. *For more information, refer to the Data Reordering topic in the EMIF Handbook.* (Identifier: CTRL_DDR3_REORDER_EN)
Starvation limit for each command	Specifies the **number of commands that can be served before a waiting command is served**. The controller employs a counter to ensure that all requests are served after a pre-defined interval -- this ensures that low priority requests are not ignored, when doing data reordering for efficiency. The valid range for this parameter is from 1 to 63. *For more information, refer to the Starvation Control topic in the EMIF Handbook.* (Identifier: CTRL_DDR3_STARVE_LIMIT)
Enable Command Priority Control	Select this parameter to enable user-requested command priority control on the controller top level. This parameter instructs the controller to treat a read or write request as high-priority. The controller attempts to fill high-priority requests sooner, to reduce latency. **Connect this interface to the conduit of your logic block that determines when the external memory interface IP treats the read or write request as a high-priority command.** (Identifier: CTRL_DDR3_USER_PRIORITY_EN)

Table 195. Group: Controller / Configuration, Status and Error Handling

Display Name	Description
Enable Memory-Mapped Configuration and Status Register (MMR) Interface	Enable this parameter to change or read memory timing parameters, memory address size, mode register settings, controller status, and request sideband operations. (Identifier: CTRL_DDR3_MMR_EN)
Enable Error Detection and Correction Logic with ECC	Enables error-correction code (ECC) for **single-bit error correction and double-bit error detection**. *ECC is implemented as soft logic.* (Identifier: CTRL_DDR3_ECC_EN)
Enable Auto Error Correction to External Memory	Specifies that the controller automatically schedule and perform a write back to the external memory when a single-bit error is detected. Regardless of whether the option is enabled or disabled, the ECC feature always corrects single-bit errors before returning the read data to user logic. (Identifier: CTRL_DDR3_ECC_AUTO_CORRECTION_EN)
Enable ctrl_ecc_readdataerror signal to indicate uncorrectable data errors	Select this option to enable the `ctrl_ecc_readdataerror` signal on the controller top level. The signal has the same timing as the read data valid signal of the Controller Avalon Memory-Mapped interface, and is asserted

continued...

Display Name	Description
	high to indicate that the read data returned by the Controller in the same cycle contains errors uncorrectable by the ECC logic. (Identifier: CTRL_DDR3_ECC_READDATAERROR_EN)

Table 196. Group: Controller / Data Bus Turnaround Time

Display Name	Description
Additional read-to-write turnaround time (same rank)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from **a read to a write within the same logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR3_RD_TO_WR_SAME_CHIP_DELTA_CYCS)
Additional write-to-read turnaround time (same rank)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **write to a read within the same logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR3_WR_TO_RD_SAME_CHIP_DELTA_CYCS)
Additional read-to-read turnaround time (different ranks)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **read of one logical rank to a read of another logical rank**. This can resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR3_RD_TO_RD_DIFF_CHIP_DELTA_CYCS)
Additional read-to-write turnaround time (different ranks)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **read of one logical rank to a write of another logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR3_RD_TO_WR_DIFF_CHIP_DELTA_CYCS)
Additional write-to-write turnaround time (different ranks)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **write of one logical rank to a write of another logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR3_WR_TO_WR_DIFF_CHIP_DELTA_CYCS)
Additional write-to-read turnaround time (different ranks)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **write of one logical rank to a read of another logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR3_WR_TO_RD_DIFF_CHIP_DELTA_CYCS)

6.1.8. Intel Stratix 10 EMIF IP DDR3 Parameters: Diagnostics

Table 197. Group: Diagnostics / Simulation Options

Display Name	Description
Calibration mode	Specifies whether to **skip memory interface calibration** during simulation, or to **simulate the full calibration** process.

continued...

Display Name	Description
	Simulating the full calibration process can take hours (or even days), depending on the width and depth of the memory interface. You can achieve much faster simulation times by skipping the calibration process, but that is only expected to work when the memory model is ideal and the interconnect delays are zero. *If you enable this parameter, the interface still performs some memory initialization before starting normal operations.* Abstract PHY is supported with skip calibration. (Identifier: DIAG_DDR3_SIM_CAL_MODE_ENUM)
Abstract phy for fast simulation	Specifies that the system use Abstract PHY for simulation. **Abstract PHY replaces the PHY with a model for fast simulation and can reduce simulation time by 3-10 times.** Abstract PHY is available for certain protocols and device families, and only when you select **Skip Calibration**. (Identifier: DIAG_DDR3_ABSTRACT_PHY)
Show verbose simulation debug messages	This option allows adjusting the verbosity of the simulation output messages. (Identifier: DIAG_DDR3_SIM_VERBOSE)

Table 198. Group: Diagnostics / Calibration Debug Options

Display Name	Description
Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies the connectivity of an Avalon slave interface for use by the Quartus Prime EMIF Debug Toolkit or user core logic. If you set this parameter to "**Disabled**", no debug features are enabled. If you set this parameter to "**Export**", an Avalon slave interface named "`cal_debug`" is exported from the IP. To use this interface with the EMIF Debug Toolkit, you must instantiate and connect an EMIF debug interface IP core to it, or connect it to the `cal_debug_out` interface of another EMIF core. If you select "**Add EMIF Debug Interface**", an EMIF debug interface component containing a JTAG Avalon Master is connected to the debug port, allowing the core to be accessed by the EMIF Debug Toolkit. *Only one EMIF debug interface should be instantiated per I/O column.* You can chain additional EMIF or PHYLite cores to the first by enabling the "**Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option for all cores in the chain, and selecting "**Export**" for the "**Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option on all cores after the first. (Identifier: DIAG_DDR3_EXPORT_SEQ_AVALON_SLAVE)
Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies that the IP export an Avalon-MM master interface (`cal_debug_out`) which can connect to the `cal_debug` interface of other EMIF cores residing in the same I/O column. **This parameter applies only if the EMIF Debug Toolkit or On-Chip Debug Port is enabled.** *Refer to the Debugging Multiple EMIFs wiki page for more information about debugging multiple EMIFs.* (Identifier: DIAG_DDR3_EXPORT_SEQ_AVALON_MASTER)
First EMIF Instance in the Avalon Chain	If selected, this EMIF instance will be the head of the Avalon interface chain connected to the master. For simulation purposes it is needed to identify the first EMIF instance in the avalon Chain. (Identifier: DIAG_DDR3_EXPORT_SEQ_AVALON_HEAD_OF_CHAIN)
Interface ID	Identifies interfaces within the I/O column, for use by the EMIF Debug Toolkit and the On-Chip Debug Port. Interface IDs should be unique among EMIF cores within the same I/O column. If the **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** parameter is set to **Disabled**, the interface ID is unused. (Identifier: DIAG_DDR3_INTERFACE_ID)
Use Soft NIOS Processor for On-Chip Debug	Enables a soft Nios processor as a peripheral component to access the **On-Chip Debug Port**. *Only one interface in a column can activate this option.* (Identifier: DIAG_SOFT_NIOS_MODE)

Table 199. Group: Diagnostics / Example Design

Display Name	Description
Number of core clocks sharing slaves to instantiate in the example design	Specifies the number of core clock sharing slaves to instantiate in the example design. This parameter applies only if you set the "**Core clocks sharing**" parameter in the "**General**" tab to "**Master**" or "**Slave**". (Identifier: DIAG_DDR3_EX_DESIGN_NUM_OF_SLAVES)
Enable In-System-Sources-and-Probes	Enables In-System-Sources-and-Probes in the example design for *common debug signals, such as calibration status or example traffic generator per-bit status*. This parameter must be enabled if you want to do driver margining using the EMIF Debug Toolkit. (Identifier: DIAG_DDR3_EX_DESIGN_ISSP_EN)

Table 200. Group: Diagnostics / Traffic Generator

Display Name	Description
Use configurable Avalon traffic generator 2.0	This option allows users to add the new configurable Avalon traffic generator to the example design. (Identifier: DIAG_DDR3_USE_TG_AVL_2)
Bypass the default traffic pattern	Specifies that the controller/interface bypass the traffic generator 2.0 default pattern after reset. If you do not enable this parameter, the traffic generator does not assert a pass or fail status until the generator is configured and signaled to start by its Avalon configuration interface. (Identifier: DIAG_DDR3_BYPASS_DEFAULT_PATTERN)
Bypass the user-configured traffic stage	Specifies that the controller/interface bypass the user-configured traffic generator's pattern after reset. If you do not enable this parameter, the traffic generator does not assert a pass or fail status until the generator is configured and signaled to start by its Avalon configuration interface. Configuration can be done by connecting to the traffic generator via the EMIF Debug Toolkit, or by using custom logic connected to the Avalon-MM configuration slave port on the traffic generator. Configuration can also be simulated using the example testbench provided in the `altera_emif_avl_tg_2_tb.sv` file. (Identifier: DIAG_DDR3_BYPASS_USER_STAGE)
Bypass the traffic generator repeated-writes/repeated-reads test pattern	Specifies that the controller/interface bypass the traffic generator's repeat test stage. *If you do not enable this parameter, every write and read is repeated several times.* (Identifier: DIAG_DDR3_BYPASS_REPEAT_STAGE)
Bypass the traffic generator stress pattern	Specifies that the controller/interface bypass the traffic generator's stress pattern stage. (Stress patterns are meant to create worst-case signal integrity patterns on the data pins.) If you do not enable this parameter, the traffic generator does not assert a pass or fail status until the generator is configured and signaled to start by its Avalon configuration interface. (Identifier: DIAG_DDR3_BYPASS_STRESS_STAGE)
Run diagnostic on infinite test duration	Specifies that the traffic generator run indefinitely until the first error is detected. (Identifier: DIAG_DDR3_INFI_TG2_ERR_TEST)
Export Traffic Generator 2.0 configuration interface	Specifies that the IP export an Avalon-MM slave port for configuring the Traffic Generator. *This is required only if you are configuring the traffic generator through user logic and not through through the EMIF Debug Toolkit.* (Identifier: DIAG_TG_AVL_2_EXPORT_CFG_INTERFACE)

Table 201. Group: Diagnostics / Performance

Display Name	Description
Enable Efficiency Monitor	Adds an Efficiency Monitor component to the Avalon-MM interface of the memory controller, allowing you to view efficiency statistics of the interface. You can access the efficiency statistics using the EMIF Debug Toolkit. (Identifier: DIAG_DDR3_EFFICIENCY_MONITOR)
Disable P2C Register Stage	Disable core register stages for signals entering the core fabric from the periphery. If the core register stages are disabled, latency is reduced but users must ensure that they do not connect the periphery directly to a DSP or a RAM block, without first registering the signals. (Identifier: DIAG_DDR3_DISABLE_AFI_P2C_REGISTERS)

Table 202. Group: Diagnostics / Miscellaneous

Display Name	Description
Use short Qsys interface names	Specifies the use of short interface names, for improved usability and consistency with other Qsys components. If this parameter is disabled, the names of Qsys interfaces exposed by the IP will include the type and direction of the interface. Long interface names are supported for backward-compatibility and will be removed in a future release. (Identifier: SHORT_QSYS_INTERFACE_NAMES)
Export PLL lock signal	Specifies whether to export the pll_locked signal at the IP top-level to indicate status of PLL. (Identifier: DIAG_EXPORT_PLL_LOCKED)

6.1.9. Intel Stratix 10 EMIF IP DDR3 Parameters: Example Designs

Table 203. Group: Example Designs / Available Example Designs

Display Name	Description
Select design	Specifies the *creation of a full Quartus Prime project, instantiating an external memory interface and an example traffic generator, according to your parameterization*. After the design is created, you can specify the target device and pin location assignments, run a full compilation, verify timing closure, and test the interface on your board using the programming file created by the Quartus Prime assembler. The '**Generate Example Design**' button lets you generate simulation or synthesis file sets. (Identifier: EX_DESIGN_GUI_DDR3_SEL_DESIGN)

Table 204. Group: Example Designs / Example Design Files

Display Name	Description
Simulation	Specifies that the '**Generate Example Design**' button create all necessary file sets for simulation. Expect a short additional delay as the file set is created. *If you do not enable this parameter, simulation file sets are not created.* Instead, the output directory will contain the `ed_sim.qsys` file which holds Qsys details of the simulation example design, and a `make_sim_design.tcl` file with other corresponding `tcl` files. You can run `make_sim_design.tcl` from a command line to generate the simulation example design. The generated example designs for various simulators are **stored in the /sim sub-directory**. (Identifier: EX_DESIGN_GUI_DDR3_GEN_SIM)
Synthesis	Specifies that the '**Generate Example Design**' button create all necessary file sets for synthesis. Expect a short additional delay as the file set is created. *If you do not enable this parameter, synthesis file sets are not created.* Instead, the output directory will contain the `ed_synth.qsys` file which holds Qsys details of the synthesis example design, and a

continued...

Display Name	Description
	`make_qii_design.tcl` script with other corresponding `tcl` files. You can run `make_qii_design.tcl` from a command line to generate the synthesis example design. The generated example design is **stored in the /qii sub-directory**. (Identifier: EX_DESIGN_GUI_DDR3_GEN_SYNTH)

Table 205. Group: Example Designs / Generated HDL Format

Display Name	Description
Simulation HDL format	This option lets you choose the format of HDL in which generated simulation files are created. (Identifier: EX_DESIGN_GUI_DDR3_HDL_FORMAT)

Table 206. Group: Example Designs / Target Development Kit

Display Name	Description
Select board	Specifies that *when you select a development kit with a memory module, the generated example design contains all settings and fixed pin assignments to run on the selected board. You must select a development kit preset to generate a working example design for the specified development kit*. Any IP settings not applied directly from a development kit preset will not have guaranteed results when testing the development kit. To exclude hardware support of the example design, select '**none**' from the '**Select board**' pull down menu. When you apply a development kit preset, all IP parameters are automatically set appropriately to match the selected preset. If you want to save your current settings, you should do so before you apply the preset. You can save your settings under a different name using **File->Save as**. (Identifier: EX_DESIGN_GUI_DDR3_TARGET_DEV_KIT)
PARAM_EX_DESIGN_PREV_PRESET_NAME	PARAM_EX_DESIGN_PREV_PRESET_DESC (Identifier: EX_DESIGN_GUI_DDR3_PREV_PRESET)

6.2. Register Map IP-XACT Support for Intel Stratix 10 EMIF DDR3 IP

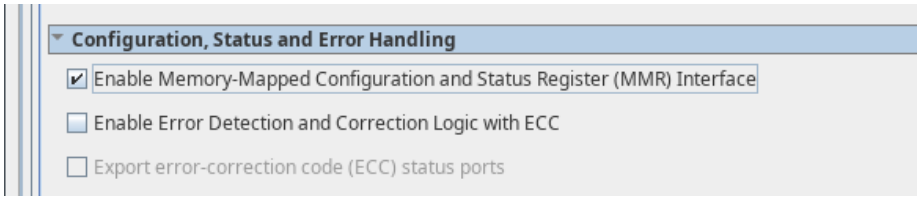
IP-XACT is an XML format that describes reusable intellectual property (IP).

When you generate an EMIF DDR3 design example from the Intel Quartus Prime software version 21.3 or later, the generated .ip file includes IP-XACT information for that IP. The generated IP-XACT information includes the register map for the DDR3 IP, Traffic Generator 2.0 (TG2), and Efficiency Monitor. The IP-XACT information for Intel Stratix 10 EMIF IP Memory-Mapped Registers (MMR) and Efficiency Monitor is included in `ed_synth_emif_fm_0.ip`, and the IP-XACT information for Traffic Generator 2.0 is included in `ed_synth_tg.ip`.

IP-XACT information is generated only with the design example. To enable generation of the IP-XACT information, follow these steps:

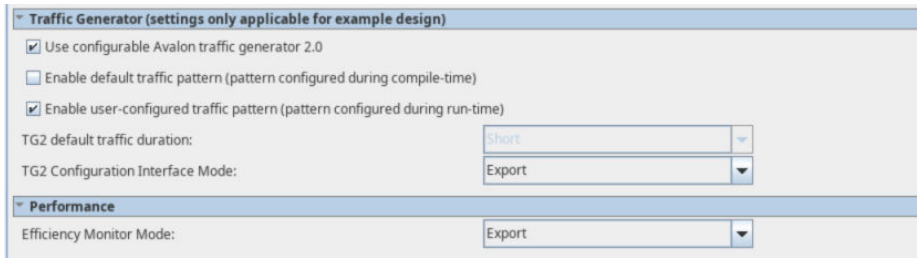
1. To enable generation of the IP-XACT information for Intel Stratix 10 IP MMR, check the **Enable Memory-Mapped Configuration and Status Register (MMR) Interface** box on the **Controller** tab of the parameter editor.

Figure 43. Enabling IP-XACT Generation for MMR Registers



2. To enable generation of IP-XACT information for TG2, check the **Use configurable Avalon traffic generator 2.0** box and set **TG2 Configuration Interface Mode** to *Export* on the **Diagnostics** tab of the parameter editor. To include IP-XACT information for the Efficiency Monitor, set the **Efficiency Monitor Mode** to *Export*.

Figure 44. Enabling IP-XACT Generation for TG2 and Efficiency Monitor



For information on the registers available for the Intel Stratix 10 EMIF IP, refer to *Intel Stratix 10 EMIF IP Memory Mapped Register (MMR) Tables* in the *End-User Signals* chapter.

For information on the registers available for Traffic Generator 2.0, refer to *Configuration and Status Registers* in the *Debugging* chapter.

For information on the registers available for the Efficiency Monitor, refer to *Control and Status Registers* in the *Debugging* chapter.

6.3. Board Skew Equations

The following table presents the underlying equations for the board skew parameters.

6.3.1. Equations for DDR3 Board Skew Parameters

Table 207. Board Skew Parameter Equations

Parameter	Description/Equation
Maximum CK delay to DIMM/device	The delay of the longest CK trace from the FPGA to any DIMM/device. $\max_r[\max_n(CK_{n\_r}PathDelay)]$

continued...

Parameter	Description/Equation
	Where *n* is the number of memory clock and r is the number rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clocks in each rank DIMM, the maximum CK delay is expressed by the following equation: $\max(CK_1PathDelayrank1,\ CK_2PathDelayrank1, CK_1PathDelayrank2,\ CK_2PathDelayrank2)$
Maximum DQS delay to DIMM/device	The delay of the longest DQS trace from the FPGA to the DIMM/device. $\max_r\left[\max_n\left(DQS_{n\_r}PathDelay\right)\right]$ Where *n* is the number of DQS and r is the number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 DQS in each rank DIMM, the maximum DQS delay is expressed by the following equation: $\max(DQS_1PathDelayrank1,\ DQS_2PathDelayrank1,\ DQS_1PathDelayrank2,\ DQS_2PathDelayrank2)$
Average delay difference between DQS and CK	The average delay difference between the DQS signals and the CK signal, calculated by averaging the longest and smallest DQS delay minus the CK delay. Positive values represent DQS signals that are longer than CK signals and negative values represent DQS signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the DQS signals for appropriate setup and hold margins. $\max_r\left[\max_{n,\ m}\right]\frac{\left\{\left(DQS_{m\_r}Delay - CK_{n\_r}Delay\right)\right\} + \min_r\left[\min_{n,\ m}\right]\left\{\left(DQS_{m\_r}Delay - CK_{n\_r}Delay\right)\right\}}{2}$ Where *n* is the number of memory clock, *m* is the number of DQS, and *r* is the number of rank of DIMM/device. When using discrete components, the calculation differs slightly. Find the minimum and maximum values for (DQS-CK) over all groups and then divide by 2. Calculate the (DQS-CK) for each DQS group, by using the appropriate CLK for that group. For example, in a configuration with 5 x16 components, with each component having two DQS groups: To find the minimum and maximum, calculate the minimum and maximum of (DQS0 – CK0, DQS1 – CK0, DQS2 –CK1, DQS3 – CK1, and so forth) and then divide the result by 2.
Maximum Board skew within DQS group	The largest skew between all DQ and DM pins in a DQS group. Enter your board skew only. Package skew is calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins. $\left[\max_{\substack{groups\\g}}\left[\max DQ_g - \min DQ_g\right]\right]$
Maximum skew between DQS groups	The largest skew between DQS signals in different DQS groups. $\left[\max_{\substack{groups\\g}}\left[DQS_g\right]\right] - \left[\min_{\substack{groups\\g}}\left[DQS_g\right]\right]$
Maximum system skew within address/command bus	$(MaxAC - MinAC)$ The largest skew between the address and command signals. Enter combined board and package skew. In the case of a component, find the maximum address/command and minimum address/command values across all component address signals.
Average delay difference between address/command and CK	A value equal to the average of the longest and smallest address/command signal delays, minus the delay of the CK signal. The value can be positive or negative. The average delay difference between the address/command and CK is expressed by the following equation: $\frac{\sum_{n=1}^{n=n}\left[\left(\frac{LongestACPathDelay + ShortestACPathDelay}{2}\right) - CK_nPathDelay\right]}{n}$ where *n* is the number of memory clocks.
Maximum delay difference between DIMMs/devices	The largest propagation delay on DQ signals between ranks. For example, in a two-rank configuration where you place DIMMs in different slots there is also a propagation delay for DQ signals going to and coming back from the furthest DIMM compared to the nearest

continued...

Parameter	Description/Equation
	DIMM. This parameter is applicable only when there is more than one rank.`Maxr { max n,m [(DQn_r path delay– DQn_r+1 path delay), (DQSm_r path delay–DQSm_r+1 path delay)]}` Where *n* is the number of DQ, *m* is the number of DQS and *r* is number of rank of DIMM/device .

6.4. Pin and Resource Planning

The following topics provide guidelines on pin placement for external memory interfaces.

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

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

Once all the requirements are known for your external memory interface, you can begin planning your system.

6.4.1. Interface Pins

DQS (data strobe or data clock) and DQ (data) pins are listed for EMIF supported banks in the device pin tables and are fixed at specific locations in the device. You must adhere to these pin locations to optimize routing, minimize skew, and maximize margins. Always check the device pin table for the actual locations of the DQS and DQ pins, and the EMIF pin table for location of address and control pins.

Pin tables are available here: https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html?1.

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

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

- Maximum possible interface width in any particular device is limited by the number of DQS groups available.
- Sufficient clock networks are available to the interface PLL as required by the IP.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other address and command, and clock pin placement requirements.

Note: The greater the number of banks, the greater the skew, hence Intel recommends that you always generate a test project of your desired configuration and confirm that it meets timing.

6.4.1.1. Estimating Pin Requirements

You should use the Intel Quartus Prime software for final pin fitting. However, you can estimate whether you have enough pins for your memory interface using the EMIF Device Selector on www.altera.com, or perform the following steps:

1. Determine how many read/write data pins are associated per data strobe or clock pair.
2. Calculate the number of other memory interface pins needed, including any other clocks (write clock or memory system clock), address, command, and RZQ. Refer to the External Memory Interface Pin Table to determine necessary Address/Command/Clock pins based on your desired configuration.
3. Calculate the total number of I/O banks required to implement the memory interface, given that an I/O bank supports up to 48 GPIO pins.

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

Related Information

Intel FPGA IP for External Memory Interfaces - Support Center

6.4.1.2. DIMM Options

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

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

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

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

Table 208. UDIMM and RDIMM Pin Options for DDR3

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)
Data	72 bit DQ[71:0] =	72 bit DQ[71:0] =	72 bit DQ[71:0] =	72 bit DQ[71:0]=

continued...

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)
	{CB[7:0], DQ[63:0]}	{CB[7:0], DQ[63:0]}	{CB[7:0], DQ[63:0]}	{CB[7:0], DQ[63:0]}
Data Mask	DM[8:0]	DM[8:0]	DM[8:0]	DM[8:0]
Data Strobe	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]	DQS[8:0] and DQS#[8:0]
Address	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]	BA[2:0], A[15:0]– 2 GB: A[13:0] 4 GB: A[14:0] 8 GB: A[15:0]
Clock	CK0/CK0#	CK0/CK0#, CK1/CK1#	CK0/CK0#	CK0/CK0#
Command	ODT, CS#, CKE, RAS#, CAS#, WE#	ODT[1:0], CS#[1:0], CKE[1:0], RAS#, CAS#, WE#	ODT, CS#[1:0], CKE, RAS#, CAS#, WE# [2]	ODT[1:0], CS#[1:0], CKE[1:0], RAS#, CAS#, WE#
Parity	—	—	PAR, ALERT	PAR, ALERT
Other Pins	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#

6.4.1.3. Maximum Number of Interfaces

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

Unless otherwise noted, the calculation for the maximum number of interfaces is based on independent interfaces where the address or command pins are not shared.

Note: You may need to share PLL clock outputs depending on your clock network usage.

For interface information for Intel Stratix 10, consult the EMIF Device Selector on www.altera.com.

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

Related Information

- Intel FPGA IP for External Memory Interfaces - Support Center
- Intel Stratix 10 EMIF Architecture: PLL Reference Clock Networks on page 21
- External Memory Interface Device Selector
- Intel Quartus Prime Pro Edition Handbook

6.4.2. FPGA Resources

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

6.4.2.1. OCT

You require one OCT calibration block if you are using an FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design. You can select any available OCT calibration block—it need not be within the same bank or side of the device as the memory interface pins. The only requirement is that the I/O bank where you place the OCT calibration block must use the same V_{CCIO} voltage as the memory interface.

The OCT calibration block uses a single R_{ZQ} pin. The R_{ZQ} pin in Intel Stratix 10 devices can be used as a general purpose I/O pin when it is not used to support OCT, provided the signal conforms to the bank voltage requirements.

6.4.2.2. PLL

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

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

6.4.3. Pin Guidelines for Intel Stratix 10 EMIF IP

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

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

- The numeric component of the `Bank Number` value identifies the I/O column, while the letter represents the I/O bank.
- The `Index within I/O Bank` value falls within one of the following ranges: 0 to 11, 12 to 23, 24 to 35, or 36 to 47, and represents I/O lanes 1, 2, 3, and 4, respectively.
- To determine if I/O banks are adjacent, you can refer to the I/O Pin Counts tables located in the *Intel Stratix 10 General Purpose I/O User Guide*. You can always assume I/O banks are adjacent within an I/O column except in the following conditions:
  - When an I/O bank is not bonded out on the package (contains the '-' symbol in the I/O table).
  - An I/O bank does not contain 48 pins, indicating it is only partially bonded out.
- The pairing pin for an I/O pin is located in the same I/O bank. You can identify the pairing pin by adding one to its `Index within I/O Bank` number (if it is an even number), or by subtracting one from its `Index within I/O Bank` number (if it is an odd number).

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

6.4.3.1. General Guidelines

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

If you are using the hard memory controller, you should employ the relative pin locations defined in the *<variation_name>*`/altera_emif_arch_nd_`*version number*`/`*<synth|sim>*`/`*<variation_name>*`_altera_emif_arch_nd_`*version number*`_`*<unique ID>*`_readme.txt` file, which is generated with your IP.

Note:

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

Observe the following general guidelines when placing pins for your Intel Stratix 10 external memory interface:

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

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

   Note: The pin-out scheme is a hardware requirement that you must follow, and can vary according to the topology of the memory device. Some schemes require three lanes to implement address and command pins, while others require four lanes. To determine which scheme to follow, refer to the messages window during parameterization of your IP, or to the `<variation_name>/altera_emif_arch_nd_<version>/<synth|sim>/<variation_name>_altera_emif_arch_nd_<version>_<unique ID>_readme.txt` file after you have generated your IP.
7. An unused I/O lane in the address and command bank can serve to implement a data group, such as a x8 DQS group. The data group must be from the same controller as the address and command signals.
8. An I/O lane must not be used by both address and command pins and data pins.
9. Place read data groups according to the DQS grouping in the pin table and Pin Planner. Read data strobes (such as DQS and DQS#) or read clocks (such as CQ and CQ# / QK and QK#) must reside at physical pins capable of functioning as DQS/CQ and DQSn/CQn for a specific read data group size. You must place the associated read data pins (such as DQ and Q), within the same group.

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

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

10. You can implement two x4 DQS groups with a single I/O lane. The pin table specifies which pins within an I/O lane can be used for the two pairs of DQS and DQS# signals. In addition, for x4 DQS groups you must observe the following rules:
    - There must be an even number of x4 groups in an external memory interface.
    - DQS group 0 and DQS group 1 must be placed in the same I/O lane. Similarly, DQS group 2 and group 3 must be in the same I/O lane. Generally, DQS group *X* and DQS group *X*+1 must be in the same I/O lane, where *X* is an even number.
    - When placing DQ pins in x4 mode, it is important to stay within an I/O lane when swapping pin locations. In other words, you may swap DQ pins within a given DQS group or across an adjacent DQS group, so long as you are within the same I/O lane. The following table illustrates an example, where DATA_A and DATA_B are swap groups, meaning that any pin in that index can move within that range of pins.

Index Within Lane	DQS x4 Locations
11	DATA_B[3:0]
10	DATA_B[3:0]
9	DQS_Bn
8	DQS_Bp
7	DATA_B[3:0]
6	DATA_B[3:0]
5	DQS_An
4	DQS_Ap
3	DATA_A[3:0]
2	DATA_A[3:0]
1	DATA_A[3:0]
0	DATA_A[3:0]

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

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

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

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

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

14. Be aware that for DDR4 interfaces clocked at 1333 MHz, total I/O bank usage is limited as follows:

Package	Total I/O 48 banks	Maximum number of I/O 48 banks that can be used for 1333 MHz	Remaining I/O 48 bank usage for EMIF or general-purpose I/O
1760	14	12	Do not use.
2397B	14	12	Do not use.
2912E	24	20	Do not use.

Note:

1. x4 mode does not support DM/DBI, or Intel Stratix 10 EMIF IP for HPS.
2. If you are using an Intel Stratix 10 EMIF IP-based RLDRAM 3 external memory interface, you should ensure that all the pins in a DQS group (that is, DQ, DM, DK, and QK) are placed in the same I/O bank. This requirement facilitates timing closure and is necessary for successful compilation of your design.

I/O Banks Selection

- For each memory interface, select adjacent I/O banks. To determine whether I/O banks are adjacent, refer to the I/O Pin Counts tables located in the *Intel Stratix 10 General Purpose I/O User Guide*. You can always assume I/O banks are adjacent within an I/O column except in the following conditions:
  - When an I/O bank is not bonded out on the package (contains the '-' symbol in the I/O table).
  - An I/O bank does not contain 48 pins, indicating that it is only partially bonded out.
- A memory interface can only span across I/O banks in the same I/O column.
- The number of I/O banks that you require depends on the memory interface width.
- In some device packages, the number of I/O pins in some LVDS I/O banks is less than 48 pins.

Address/Command Pins Location

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

CK Pins Assignment

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

- If the number of I/O banks is odd, assign one CK pin to the middle I/O bank.
- If the number of I/O banks is even, assign the CK pin to either of the middle two I/O banks.

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

PLL Reference Clock Pin Placement

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

- If you are sharing the PLL reference clock pin between several interfaces, the I/O banks must be adjacent. (That is, the banks must contain the same column number and letter before or after the respective I/O bank letter.)

The Intel Stratix 10 external memory interface IP does not support PLL cascading.

RZQ Pin Placement

You may place the R_{ZQ} pin in any I/O bank in an I/O column with the correct V_{CCIO} and V_{CCPT} for the memory interface I/O standard in use. However, the recommended location is in the address/command I/O bank, for greater flexibility during debug if a narrower interface project is required for testing.

DQ and DQS Pins Assignment

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

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

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

Sharing an I/O Bank Across Multiple Interfaces

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

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

6.4.3.2. x4 DIMM Implementation

DIMMS using a x4 DQS configuration require remapping of the DQS signals to achieve compatibility between the EMIF IP and the JEDEC standard DIMM socket connections.

The necessary remapping is shown in the table below. You can implement this DQS remapping in either RTL logic or in your schematic wiring connections.

Table 209. Mapping of DQS Signals Between DIMM and the EMIF IP

DIMM			Intel Quartus Prime EMIF IP	
DQS0	DQ[3:0]		DQS0	DQ[3:0]
DQS9	DQ[7:4]		DQS1	DQ[7:4]
DQS1	DQ[11:8]		DQS2	DQ[11:8]
DQS10	DQ[15:12]		DQS3	DQ[15:12]
DQS2	DQ[19:16]		DQS4	DQ[19:16]
DQS11	DQ[23:20]		DQS5	DQ[23:20]
DQS3	DQ[27:24]		DQS6	DQ[27:24]
DQS12	DQ[31:28]		DQS7	DQ[31:28]
DQS4	DQ[35:32]		DQS8	DQ[35:32]
DQS13	DQ[39:36]		DQS9	DQ[39:36]
DQS5	DQ[43:40]		DQS10	DQ[43:40]
DQS14	DQ[47:44]		DQS11	DQ[47:44]
DQS6	DQ[51:48]		DQS12	DQ[51:48]
DQS15	DQ[55:52]		DQS13	DQ[55:52]

continued...

DIMM			Intel Quartus Prime EMIF IP	
DQS7	DQ[59:56]		DQS14	DQ[59:56]
DQS16	DQ[63:60]		DQS15	DQ[63:60]
DQS8	DQ[67:64]		DQS16	DQ[67:64]
DQS17	DQ[71:68]		DQS17	DQ[71:68]

Data Bus Connection Mapping Flow

1. Connect all FPGA DQ pins accordingly to DIMM DQ pins. No remapping is required.
2. DQS/DQS*n* remapping is required either on the board schematics or in the RTL code.
3. An example mapping is shown below, with reference to the above table values:

```
FPGA (DQS0) to DIMM (DQS0)
FPGA (DQS1) to DIMM (DQS9)
FPGA (DQS2) to DIMM (DQS1)
...
FPGA (DQS16) to DIMM (DQS8)
FPGA (DQS17) to DIMM (DQS17)
```

When designing a board to support x4 DQS groups, Intel recommends that you make it compatible for x8 mode, for the following reasons:

- Provides the flexibility of x4 and x8 DIMM support.
- Allows use of x8 DQS group connectivity rules.
- Allows use of x8 timing rules for matching. Intel strongly recommends adhering to x4/x8 interoperability rules when designing a DIMM interface, even if the primary use case is to support x4 DIMMs only, because doing so facilitates debug and future migration capabilities. Regardless, the rules for length matching for two nibbles in a x4 interface must match those of the signals for a corresponding x8 interface, as the data terminations are turned on and off at the same time for both x4 DQS groups in an I/O lane. If the two x4 DQS groups were to have significantly different trace delays, it could adversely affect signal integrity.

About Pinout and Schematic Reviewing

When viewing x4 DQS mode in the Pin Planner, the 4 DQ pins do not have to be placed in the same colour-coded x4 group with the associated DQS/DQS*n* pins. This might look odd, but is not incorrect. The x4 DQS pins can be used as the strobe for any DQ pins placed within a x8 DQS group in an I/O lane.

Necessary checks to perform if the DQS groups are remapped in the RTL code

1. In the Pin Planner, view x8 DQS groups and check the following:
   a. Check that DQ[7:0] is in x8 group, DQ[15:8] is in another DQS group, and so forth.
   b. Check that DSQ0 and DQS9 are in the DQS group with DQ[7:0], DQS1 and DQS10 are in the DQS group with DQ[15:8], and so forth. This is the *DIMM* numbering convention column shown in the table at the beginning of this topic.
2. In the Pin Planner, view x4 DQS groups and check the following:

a. Check that all the DQS signals are on pins marked S and Sbar.

3. On the schematic, check the following DIMM connections:
   a. Check that DQS*x* on the DIMM maps to the DQS*x* on the FPGA pinout (for values of *x* from 0 to 17).
   b. Check that DQ*y* on the DIMM maps to the DQ*y* on the FPGA pinout. Note that there is scope for swapping pins within the x4/x8 DQS group to optimize the PCB layout.

Necessary checks to perform if the DQS groups are remapped on the schematic

1. In the Pin Planner, view x8 DQS groups and check the following:
   a. Check that DQ[7:0] is in x8 group, DQ[15:8] is in another DQS group, and so forth.
   b. Check that DSQ0 and DQS1 are in the DQS group with DQ[7:0], DQS2 and DQS3 are in the DQS group with DQ[15:8], and so forth. This is the *Intel Quartus Prime EMIF IP* mapping shown in the table at the beginning of this topic.
2. In the Pin Planner, view x4 DQS groups and check the following:
   a. Check that all the DQS signals are on pins marked S and Sbar.
3. On the schematic, check the following DIMM connections:
   a. Referring to the table above, check that DQS has the remapping between the FPGA (Intel Quartus Prime EMIF IP) and DIMM pinout (*DIMM*).
   b. Check that DQ*y* on the DIMM maps to the DQ*y* on the FPGA pinout. Note that there is scope for swapping pins within the x4/x8 DQS group to optimize the PCB layout.

6.4.3.3. Command and Address Signals

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

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

Although DDR4 operates in fundamentally the same way as other SDRAM, there are no longer dedicated pins for RAS#, CAS#, and WE#, as those are now shared with higher-order address pins. DDR4 still has CS#, CKE, ODT, and RESET# pins, similar to DDR3. DDR4 introduces some additional pins, including the ACT# (activate) pin and BG (bank group) pins. Depending on the memory format and the functions enabled, the following pins might also exist in DDR4: PAR (address command parity) pin and the ALERT# pin (1.2V I/O standard).

6.4.3.4. Clock Signals

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

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

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

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

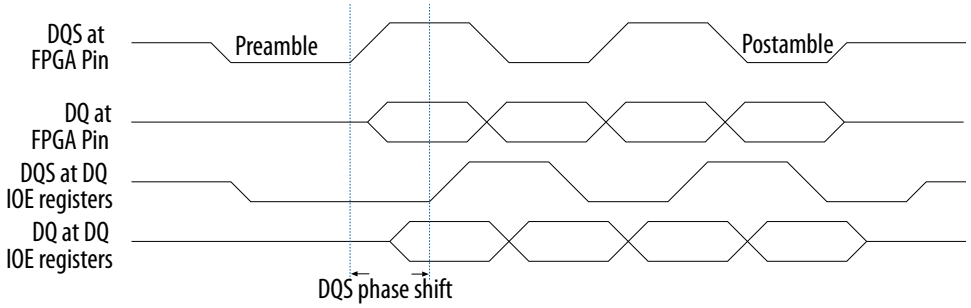
6.4.3.5. Data, Data Strobes, DM/DBI, and Optional ECC Signals

DDR3 and DDR4 SDRAM use bidirectional differential data strobes. Differential DQS operation enables improved system timing due to reduced crosstalk and less simultaneous switching noise on the strobe output drivers. The DQ pins are also bidirectional.

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

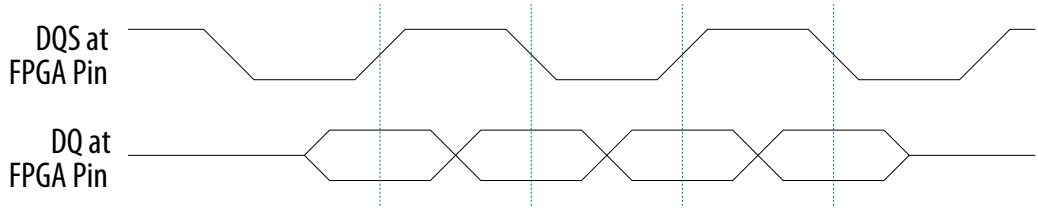
The DQ signals are edge-aligned with the DQS signal during a read from the memory and are center-aligned with the DQS signal during a write to the memory. The memory controller shifts the DQ signals by –90 degrees during a write operation to center align the DQ and DQS signals. The PHY IP delays the DQS signal during a read, so that the DQ and DQS signals are center aligned at the capture register. Intel devices use a phase-locked loop (PLL) to center-align the DQS signal with respect to the DQ signals during writes and Intel devices use dedicated DQS phase-shift circuitry to shift the incoming DQS signal during reads. The following figure shows an example where the DQS signal is shifted by 90 degrees for a read from the DDR3 SDRAM.

Figure 45. Edge-aligned DQ and DQS Relationship During a SDRAM Read in Burst-of-Four Mode



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

Figure 46. DQ and DQS Relationship During a SDRAM Write in Burst-of-Four Mode



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

The DQS signal is generated on the positive edge of the system clock to meet the t_{DQSS} requirement. DQ and DM signals use a clock shifted –90 degrees from the system clock, so that the DQS edges are centered on the DQ or DM signals when they arrive at the DDR3 SDRAM. The DQS, DQ, and DM board trace lengths need to be tightly matched (within 20 ps).

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

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

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

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

6.4.3.6. Resource Sharing Guidelines (Multiple Interfaces)

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

PLL Reference Clock Pin

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

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

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

Note: You can place the `pll_ref_clk` pin in the address and command I/O bank or in a data I/O bank, there is no impact on timing. However, for greatest flexibility during debug (such as when creating designs with narrower interfaces), the recommended placement is in the address and command I/O bank.

Core Clock Network

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

Observe the following guidelines for sharing the core clock network:

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

I/O Bank

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

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

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

Hard Nios Processor

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

6.4.3.7. Ping-Pong PHY Implementation

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

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

For pin allocation information for Intel Stratix 10 devices, refer to *External Memory Interface Pin Information for Intel Stratix 10 Devices* on www.altera.com.

Additional Requirements for DDR3 and DDR4 Ping-Pong PHY Interfaces

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

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

Related Information

- Pin-Out Files for Intel FPGA Devices
- Functional Description— Intel Stratix 10 EMIF IP

- External Memory Interface Pin Information for Intel Stratix 10 Devices
- Restrictions on I/O Bank Usage for Stratix 10 EMIF IP with HPS

6.5. DDR3 Board Design Guidelines

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

The following areas are discussed:

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

I/O Standards

DDR3 SDRAM interface signals use one of the following JEDEC* I/O signaling standards:

- SSTL-15—for DDR3.
- SSTL-135—for DDR3L.

Termination Schemes

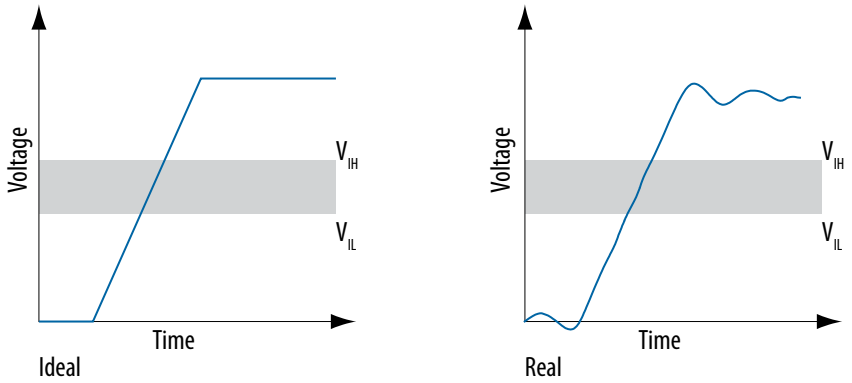
It is important to understand the trade-offs between different types of termination schemes, the effects of output drive strengths, and different loading types, so that you can swiftly navigate through the multiple combinations and choose the best possible settings for your designs.

The following key factors affect signal quality at the receiver:

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

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

Figure 47. Ideal and Real Signal at the Receiver



Related Information

JEDEC.org

6.5.1. Terminations and Slew Rates with Intel Stratix 10 Devices

The following topics describe termination and slew rate considerations for Intel Stratix 10 devices.

6.5.1.1. Dynamic On-Chip Termination (OCT) in Intel Stratix 10 Devices

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

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

Check the FPGA I/O tab of the parameter editor to determine the I/O standards and termination values supported for data, address and command, and memory clock signals.

Related Information

Choosing Terminations on Intel Stratix 10 Devices on page 178

6.5.1.2. Choosing Terminations on Intel Stratix 10 Devices

To determine optimal on-chip termination (OCT) and on-die termination (ODT) values for best signal integrity, you should simulate your memory interface in HyperLynx or a similar tool.

If the optimal OCT and ODT termination values as determined by simulation are not available in the list of available values in the parameter editor, select the closest available termination values for OCT and ODT.

For information about available ODT choices, refer to your memory vendor data sheet.

Related Information

Dynamic On-Chip Termination (OCT) in Intel Stratix 10 Devices on page 178

6.5.1.3. On-Chip Termination Recommendations for Intel Stratix 10 Devices

- A value of 34 to 40 ohms is a good starting point for output mode drive strength.
- Input mode (parallel termination) for Data and Data Strobe signals: A value of 40 or 60 ohms is a good starting point for FPGA side input termination.

6.5.1.4. Slew Rates

For optimum timing margins and best signal integrity for the address, command, and memory clock signals, you should generally use fast slew rates and external terminations.

In board simulation, fast slew rates may show a perceived signal integrity problem, such as reflections or a nonmonotonic waveform in the SSTL I/O switching region. Such indications may cause you to consider using slow slew rate options for either the address and command signals or the memory clock, or both.

If you set the **FPGA I/O tab parameter options ➤ Address/Command ➤ Slew Rate** and **Memory Clock ➤ Slew Rate** parameters to different values, a warning message appears: .

```
Warning: .emif_0: When the address/command signals and the memory clock signals
do not use the same slew rate setting, signals using the "Slow" setting are
delayed relative to signals using "Fast" setting. For accurate timing analysis,
you must perform I/O simulation and manually include the delay as board skew.
To avoid the issue, use the same slew rate setting for both address/command
signals and memory clock signals whenever possible.
```

Note: The warning message applies only to board-level simulation, and does not require any delay adjustments in the PCB design or Board tab parameter settings.

Due to limitations of the IBIS model correlation tolerance and the accuracy of the board simulation model, it is possible for signal integrity problems to appear when using fast slew rate during simulation but not occur during operation on hardware. If you observe a signal integrity problem during simulation with a fast slew rate, use an oscilloscope to view the signal at that point in hardware, to verify whether the problem exists on hardware, or only in simulation.

If the signal integrity problem exists on hardware as well as in simulation, using different slew rates for the address and command signals and the clock remains a valid approach, and the address and command calibration stage will help to improve the address and command to clock setup and hold time margins.

6.5.2. Channel Signal Integrity Measurement

As external memory interface data rates increase, so does the importance of proper channel signal integrity measurement.By measuring the actual channel loss during the layout process and including that data in your parameterization, a realistic assessment of margins is achieved.

6.5.2.1. Importance of Accurate Channel Signal Integrity Information

Default values for channel loss (or eye reduction) can be used when calculating timing margins, however those default values may not accurately reflect the channel loss in your system.If the channel loss in your system is different than the default values, the calculated timing margins vary accordingly.

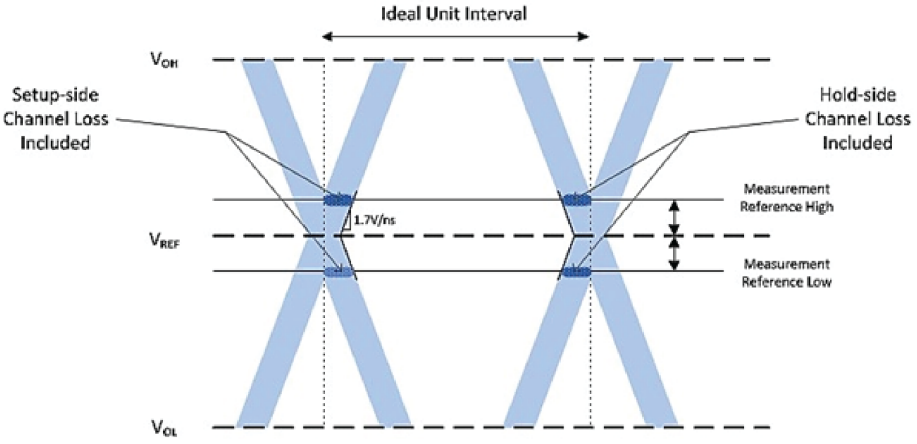
If your actual channel loss is greater than the default channel loss, and if you rely on default values, the available timing margins for the entire system are lower than the values calculated during compilation. By relying on default values that do not accurately reflect your system, you may be lead to believe that you have good timing margin, while in reality, your design may require changes to achieve good channel signal integrity.

6.5.2.2. Understanding Channel Signal Integrity Measurement

To measure channel signal integrity you need to measure the channel loss for various signals.For a particular signal or signal trace, channel loss is defined as loss of the eye width at +/- V_{IH}(ac and dc) +/- V_{IL}(ac and dc). V_{IH}/V_{IL} above or below V_{REF} is used to align with various requirements of the timing model for memory interfaces.

The example below shows a reference eye diagram where the channel loss on the setup- or leading-side of the eye is equal to the channel loss on the hold- or lagging-side of the eye; however, it does not necessarily have to be that way. Because the calibrating PHY calibrates to the center of the read and write eye, the Board Settings tab has parameters for the total extra channel loss for Write DQ and Read DQ. For address and command signals which are not-calibrated, the Board Settings tab allows you to enter setup- and hold-side channel losses that are not equal, allowing the Intel Quartus Prime software to place the clock statically within the center of the address and command eye.

Figure 48. Equal Setup and Hold-side Losses



6.5.2.3. How to Enter Calculated Channel Signal Integrity Values

You should enter calculated channel loss values in the **Channel Signal Integrity** section of the **Board** (or **Board Timing**) tab of the parameter editor.

For Intel Stratix 10 external memory interfaces, the default channel loss displayed in the parameter editor is based on the selected configuration (different values for single rank versus dual rank), and on internal Intel reference boards. You should replace the default value with the value that you calculate.

6.5.2.4. Guidelines for Calculating DDR3 Channel Signal Integrity

Address and Command ISI and Crosstalk

Simulate the address/command and control signals and capture eye at the DRAM pins, using the memory clock as the trigger for the memory interface's address/command and control signals. Measure the setup and hold channel losses at the voltage thresholds mentioned in the memory vendor's data sheet.

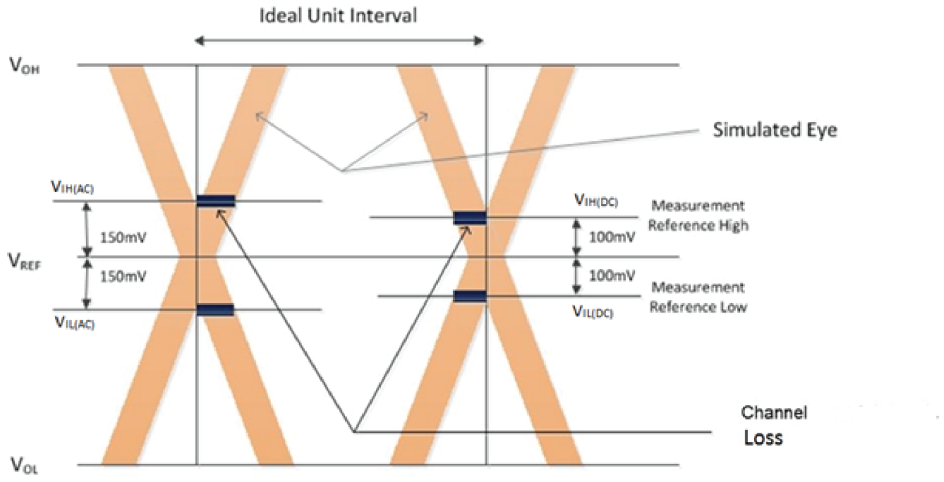
Address and command channel loss = Measured loss on the setup side + measured loss on the hold side.

$V_{REF} = V_{DD}/2 = 0.75$ V for DDR3

You should select the V_{IH} and V_{IL} voltage levels appropriately for the DDR3L memory device that you are using. Check with your memory vendor for the correct voltage levels, as the levels may vary for different speed grades of device.

The following figure illustrates a DDR3 example where $V_{IH(AC)}$/ $V_{IL(AC)}$ is +/- 150 mV and $V_{IH(DC)}$/ $V_{IL(DC)}$ is +/- 100 mV.

Figure 49.



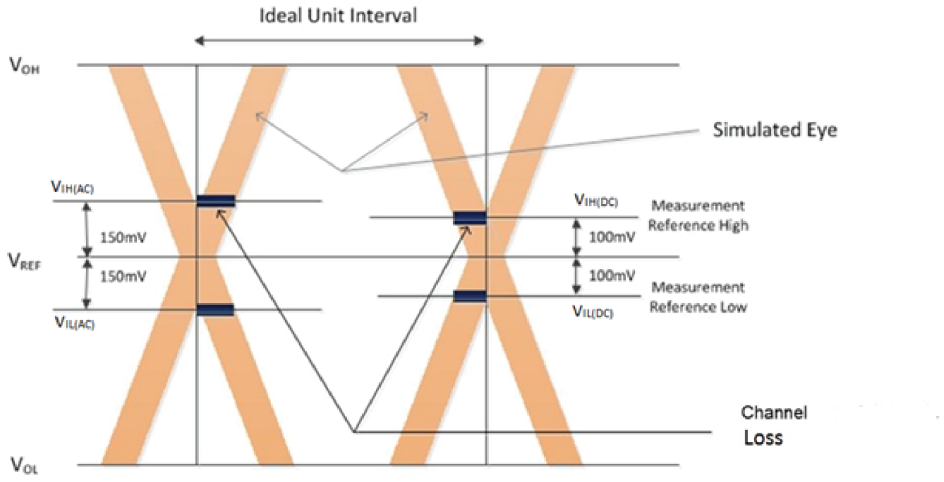
Write DQ ISI and Crosstalk

Simulate the write DQ signals and capture eye at the DRAM pins, using DQ Strobe (DQS) as a trigger for the DQ signals of the memory interface simulation. Measure the setup and hold channel losses at the V_{IH} and V_{IL} mentioned in the memory vendor's data sheet. The following figure illustrates a DDR3 example where $V_{IH(AC)}$/ $V_{IL(AC)}$ is +/- 150 mV and $V_{IH(DC)}$/ $V_{IL(DC)}$ is +/- 100 mV.

Write Channel Loss = Measured Loss on the Setup side + Measured Loss on the Hold side

$V_{REF} = V_{DD}/2 = 0.75V$ for DDR3

Figure 50.



Read DQ ISI and Crosstalk

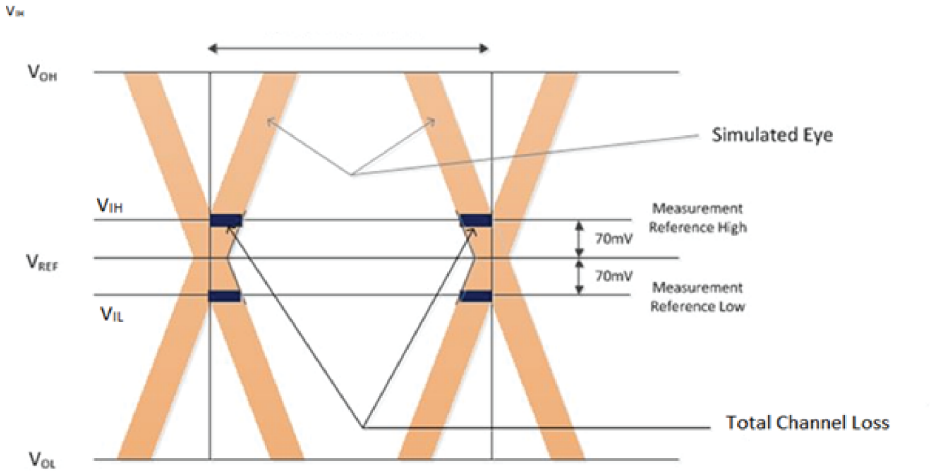
Simulate read DQ signals and capture eye at the FPGA die. Do not measure at the pin, because you might see unwanted reflections that could create a false representation of the eye opening at the input buffer of the FPGA. Use DQ Strobe (DQS) as a trigger for the DQ signals of your memory interface simulation. Measure the eye opening at +/- 70 mV (V_{IH}/V_{IL}) with respect to V_{REF}.

Read Channel Loss = (UI) - (Eye opening at +/- 70 mV with respect to V_{REF})

UI = Unit interval. For example, if you are running your interface at 800 Mhz, the effective data is 1600 Mbps, giving a unit interval of 1/1600 = 625 ps

V_{REF} = VDD/2 = 0.75 V for DDR3

Figure 51.

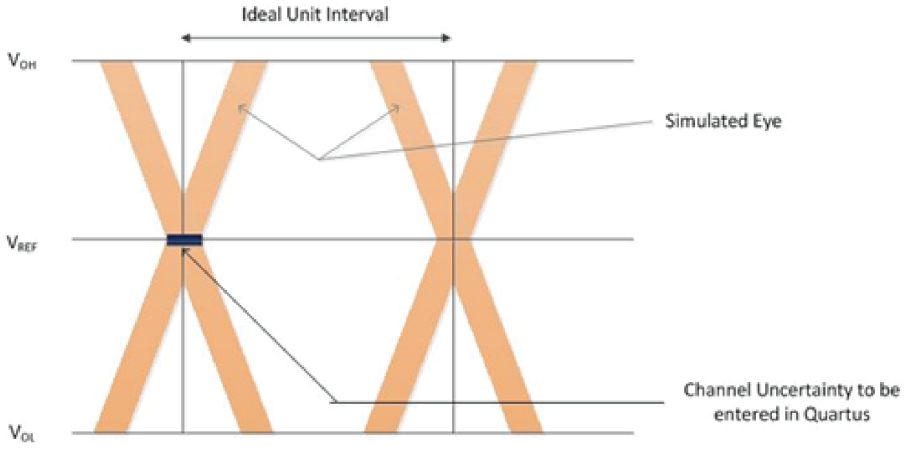


Write/Read DQS ISI and Crosstalk

Simulate the Write/Read DQS and capture eye, and measure the uncertainty at V_{REF}.

V_{REF} = VDD/2 = 0.75 V for DDR3

Figure 52.

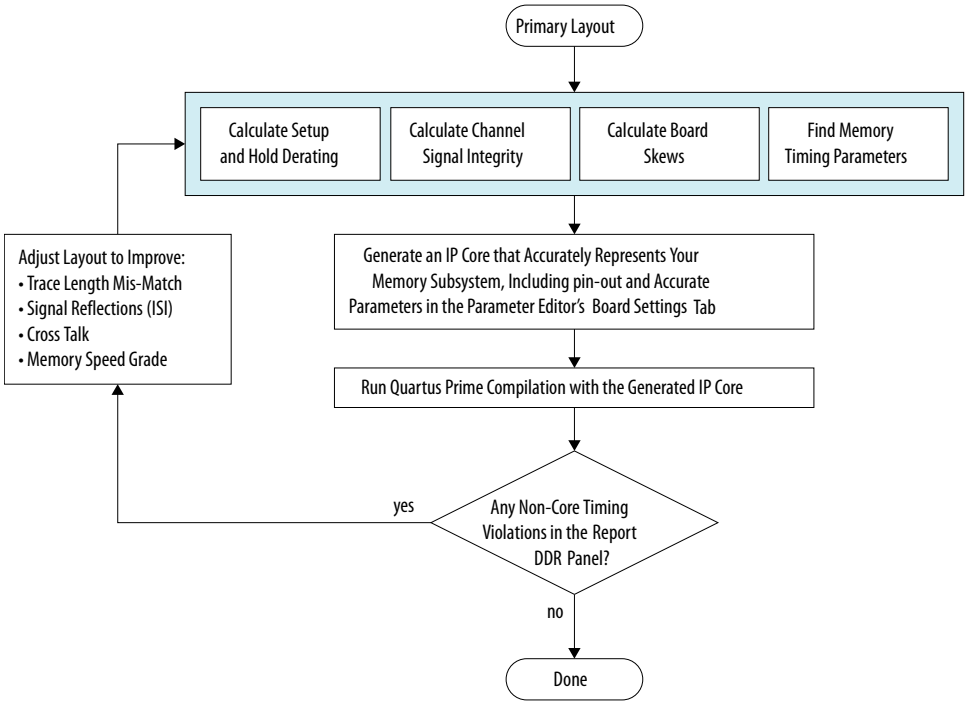


6.5.3. Layout Approach

For all practical purposes, you can regard the Timing Analyzer report on your memory interface as definitive for a given set of memory and board timing parameters.

You can find timing information under **Report DDR** in the Timing Analyzer and on the **Timing Analysis** tab in the parameter editor.

The following flowchart illustrates the recommended process to follow during the board design phase, to determine timing margin and make iterative improvements to your design.



Board Skew

For information on calculating board skew parameters, refer to *Board Skew Equations*, in this chapter.

The Board Skew Parameter Tool is an interactive tool that can help you calculate board skew parameters if you know the absolute delay values for all the memory related traces.

Memory Timing Parameters

For information on the memory timing parameters to be entered into the parameter editor, refer to the datasheet for your external memory device.

Related Information

Board Skew Parameter Tool

6.5.4. Design Layout Guidelines

The general layout guidelines in the following topic apply to DDR3 and DDR4 SDRAM interfaces.

These guidelines help you plan your board layout, but are not meant as strict rules that you must adhere to. Intel recommends that you perform your own board-level simulations to ensure that the layout you choose for your board allows you to achieve your desired performance.

For more information about how the memory manufacturers route these address and control signals on their DIMMs, refer to the Cadence PCB browser from the Cadence website, at www.cadence.com. You can find the various JEDEC example DIMM layouts on the JEDEC website, at www.jedec.org.

For assistance in calculating board skew parameters, refer to the board skew calculator tool, which you can find at the Intel website.

Note:

1. The following layout guidelines include several +/- length based rules. These length based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristic of the interface. They do not include any margin for crosstalk.
2. To ensure reliable timing closure to and from the periphery of the device, you should register signals to and from the periphery before you connect any further logic.

Intel recommends that you get accurate time base skew numbers for your design when you simulate the specific implementation.

Related Information

- JEDEC.org
- https://www.cadence.com/
- Board Skew Parameter Tool
- https://eda.sw.siemens.com/

6.5.4.1. General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Intel recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

Table 210. General Layout Guidelines

Parameter	Guidelines
Impedance	• All unused via pads must be removed, because they cause unwanted capacitance. • Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.
Decoupling Parameter	• Use 0.1 uF in 0402 size to minimize inductance • Make VTT voltage decoupling close to termination resistors • Connect decoupling caps between VTT and ground • Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin • Verify the capacitive decoupling using the Intel Power Distribution Network Design Tool
Power	• Route GND and V_{CC} as planes • Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation • Route VTT as islands or 250-mil (6.35-mm) power traces • Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces
General Routing	All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Intel recommends that signals from the same net group always be routed on the same layer. • Use 45° angles (*not* 90° corners) • Avoid T-Junctions for critical nets or clocks • Avoid T-junctions greater than 250 mils (6.35 mm) • Disallow signals across split planes • Restrict routing other signals close to system reset signals • Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks

Related Information

Power Distribution Network

6.5.4.2. Layout Guidelines

The following table lists layout guidelines.

Unless otherwise specified, the guidelines in the following table apply to the following topologies:

- DIMM—UDIMM topology
- DIMM—RDIMM topology
- DIMM—LRDIMM topology
- Not all versions of the Intel Quartus Prime software support LRDIMM.
- Discrete components laid out in UDIMM topology
- Discrete components laid out in RDIMM topology

These guidelines are recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface.

For supported frequencies and topologies, refer to the *External Memory Interface Spec Estimator* https://www.intel.com/content/www/us/en/programmable/support/support-resources/external-memory.html.

For frequencies greater than 800 MHz, when you are calculating the delay associated with a trace, you must take the FPGA package delays into consideration.

Table 211. Layout Guidelines *(1)*

Parameter	Guidelines
Decoupling Parameter	• Make VTT voltage decoupling close to the components and pull-up resistors. • Connect decoupling caps between VTT and VDD using a 0.1□F cap for every other VTT pin. • Use a 0.1 uF cap and 0.01 uF cap for every VDDQ pin.
Maximum Trace Length	• Even though there are no hard requirements for minimum trace length, you need to simulate the trace to ensure the signal integrity. Shorter routes result in better timing. • For DIMM topology only: — Maximum trace length for all signals from FPGA to the first DIMM slot is 4.5 inches. — Maximum trace length for all signals from DIMM slot to DIMM slot is 0.425 inches. • For discrete components only: — Maximum trace length for address, command, control, and clock from FPGA to the first component must not be more than 7 inches. — Maximum trace length for DQ, DQS, DQS#, and DM from FPGA to the first component is 5 inches.
General Routing	• Route over appropriate VCC and GND planes. • Keep signal routing layers close to GND and power planes.
Spacing Guidelines	• Avoid routing two signal layers next to each other. Always make sure that the signals related to memory interface are routed between appropriate GND or power layers. • For DQ/DQS/DM traces: Maintain at least 3H spacing between the edges (air-gap) for these traces. (Where H is the vertical distance to the closest return path for that particular trace.) • For Address/Command/Control traces: Maintain at least 3H spacing between the edges (air-gap) these traces. (Where H is the vertical distance to the closest return path for that particular trace.) • For Clock traces: Maintain at least 5H spacing between two clock pair or a clock pair and any other memory interface trace. (Where H is the vertical distance to the closest return path for that particular trace.)
Clock Routing	• Route clocks on inner layers with outer-layer run lengths held to under 500 mils (12.7 mm). • Route clock signals in a daisy chain topology from the first SDRAM to the last SDRAM. The maximum length of the first SDRAM to the last SDRAM must not exceed 0.69 tCK for DDR3 and 1.5 tCK for DDR4. For different DIMM configurations, check the appropriate JEDEC specification. • These signals should maintain the following spacings: • Clocks should maintain a length-matching between clock pairs of ±5 ps. • Clocks should maintain a length-matching between positive (p) and negative (n) signals of ±2 ps, routed in parallel. • Space between different pairs should be at least two times the trace width of the differential pair to minimize loss and maximize interconnect density. • To avoid mismatched transmission line to via, Intel recommends that you use Ground Signal Signal Ground (GSSG) topology for your clock pattern—GND|CLKP|CKLN|GND. • Route all addresses and commands to match the clock signals to within ±20 ps to each discrete memory component. Refer to the following figure.

continued...

Parameter	Guidelines
Address and Command Routing	• Route address and command signals in a daisy chain topology from the first SDRAM to the last SDRAM. The maximum length of the first SDRAM to the last SDRAM must not be more than 0.69 tCK for DDR3 and 1.5 tCK for DDR4. For different DIMM configurations, check the appropriate JEDEC specifications. • UDIMMs are more susceptible to cross-talk and are generally noisier than buffered DIMMs. Therefore, route address and command signals of UDIMMs on a different layer than data signals (DQ) and data mask signals (DM) and with greater spacing. • Do not route differential clock (CK) and clock enable (CKE) signals close to address signals. • Route all addresses and commands to match the clock signals to within ±20 ps to each discrete memory component. Refer to the following figure.
DQ, DM, and DQS Routing Rules	• All the trace length matching requirements are from the FPGA package ball to the SDRAM package ball, which means you must consider trace mismatching on different DIMM raw cards. • Match in length all DQ, DQS, and DM signals within a given byte-lane group with a maximum deviation of ±10 ps. • Ensure to route all DQ, DQS, and DM signals within a given byte-lane group on the same layer to avoid layer to layer transmission velocity differences, which otherwise increase the skew within the group. • Do not count on FPGAs to deskew for more than 20 ps of DQ group skew. The skew algorithm only removes the following possible uncertainties: — Minimum and maximum die IOE skew or delay mismatch — Minimum and maximum device package skew or mismatch — Board delay mismatch of 20 ps — Memory component DQ skew mismatch — Increasing any of these four parameters runs the risk of the deskew algorithm limiting, failing to correct for the total observed system skew. If the algorithm cannot compensate without limiting the correction, timing analysis shows reduced margins. • For memory interfaces with leveling, the timing between the DQS and clock signals on each device calibrates dynamically to meet tDQSS. To make sure the skew is not too large for the leveling circuit's capability, follow these rules: — Propagation delay of clock signal must not be shorter than propagation delay of DQS signal at every device: (CKi) – DQSi > 0; 0 < *i* < number of components – 1 . For DIMMs, ensure that the CK trace is longer than the longest DQS trace at the DIMM connector. — Total skew of CLK and DQS signal between groups is less than one clock cycle: (CKi+ DQSi) max – (CKi+ DQSi) min < 1 × tCK(If you are using a DIMM topology, your delay and skew must take into consideration values for the actual DIMM.)

continued...

Parameter	Guidelines
Spacing Guidelines	• Avoid routing two signal layers next to each other. Always ensure that the signals related to the memory interface are routed between appropriate GND or power layers. • For DQ/DQS/DM traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace. • For Address/Command/Control traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace. • For Clock traces: Maintain at least 5H spacing between two clock pairs or a clock pair and any other memory interface trace, where H is the vertical distance to the closest return path for that particular trace.
Intel Quartus Prime Software Settings for Board Layout	• To perform timing analyses on board and I/O buffers, use a third-party simulation tool to simulate all timing information such as skew, ISI, crosstalk, and type the simulation result into the Board Settings tab in the parameter editor. • Do not use advanced I/O timing model (AIOT) or board trace model unless you do not have access to any third party tool. AIOT provides reasonable accuracy but tools like HyperLynx provide better results.
Notes to Table: 1. For point-to-point and DIMM interface designs, refer to the Micron website, www.micron.com.	

Related Information

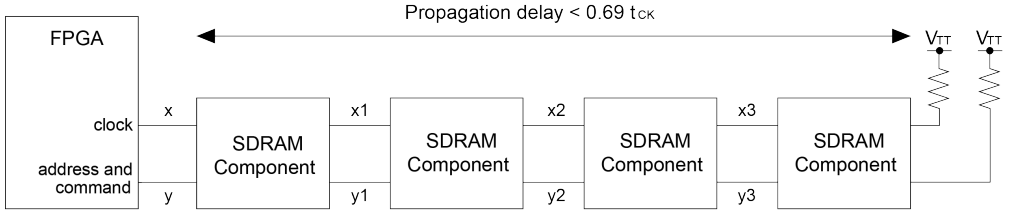
Package Deskew on page 193

6.5.4.3. Length Matching Rules

The following topics provide guidance on length matching for different types of SDRAM signals.

Route all addresses and commands to match the clock signals to within ±20 ps to each discrete memory component. The following figure shows the component routing guidelines for address and command signals.

Figure 53. SDRAM Component Address and Command Routing Guidelines



If using discrete components:
x = y ± 20 ps
x + x1 = y + y1 ± 20 ps
x + x1 + x2 = y + y1 + y2 ± 20 ps
x + x1 + x2 + x3 = y + y1 + y2 +y3 ± 20 ps

If using a DIMM topology:
x=y +/- 20 ps

The `alert_n` signal requires an external pull-up resistor to 1.2V using a typical pull-up resistor value of 10,000 ohms.

The timing between the DQS and clock signals on each device calibrates dynamically to meet tDQSS. The following figure shows the delay requirements to align DQS and clock signals. To ensure that the skew is not too large for the leveling circuit's capability, follow these rules:

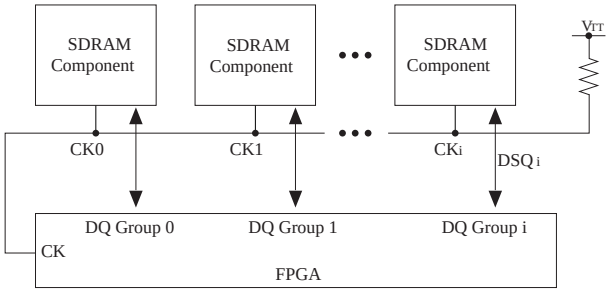
- Propagation delay of clock signal must not be shorter than propagation delay of DQS signal at every device:

```
CKi  – DQSi > 0; 0 < i < number of components – 1
```

- Total skew of CLK and DQS signal between groups is less than one clock cycle:

```
(CKi + DQSi) max – (CKi + DQSi) min < 1 × tCK
```

Figure 54. Delaying DQS Signal to Align DQS and Clock



CKi = Clock signal propagation delay to device i
DQSi = DQ/DQS signals propagation delay to group i

Clk pair matching—If you are using a DIMM (UDIMM, RDIMM, or LRDIMM) topology, match the trace lengths up to the DIMM connector. If you are using discrete components, match the lengths for all the memory components connected in the fly-by chain.

DQ group length matching—If you are using a DIMM (UDIMM, RDIMM, or LRDIMM) topology, apply the DQ group trace matching rules described in the guideline table earlier up to the DIMM connector. If you are using discrete components, match the lengths up to the respective memory components.

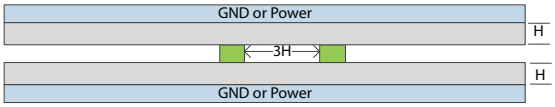
When you are using DIMMs, it is assumed that lengths are tightly matched within the DIMM itself. You should check that appropriate traces are length-matched within the DIMM.

6.5.4.4. Spacing Guidelines

This topic provides recommendations for minimum spacing between board traces for various signal traces.

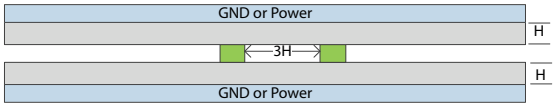
Spacing Guidelines for DQ, DQS, and DM Traces

Maintain a minimum of 3H spacing between the edges (air-gap) of these traces. (Where *H* is the vertical distance to the closest return path for that particular trace.)



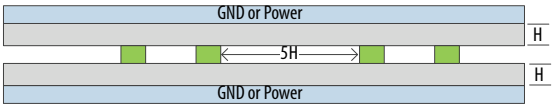
Spacing Guidelines for Address and Command and Control Traces

Maintain at least 3H spacing between the edges (air-gap) of these traces. (Where *H* is the vertical distance to the closest return path for that particular trace.)



Spacing Guidelines for Clock Traces

Maintain at least 5H spacing between two clock pair or a clock pair and any other memory interface trace. (Where *H* is the vertical distance to the closest return path for that particular trace.)



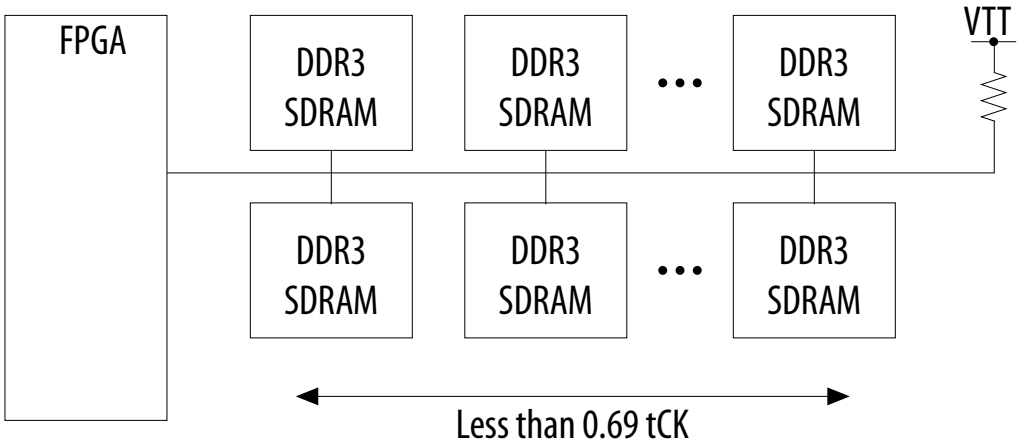
6.5.4.5. Fly-By Network Design for Clock, Command, and Address Signals

The EMIF IP requires the flight-time skew between the first SDRAM component and the last SDRAM component to be less than 0.69 tCK for memory clocks. This constraint limits the number of components you can have for each fly-by network.

If you design with discrete components, you can choose to use one or more fly-by networks for the clock, command, and address signals.

The following figure shows an example of a single fly-by network topology.

Figure 55. Single Fly-By Network Topology

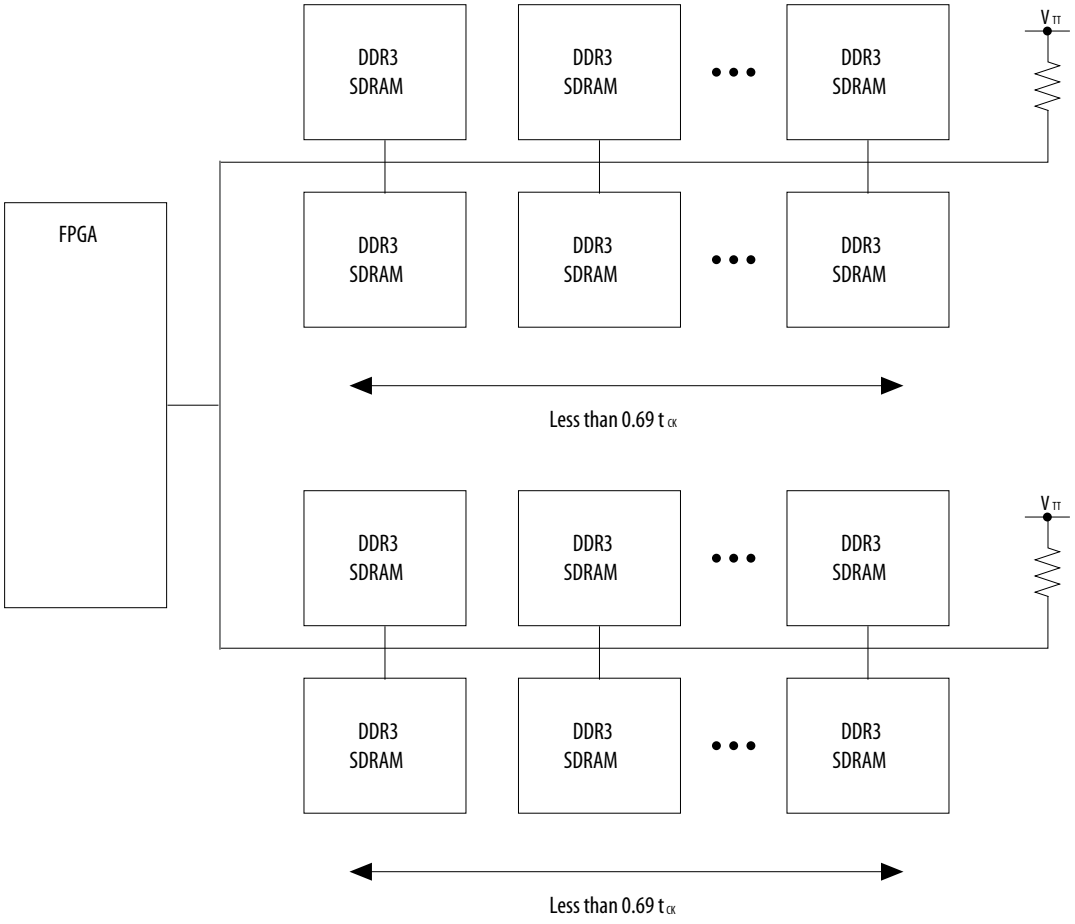


Every SDRAM component connected to the signal is a small load that causes discontinuity and degrades the signal. When using a single fly-by network topology, to minimize signal distortion, follow these guidelines:

- Use ×16 device instead ×4 or ×8 to minimize the number of devices connected to the trace.
- Keep the stubs as short as possible.
- Even with added loads from additional components, keep the total trace length short; keep the distance between the FPGA and the first SDRAM component less than 5 inches.
- Simulate clock signals to ensure a decent waveform.

The following figure shows an example of a double fly-by network topology. This topology is not rigid but you can use it as an alternative option. The advantage of using this topology is that you can have more SDRAM components in a system without violating the 0.69 tCK rule. However, as the signals branch out, the components still create discontinuity.

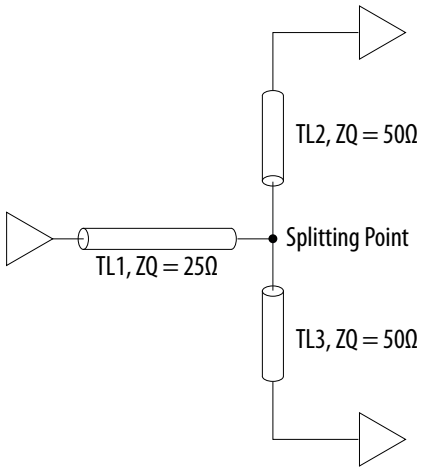
Figure 56. Double Fly-By Network Topology



You must perform simulations to find the location of the split, and the best impedance for the traces before and after the split.

The following figure shows a way to minimize the discontinuity effect. In this example, keep TL2 and TL3 matches in length. Keep TL1 longer than TL2 and TL3, so that it is easier to route all the signals during layout.

Figure 57. Minimizing Discontinuity Effect



You can also consider using a DIMM on each branch to replace the components. Because the trace impedance on the DIMM card is 40-ohm to 60-ohm, perform a board trace simulation to control the reflection to within the level your system can tolerate.

Using the fly-by daisy chain topology increases the complexity of the datapath and controller design to achieve leveling, but also greatly improves performance and eases board layout for SDRAM implementations.

You can also use the SDRAM components without leveling in a design if it may result in a more optimal solution, or use with devices that support the required electrical interface standard, but do not support the required read and write leveling functionality.

6.5.5. Package Deskew

Trace lengths inside the device package are not uniform for all package pins. The nonuniformity of package traces can affect system timing for high frequencies. A package deskew option is available in the Intel Quartus Prime software.

If you do not enable the package deskew option, the Intel Quartus Prime software uses the package delay numbers to adjust skews on the appropriate signals; you do not need to adjust for package delays on the board traces. If you do enable the package deskew option, the Intel Quartus Prime software does not use the package delay numbers for timing analysis, and you must deskew the package delays with the board traces for the appropriate signals for your design.

Related Information

Layout Guidelines on page 186

6.5.5.1. DQ/DQS/DM Deskew

To get the package delay information, follow these steps:

1. Select the **FPGA DQ/DQS Package Skews Deskewed on Board** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Compile your design.
5. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

6.5.5.2. Address and Command Deskew

Deskew address and command delays as follows:

1. Select the **FPGA Address/Command Package Skews Deskewed on Board** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Compile your design.
5. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

6.5.5.3. Package Deskew Recommendations for Intel Stratix 10 Devices

The following table shows package deskew recommendations for Intel Stratix 10 devices.

As operating frequencies increase, it becomes increasingly critical to perform package deskew. The frequencies listed in the table are the *minimum* frequencies for which you must perform package deskew.

If you plan to use a listed protocol at the specified frequency or higher, you must perform package deskew.

Protocol	Minimum Frequency (MHz) for Which to Perform Package Deskew		
	Single Rank	Dual Rank	Quad Rank
DDR4	933	800	667
DDR3	933	800	667
QDR IV	933	Not applicable	Not applicable
RLDRAM 3	933	667	Not applicable
QDR II, II+, II+ Xtreme	Not required	Not applicable	Not applicable

6.5.5.4. Deskew Example

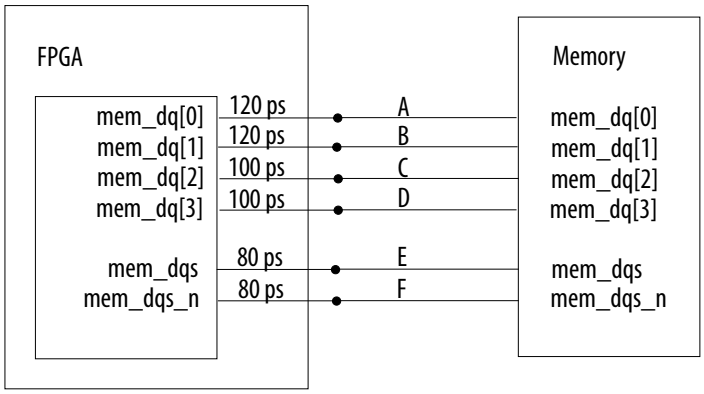
Consider an example where you want to deskew an interface with 4 DQ pins, 1 DQS pin, and 1 DQSn pin.

Let's assume an operating frequency of 667 MHz, and the package lengths for the pins reported in the **.pin** file as follows:

```
dq[0] = 120 ps
dq[1] = 120 ps
dq[2] = 100 ps
dq[3] = 100 ps
dqs   = 80 ps
dqs_n = 80 ps
```

The following figure illustrates this example.

Figure 58. Deskew Example

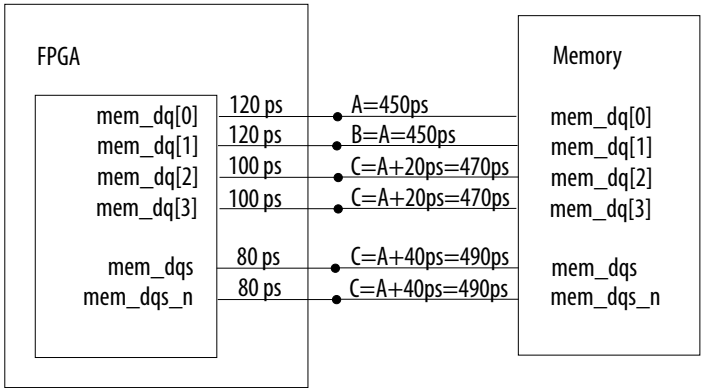


When you perform length matching for all the traces in the DQS group, you must take package delays into consideration. Because the package delays of traces A and B are 40 ps longer than the package delays of traces E and F, you would need to make the board traces for E and F 40 ps longer than the board traces for A and B.

A similar methodology would apply to traces C and D, which should be 20 ps longer than the lengths of traces A and B.

The following figure shows this scenario with the length of trace A at 450 ps.

Figure 59. Deskew Example with Trace Delay Calculations



When you enter the board skews into the Board Settings tab of the DDR3 parameter editor, you should calculate the board skew parameters as the sums of board delay and corresponding package delay. If a pin does not have a package delay (such as address and command pins), you should use the board delay only.

The example of the preceding figure shows an ideal case where board skews are perfectly matched. In reality, you should allow plus or minus 10 ps of skew mismatch within a DQS group (DQ/DQS/DM).

6.5.5.5. Package Migration

Package delays can be different for the same pin in different packages. If you want to use multiple migratable packages in your system, you should compensate for package skew as described in this topic.The information in this topic applies to Intel Stratix 10 devices.

Scenario 1

Your PCB is designed for multiple migratable devices, but you have only one device with which to go to production.

Assume two migratable packages, device A and device B, and that you want to go to production with device A. Follow these steps:

1. Perform package deskew for device A.
2. Compile your design for device A, with the **Package Skew** option enabled.
3. Note the skews in the `<core_name>.pin` file for device A. Deskew these package skews with board trace lengths as described in the preceding examples.
4. Recompile your design for device A.
5. For device B, open the parameter editor and deselect the **Package Deskew** option.
6. Calculate board skew parameters, only taking into account the board traces for device B, and enter that value into the parameter editor for device B.
7. Regenerate the IP and recompile the design for device B.
8. Verify that timing requirements are met for both device A and device B.

Scenario 2

Your PCB is designed for multiple migratable devices, and you want to go to production with all of them.

Assume you have device A and device B, and plan to use both devices in production. Follow these steps:

1. Do not perform any package deskew compensation for either device.
2. Compile a Quartus Prime design for device A with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
3. Verify that the **Report DDR** timing report meets your timing requirements.
4. Compile a Quartus Prime design for device B with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
5. Verify that the **Report DDR** timing report meets your timing requirements.

7. Intel Stratix 10 EMIF IP for DDR4

This chapter contains IP parameter descriptions, board skew equations, pin planning information, and board design guidance for Intel Stratix 10 external memory interfaces for DDR4.

7.1. Parameter Descriptions

The following topics describe the parameters available on each tab of the IP parameter editor, which you can use to configure your IP.

7.1.1. Intel Stratix 10 EMIF IP DDR4 Parameters: General

Table 212. Group: General / Interface

Display Name	Description
Configuration	Specifies the configuration of the memory interface. The available options depend on the protocol and the targeted FPGA product. (Identifier: PHY_DDR4_CONFIG_ENUM)
Instantiate two controllers sharing a Ping Pong PHY	Specifies the instantiation of two identical memory controllers that share an address/command bus through the use of Ping Pong PHY. This parameter is available only if you specify the **Hard PHY and Hard Controller** option. When this parameter is enabled, the IP exposes two independent Avalon interfaces to the user logic, and a single external memory interface with double width for the data bus and the CS#, CKE, ODT, and CK/CK# signals. (Identifier: PHY_DDR4_USER_PING_PONG_EN)
Use clamshell layout	When clamshell layout is used, each rank requires two CS pins to configure the top and bottom memory chips separately. (Identifier: PHY_DDR4_USER_CLAMSHELL_EN)

Table 213. Group: General / Clocks

Display Name	Description
Memory clock frequency	Specifies the **operating frequency** of the memory interface in MHz. If you change the memory frequency, you should update the memory latency parameters on the **Memory** tab and the memory timing parameters on the **Mem Timing** tab. (Identifier: PHY_DDR4_MEM_CLK_FREQ_MHZ)
Use recommended PLL reference clock frequency	Specifies that the PLL reference clock frequency is automatically calculated for best performance. *If you want to specify a different PLL reference clock frequency, uncheck the check box for this parameter.* (Identifier: PHY_DDR4_DEFAULT_REF_CLK_FREQ)
PLL reference clock frequency	This parameter tells the IP what PLL reference clock frequency the user will supply. Users must select a valid PLL reference clock frequency from the list. The values in the list can change when the memory interface frequency

continued...

Display Name	Description
	changes and/or the clock rate of user logic changes. It is recommended to use the fastest possible PLL reference clock frequency because it leads to better jitter performance. Selection is required only if the user does not check the "Use recommended PLL reference clock frequency" option. (Identifier: PHY_DDR4_USER_REF_CLK_FREQ_MHZ)
PLL reference clock jitter	Specifies the **peak-to-peak jitter** on the PLL reference clock source. The clock source of the PLL reference clock must meet or exceed the following jitter requirements: 10ps peak to peak, or 1.42ps RMS at 1e-12 BER, 1.22ps at 1e-16 BER. (Identifier: PHY_DDR4_REF_CLK_JITTER_PS)
Clock rate of user logic	Specifies the relationship between the user logic clock frequency and the memory clock frequency. For example, if the memory clock sent from the FPGA to the memory device is toggling at 800MHz, a quarter-rate interface means that the user logic in the FPGA runs at 200MHz. The list of available options is dependent on the memory protocol and device family. (Identifier: PHY_DDR4_RATE_ENUM)
Core clocks sharing	When a design contains multiple interfaces of the same protocol, rate, frequency, and PLL reference clock source, they can share a common set of core clock domains. By sharing core clock domains, they **reduce clock network usage and avoid clock synchronization logic between the interfaces**. To share core clocks, denote one of the interfaces as **"Master"**, and the remaining interfaces as **"Slave"**. In the RTL, connect the `clks_sharing_master_out` signal from the master interface to the `clks_sharing_slave_in` signal of all the slave interfaces. Both master and slave interfaces still expose their own output clock ports in the RTL (for example, `emif_usr_clk`, `afi_clk`), but the physical signals are equivalent, hence it does not matter whether a clock port from a master or a slave is used. *As the combined width of all interfaces sharing the same core clock increases, you may encounter timing closure difficulty for transfers between the FPGA core and the periphery.* (Identifier: PHY_DDR4_CORE_CLKS_SHARING_ENUM)
Export clks_sharing_slave_out to facilitate multi-slave connectivity	When more than one slave exist, you can either connect the clks_sharing_master_out interface from the master to the clks_sharing_slave_in interface of all the slaves (i.e. one-to-many topology), OR, you can connect the clks_sharing_master_out interface to one slave, and connect the clks_sharing_slave_out interface of that slave to the next slave (i.e. daisy-chain topology). Both approaches produce the same result. The daisy-chain approach may be easier to achieve in the Platform Designer tool, whereas the one-to-many approach may be more intuitive. (Identifier: PHY_DDR4_CORE_CLKS_SHARING_EXPOSE_SLAVE_OUT)
Specify additional core clocks based on existing PLL	Displays additional parameters allowing you to create additional output clocks based on the existing PLL. This parameter **provides an alternative clock-generation mechanism for when your design exhausts available PLL resources**. The additional output clocks that you create can be fed into the core. Clock signals created with this parameter are synchronous to each other, but asynchronous to the memory interface core clock domains (such as `emif_usr_clk` or `afi_clk`). *You must follow proper clock-domain-crossing techniques when transferring data between clock domains.* (Identifier: PLL_ADD_EXTRA_CLKS)

Table 214. Group: General / Clocks / Additional Core Clocks

Display Name	Description
Number of additional core clocks	Specifies the number of additional output clocks to create from the PLL. (Identifier: PLL_USER_NUM_OF_EXTRA_CLKS)

Table 215. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_0

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_5)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_5)

Table 216. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_1

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_6)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_6)

Table 217. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_2

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_7)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_7)

Table 218. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_3

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_8)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_8)

7.1.2. Intel Stratix 10 EMIF IP DDR4 Parameters: FPGA I/O

You should use Hyperlynx* or similar simulators to determine the best settings for your board. Refer to the EMIF Simulation Guidance wiki page for additional information.

Table 219. Group: FPGA I/O / FPGA I/O Settings

Display Name	Description
Voltage	The voltage level for the I/O pins driving the signals between the memory device and the FPGA memory interface. (Identifier: PHY_DDR4_IO_VOLTAGE)
Use default I/O settings	Specifies that a legal set of I/O settings are automatically selected. The default I/O settings are not necessarily optimized for a specific board. *To achieve optimal signal integrity, perform I/O simulations with IBIS models and enter the I/O settings manually, based on simulation results*. (Identifier: PHY_DDR4_DEFAULT_IO)

Table 220. Group: FPGA I/O / FPGA I/O Settings / Address/Command

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the address/command pins of the memory interface. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR4_USER_AC_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR4_USER_AC_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_DDR4_USER_AC_SLEW_RATE_ENUM)

Table 221. Group: FPGA I/O / FPGA I/O Settings / Memory Clock

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the memory clock pins. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR4_USER_CK_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR4_USER_CK_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_DDR4_USER_CK_SLEW_RATE_ENUM)

Table 222. Group: FPGA I/O / FPGA I/O Settings / Data Bus

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the data and data clock/strobe pins of the memory interface. The selected I/O standard option configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_DDR4_USER_DATA_IO_STD_ENUM)
Output mode	This parameter allows you to change the output current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR4_USER_DATA_OUT_MODE_ENUM)
Input mode	This parameter allows you to change the input termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_DDR4_USER_DATA_IN_MODE_ENUM)
Use recommended initial Vrefin	Specifies that the initial Vrefin setting is calculated automatically, to a reasonable value based on termination settings. (Identifier: PHY_DDR4_USER_AUTO_STARTING_VREFIN_EN)
Initial Vrefin	Specifies the **initial value for the reference voltage on the data pins(Vrefin)**. This value is entered as a percentage of the supply voltage level on the I/O pins. The specified value serves as a starting point and may be overridden by calibration to provide better timing margins. If you choose to **skip Vref calibration** (**Diagnostics** tab), this is the value that is used as the Vref for the interface. (Identifier: PHY_DDR4_USER_STARTING_VREFIN)

Table 223. Group: FPGA I/O / FPGA I/O Settings / PHY Inputs

Display Name	Description
PLL reference clock I/O standard	Specifies the I/O standard for the PLL reference clock of the memory interface. (Identifier: PHY_DDR4_USER_PLL_REF_CLK_IO_STD_ENUM)
RZQ I/O standard	Specifies the I/O standard for the RZQ pin used in the memory interface. (Identifier: PHY_DDR4_USER_RZQ_IO_STD_ENUM)

7.1.3. Intel Stratix 10 EMIF IP DDR4 Parameters: Memory

Table 224. Group: Memory / Topology

Display Name	Description
Memory format	Specifies the format of the external memory device. The following formats are supported: **Component** - a Discrete memory device; **UDIMM** - Unregistered/Unbuffered DIMM where address/control, clock, and data are unbuffered; **RDIMM** - Registered DIMM where address/control and clock are buffered; **LRDIMM** - Load Reduction DIMM where address/control, clock, and data are buffered. **LRDIMM** reduces the load to increase memory speed and supports higher densities than RDIMM; **SODIMM** - Small Outline DIMM is similar to UDIMM but smaller in size and is typically used for systems with limited space. Some memory protocols may not be available in all formats. (Identifier: MEM_DDR4_FORMAT_ENUM)
DQ width	Specifies the total number of data pins in the interface. (Identifier: MEM_DDR4_DQ_WIDTH)
DQ pins per DQS group	Specifies the total number of DQ pins per DQS group. (Identifier: MEM_DDR4_DQ_PER_DQS)
Number of clocks	Specifies the number of CK/CK# clock pairs exposed by the memory interface. Usually more than 1 pair is required for RDIMM/LRDIMM formats. The value of this parameter depends on the memory device selected; *refer to the data sheet for your memory device.* (Identifier: MEM_DDR4_CK_WIDTH)
Number of chip selects	Specifies the total number of chip selects in the interface, up to a maximum of 4. This parameter applies to **discrete components only**. (Identifier: MEM_DDR4_DISCRETE_CS_WIDTH)
Number of DIMMs	Total number of DIMMs. (Identifier: MEM_DDR4_NUM_OF_DIMMS)
Chip ID width	Specifies the number of chip ID pins. Only applicable to *registered and load-reduced DIMMs* that use 3DS/TSV memory devices. (Identifier: MEM_DDR4_CHIP_ID_WIDTH)
Number of physical ranks per DIMM	Number of ranks per DIMM. For LRDIMM, this represents the number of physical ranks on the DIMM behind the memory buffer (Identifier: MEM_DDR4_RANKS_PER_DIMM)
Row address width	Specifies the number of row address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of address pins needed for access to all available rows. (Identifier: MEM_DDR4_ROW_ADDR_WIDTH)
Column address width	Specifies the number of column address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of address pins needed for access to all available columns. (Identifier: MEM_DDR4_COL_ADDR_WIDTH)
Bank address width	Specifies the number of bank address pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of bank address pins needed for access to all available banks. (Identifier: MEM_DDR4_BANK_ADDR_WIDTH)

continued...

Display Name	Description
Bank group width	Specifies the number of bank group pins. *Refer to the data sheet for your memory device.* The density of the selected memory device determines the number of bank group pins needed for access to all available bank groups. (Identifier: MEM_DDR4_BANK_GROUP_WIDTH)
Data mask	Indicates whether the interface uses data mask (DM) pins. This feature allows specified portions of the data bus to be written to memory (not available in x4 mode). **One DM pin exists per DQS group.** (Identifier: MEM_DDR4_DM_EN)
Write DBI	Indicates whether the interface uses write data bus inversion (DBI). This feature provides **better signal integrity and write margin**. This feature is unavailable if Data Mask is enabled or in x4 mode. (Identifier: MEM_DDR4_WRITE_DBI)
Read DBI	Specifies whether the interface uses read data bus inversion (DBI). Enable this feature for **better signal integrity and read margin**. This feature is not available in x4 configurations. (Identifier: MEM_DDR4_READ_DBI)
Enable address mirroring for odd chip-selects	Enabling address mirroring for multi-CS discrete components. Typically used when components are arranged in a clamshell layout. (Identifier: MEM_DDR4_DISCRETE_MIRROR_ADDRESSING_EN)
Enable address mirroring for odd ranks	Enabling address mirroring for dual-rank or quad-rank DIMM. (Identifier: MEM_DDR4_MIRROR_ADDRESSING_EN)
Enable ALERT#/PAR pins	Allows address/command calibration, which may provide better margins on the address/command bus. The `alert_n` signal is not accessible in the AFI or Avalon domains. This means there is no way to know whether a parity error has occurred during user mode. The parity pin is a dedicated pin in the address/command bank, but the `alert_n` pin can be placed in any bank that spans the memory interface. You should explicitly choose the location of the `alert_n` pin and place it in the address/command bank. Address/command parity is checked only during calibration, not in user mode. Because the `alert_n` pin is not accessible via the AFI or Avalon interfaces, changing the address/command parity latency option from the default value in advanced mode register settings, is not recommended. (Identifier: MEM_DDR4_ALERT_PAR_EN)
ALERT# pin placement	Specifies placement for the `mem_alert_n` signal. If you select **"I/O Lane with Address/Command Pins"**, you can pick the I/O lane and pin index in the add/cmd bank with the subsequent drop down menus. If you select **"I/O Lane with DQS Group"**, you can specify the DQS group with which to place the `mem_alert_n` pin. If you select **"Automatically select a location"**, the IP automatically selects a pin for the `mem_alert_n` signal. If you select this option, no additional location constraints can be applied to the `mem_alert_n` pin, or a fitter error will result during compilation. For optimum signal integrity, you should choose **"I/O Lane with Address/Command Pins"**. For interfaces containing multiple memory devices, it is recommended to connect the ALERT# pins together to the ALERT# pin on the FPGA. (Identifier: MEM_DDR4_ALERT_N_PLACEMENT_ENUM)
DQS group of ALERT#	Select the DQS group with which the ALERT# pin is placed. (Identifier: MEM_DDR4_ALERT_N_DQS_GROUP)
Address/command I/O lane of ALERT#	Select the lane of the Address/Command I/O Tile where ALERT# pin is placed. (Identifier: MEM_DDR4_ALERT_N_AC_LANE)
Pin index of ALERT#	Select the pin of the Address/Command I/O Lane where ALERT# pin is placed. (Identifier: MEM_DDR4_ALERT_N_AC_PIN)

Table 225. Group: Memory / Latency and Burst

Display Name	Description
Memory CAS latency setting	Specifies the number of clock cycles between the read command and the availability of the first bit of output data at the memory device. Overall read latency equals the additive latency (AL) + the CAS latency (CL). Overall read latency depends on the memory device selected; *refer to the datasheet for your device.* (Identifier: MEM_DDR4_TCL)
Memory write CAS latency setting	Specifies the number of clock cycles from the release of internal write to the latching of the first data in at the memory device. This value depends on the memory device selected; *refer to the datasheet for your device.* (Identifier: MEM_DDR4_WTCL)
Memory additive CAS latency setting	Determines the posted CAS additive latency of the memory device. Enable this feature to **improve command and bus efficiency, and increase system bandwidth**. (Identifier: MEM_DDR4_ATCL_ENUM)

Table 226. Group: Memory / Mode Register Settings

Display Name	Description
Hide advanced mode register settings	Show or hide advanced mode register settings. Changing advanced mode register settings to non-default values is strongly discouraged. (Identifier: MEM_DDR4_HIDE_ADV_MR_SETTINGS)
Addr/CMD parity latency	Additional latency incurred by enabling address/command parity check after calibration. **Select a value** to enable address/command parity with the latency associated with the selected value. Select **Disable** to disable address/command parity. Address/command is enabled automatically and as-needed during calibration regardless of the value of this setting. (Identifier: MEM_DDR4_AC_PARITY_LATENCY)
Burst Length	Specifies the DRAM burst length which determines how many consecutive addresses should be accessed for a given read/write command. (Identifier: MEM_DDR4_BL_ENUM)
Read Burst Type	Indicates whether accesses within a given burst are in sequential or interleaved order. Select sequential if you are using the Intel-provided memory controller. (Identifier: MEM_DDR4_BT_ENUM)
Enable the DLL in memory device	Enable the DLL in memory device (Identifier: MEM_DDR4_DLL_EN)
Auto self-refresh method	Indicates whether to enable or disable auto self-refresh. Auto self-refresh allows the controller to issue self-refresh requests, rather than manually issuing self-refresh in order for memory to retain data. (Identifier: MEM_DDR4_ASR_ENUM)
Write CRC enable	Write CRC enable (Identifier: MEM_DDR4_WRITE_CRC)
DDR4 geardown mode	Set DDR4 geardown mode for control signals at high frequency (Identifier: MEM_DDR4_GEARDOWN)
Per-DRAM addressability	Per-DRAM addressability enable (Identifier: MEM_DDR4_PER_DRAM_ADDR)
Temperature sensor readout	Temperature sensor readout enable (Identifier: MEM_DDR4_TEMP_SENSOR_READOUT)
Fine granularity refresh	Increased frequency of refresh in exchange for shorter refresh. **Shorter tRFC** and increased cycle time can produce **higher bandwidth**. (Identifier: MEM_DDR4_FINE_GRANULARITY_REFRESH)
MPR read format	Multipurpose register readout format (Identifier: MEM_DDR4_MPR_READ_FORMAT)
Maximum power down mode	Maximum power down mode (Identifier: MEM_DDR4_MAX_POWERDOWN)
Temperature controlled refresh range	Indicates temperature controlled refresh range where normal temperature mode covers 0C to 85C and extended mode covers 0C to 95C. (Identifier: MEM_DDR4_TEMP_CONTROLLED_RFSH_RANGE)

continued...

Display Name	Description
Temperature controlled refresh enable	Indicates whether to enable temperature controlled refresh, which allows the device to adjust the internal refresh period to be longer than tREFI of the normal temperature range by skipping external refresh commands. (Identifier: MEM_DDR4_TEMP_CONTROLLED_RFSH_ENA)
Internal VrefDQ monitor	Indicates whether to enable the internal VrefDQ monitor. (Identifier: MEM_DDR4_INTERNAL_VREFDQ_MONITOR)
CS to Addr/CMD Latency	CS to Addr/CMD Latency (CAL mode) for idle state DRAM receiver power reduction (Identifier: MEM_DDR4_CAL_MODE)
Self refresh abort	Self refresh abort for latency reduction. (Identifier: MEM_DDR4_SELF_RFSH_ABORT)
Read preamble training mode enable	Read preamble training mode enable. (Identifier: MEM_DDR4_READ_PREAMBLE_TRAINING)
Read preamble	Number of read preamble cycles. This mode register setting determines the number of cycles DQS (read) will go low before starting to toggle. It is strongly recommended to use the default read preamble setting. (Identifier: MEM_DDR4_READ_PREAMBLE)
Write preamble	Write preamble cycles. It is strongly recommended to use the default write preamble setting. (Identifier: MEM_DDR4_WRITE_PREAMBLE)
ODT input buffer during powerdown mode	Indicates whether to enable on-die termination (ODT) input buffer during powerdown mode. (Identifier: MEM_DDR4_ODT_IN_POWERDOWN)
Addr/CMD persistent error	If set, Addr/CMD parity errors continue to be checked after a previous Addr/CMD parity error (Identifier: MEM_DDR4_AC_PERSISTENT_ERROR)

7.1.4. Intel Stratix 10 EMIF IP DDR4 Parameters: Mem I/O

Table 227. Group: Mem I/O / Memory I/O Settings

Display Name	Description
Output drive strength setting	Specifies the output driver impedance setting at the memory device. To obtain optimum signal integrity performance, **select option based on board simulation results.** (Identifier: MEM_DDR4_DRV_STR_ENUM)
Dynamic ODT (Rtt_WR) value	Specifies the mode of the dynamic on-die termination (ODT) during writes to the memory device (used for **multi-rank configurations**). For optimum signal integrity performance, **select this option based on board simulation results**. (Identifier: MEM_DDR4_RTT_WR_ENUM)
ODT Rtt nominal value	Determines the nominal on-die termination value applied to the DRAM. The termination is applied any time that ODT is asserted. If you specify a different value for RTT_WR, that value takes precedence over the values mentioned here. For optimum signal integrity performance, **select your option based on board simulation results**. (Identifier: MEM_DDR4_RTT_NOM_ENUM)
RTT PARK	If set, the value is applied when the DRAM is not being written **AND** ODT is not asserted HIGH. (Identifier: MEM_DDR4_RTT_PARK)
RCD CA Input Bus Termination	Specifies the input termination setting for the following pins of the registering clock driver: DA0..DA17, DBA0..DBA1, DBG0..DBG1, DACT_n, DC2, DPAR. This parameter determines the value of bits DA[1:0] of control word RC7x of the registering clock driver. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_RCD_CA_IBT_ENUM)

continued...

Display Name	Description
RCD DCS[3:0]_n Input Bus Termination	Specifies the input termination setting for the following pins of the registering clock driver: DCS[3:0]_n. This parameter determines the value of bits DA[3:2] of control word RC7x of the registering clock driver. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_RCD_CS_IBT_ENUM)
RCD DCKE Input Bus Termination	Specifies the input termination setting for the following pins of the registering clock driver: DCKE0, DCKE1. This parameter determines the value of bits DA[5:4] of control word RC7x of the registering clock driver. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_RCD_CKE_IBT_ENUM)
RCD DODT Input Bus Termination	Specifies the input termination setting for the following pins of the registering clock driver: DODT0, DODT1. This parameter determines the value of bits DA[7:6] of control word RC7x of the registering clock driver. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_RCD_ODT_IBT_ENUM)
DB Host Interface DQ RTT_NOM	Specifies the RTT_NOM setting for the host interface of the data buffer. Only "RTT_NOM disabled" is supported. This parameter determines the value of the control word BC00 of the data buffer. (Identifier: MEM_DDR4_DB_RTT_NOM_ENUM)
DB Host Interface DQ RTT_WR	Specifies the RTT_WR setting of the host interface of the data buffer. This parameter determines the value of the control word BC01 of the data buffer. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_DB_RTT_WR_ENUM)
DB Host Interface DQ RTT_PARK	Specifies the RTT_PARK setting for the host interface of the data buffer. This parameter determines the value of control word BC02 of the data buffer. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_DB_RTT_PARK_ENUM)
DB Host Interface DQ Driver	Specifies the driver impedance setting for the host interface of the data buffer. This parameter determines the value of the control word BC03 of the data buffer. **Perform board simulation to obtain the optimal value for this setting.** (Identifier: MEM_DDR4_DB_DQ_DRV_ENUM)
Use recommended initial VrefDQ value	Specifies to use the recommended initial VrefDQ value. This value is used as a starting point and may change after calibration. (Identifier: MEM_DDR4_DEFAULT_VREFOUT)
VrefDQ training value	VrefDQ training value. (Identifier: MEM_DDR4_USER_VREFDQ_TRAINING_VALUE)
VrefDQ training range	VrefDQ training range. (Identifier: MEM_DDR4_USER_VREFDQ_TRAINING_RANGE)

Table 228. Group: Mem I/O / RDIMM/LRDIMM Serial Presence Detect (SPD) Data

Display Name	Description
SPD Byte 137 - RCD Drive Strength for Command/Address	Specifies the drive strength of the registering clock driver's control and command/address outputs to the DRAM. The value must come from **Byte 137 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_137_RCD_CA_DRV)
SPD Byte 138 - RCD Drive Strength for CK	Specifies the drive strength of the registering clock driver's clock outputs to the DRAM. The value must come from **Byte 138 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_138_RCD_CK_DRV)
SPD Byte 140 - DRAM VrefDQ for Package Rank 0	Specifies the VrefDQ setting for package rank 0 of an LRDIMM. The value must come from **Byte 140 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_140_DRAM_VREFDQ_R0)

continued...

Display Name	Description
SPD Byte 141 - DRAM VrefDQ for Package Rank 1	Specifies the VrefDQ setting for package rank 1 of an LRDIMM. The value must come from **Byte 141 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_141_DRAM_VREFDQ_R1)
SPD Byte 142 - DRAM VrefDQ for Package Rank 2	Specifies the VrefDQ setting for package rank 2 (if it exists) of an LRDIMM. The value must come from **Byte 142 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_142_DRAM_VREFDQ_R2)
SPD Byte 143 - DRAM VrefDQ for Package Rank 3	Specifies the VrefDQ setting for package rank 3 (if it exists) of an LRDIMM. The value must come from **Byte 143 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_143_DRAM_VREFDQ_R3)
SPD Byte 144 - DB VrefDQ for DRAM Interface	Specifies the VrefDQ setting of the data buffer's DRAM interface. The value must come from **Byte 144 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_144_DB_VREFDQ)
SPD Byte 145-147 - DB MDQ Drive Strength and RTT	Specifies the drive strength of the MDQ pins of the data buffer's DRAM interface. The value must come from **either Byte 145 (data rate = 1866), 146 (1866 data rate = 2400), or 147 (2400 data rate = 3200) of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_145_DB_MDQ_DRV)
SPD Byte 148 - DRAM Drive Strength	Specifies the drive strength of the DRAM. The value must come from **Byte 148 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_148_DRAM_DRV)
SPD Byte 149-151 - DRAM ODT (RTT_WR and RTT_NOM)	Specifies the RTT_WR and RTT_NOM setting of the DRAM. The value must come from **either Byte 149 (data rate = 1866), 150 (1866 data rate = 2400), or 151 (2400 data rate = 3200) of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_149_DRAM_RTT_WR_NOM)
SPD Byte 152-154 - DRAM ODT (RTT_PARK)	Specifies the RTT_PARK setting of the DRAM. The value must come from **either Byte 152 (data rate = 1866), 153 (1866 data rate = 2400), or 154 (2400 data rate = 3200) of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_152_DRAM_RTT_PARK)
RCD and DB Manufacturer (LSB)	Specifies the LSB of the ID code of the registering clock driver and data buffer manufacturer. The value must come from **Byte 133 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_133_RCD_DB_VENDOR_LSB)
RCD and DB Manufacturer (MSB)	Specifies the MSB of the ID code of the registering clock driver and data buffer manufacturer. The value must come from **Byte 134 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_134_RCD_DB_VENDOR_MSB)
RCD Revision Number	Specifies the die revision of the registering clock driver. The value must come from **Byte 135 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_135_RCD_REV)
DB Revision Number	Specifies the die revision of the data buffer. The value must come from **Byte 139 of the SPD** from the DIMM vendor. (Identifier: MEM_DDR4_SPD_139_DB_REV)

Table 229. Group: Mem I/O / ODT Activation

Display Name	Description
Use Default ODT Assertion Tables	Enables the default ODT assertion pattern as determined from vendor guidelines. These settings are provided as a default only; *you should simulate your memory interface to determine the optimal ODT settings and assertion patterns.* (Identifier: MEM_DDR4_USE_DEFAULT_ODT)

7.1.5. Intel Stratix 10 EMIF IP DDR4 Parameters: Mem Timing

These parameters should be read from the table in the datasheet associated with the speed bin of the memory device (not necessarily the frequency at which the interface is running).

Table 230. Group: Mem Timing / Parameters dependent on Speed Bin

Display Name	Description
Speed bin	The speed grade of the memory device used. This parameter refers to the maximum rate at which the memory device is specified to run. (Identifier: MEM_DDR4_SPEEDBIN_ENUM)
tIS (base)	tIS (base) refers to the **setup time for the Address/Command/Control (A) bus** to the rising edge of CK. (Identifier: MEM_DDR4_TIS_PS)
tIS (base) AC level	tIS (base) AC level refers to the **voltage level which the address/command signal must cross and remain above during the setup margin window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire setup period. (Identifier: MEM_DDR4_TIS_AC_MV)
tIH (base)	tIH (base) refers to the **hold time for the Address/Command (A) bus** after the rising edge of CK. Depending on what AC level the user has chosen for a design, the hold margin can vary (this variance will be automatically determined when the user chooses the "**tIH (base) AC level**"). (Identifier: MEM_DDR4_TIH_PS)
tIH (base) DC level	tIH (base) DC level refers to the **voltage level which the address/command signal must not cross during the hold window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire hold period. (Identifier: MEM_DDR4_TIH_DC_MV)
TdiVW_total	`TdiVW_total` describes the minimum horizontal width of the DQ eye opening required by the receiver (memory device/DIMM). It is measured in UI (1UI = half the memory clock period). (Identifier: MEM_DDR4_TDIVW_TOTAL_UI)
VdiVW_total	`VdiVW_total` describes the **Rx Mask voltage**, or the minimum vertical width of the DQ eye opening required by the receiver (memory device/DIMM). It is measured in mV. (Identifier: MEM_DDR4_VDIVW_TOTAL)
tDQSQ	tDQSQ describes the **latest valid transition of the associated DQ pins for a READ**. tDQSQ specifically refers to the DQS, DQS# to DQ skew. It is the length of time between the DQS, DQS# crossing to the last valid transition of the slowest DQ pin in the DQ group associated with that DQS strobe. (Identifier: MEM_DDR4_TDQSQ_UI)
tQH	tQH specifies the **output hold time for the DQ in relation to DQS, DQS#**. It is the length of time between the DQS, DQS# crossing to the earliest invalid transition of the fastest DQ pin in the DQ group associated with that DQS strobe. (Identifier: MEM_DDR4_TQH_UI)
tDVWp	Data valid window per device per pin (Identifier: MEM_DDR4_TDVWP_UI)
tDQSCK	tDQSCK describes the **skew between the memory clock (CK) and the input data strobes (DQS) used for reads**. It is the time between the rising data strobe edge (DQS, DQS#) relative to the rising CK edge. (Identifier: MEM_DDR4_TDQSCK_PS)
tDQSS	tDQSS describes the **skew between the memory clock (CK) and the output data strobes used for writes**. It is the time between the rising data strobe edge (DQS, DQS#) relative to the rising CK edge. (Identifier: MEM_DDR4_TDQSS_CYC)
	continued...

Display Name	Description
tQSH	tQSH refers to the differential High Pulse Width, which is measured as a percentage of tCK. It is the **time during which the DQS is high for a read**. (Identifier: MEM_DDR4_TQSH_CYC)
tDSH	tDSH specifies the **write DQS hold time**. This is the time difference between the rising CK edge and the falling edge of DQS, measured as a percentage of tCK. (Identifier: MEM_DDR4_TDSH_CYC)
tDSS	tDSS describes the **time between the falling edge of DQS to the rising edge of the next CK transition**. (Identifier: MEM_DDR4_TDSS_CYC)
tWLS	tWLS describes the **write leveling setup time**. It is measured from the rising edge of CK to the rising edge of DQS. (Identifier: MEM_DDR4_TWLS_CYC)
tWLH	tWLH describes the **write leveling hold time**. It is measured from the rising edge of DQS to the rising edge of CK. (Identifier: MEM_DDR4_TWLH_CYC)
tINIT	tINIT describes the **time duration of the memory initialization after a device power-up**. After RESET_n is de-asserted, wait for another 500us until CKE becomes active. During this time, the DRAM will start internal initialization; this will be done independently of external clocks. (Identifier: MEM_DDR4_TINIT_US)
tMRD	The mode register set command cycle time, tMRD is the **minimum time period required between two MRS commands**. (Identifier: MEM_DDR4_TMRD_CK_CYC)
tRAS	tRAS describes the **activate to precharge duration**. A row cannot be deactivated until the tRAS time has been met. Therefore tRAS determines how long the memory has to wait after a activate command before a precharge command can be issued to close the row. (Identifier: MEM_DDR4_TRAS_NS)
tRCD	tRCD, **row command delay**, describes **the active to read/write time**. It is the amount of delay between the activation of a row through the RAS command and the access to the data through the CAS command. (Identifier: MEM_DDR4_TRCD_NS)
tRP	tRP refers to the **Precharge (PRE) command period**. It describes how long it takes for the memory to disable access to a row by precharging and before it is ready to activate a different row. (Identifier: MEM_DDR4_TRP_NS)
tWR	tWR refers to the **Write Recovery time**. It specifies the amount of clock cycles needed to complete a write before a precharge command can be issued. (Identifier: MEM_DDR4_TWR_NS)

Table 231. Group: Mem Timing / Parameters dependent on Speed Bin, Operating Frequency, and Page Size

Display Name	Description
tRRD_S	tRRD_S refers to the **Activate to Activate Command Period (short)**. It is the minimum time interval between two activate commands to the **different bank groups**. For 3DS devices, this parameter is the same as tRRD_S_slr (i.e. tRRD_S within the same logical rank) in the memory data sheet. (Identifier: MEM_DDR4_TRRD_S_CYC)
tRRD_L	tRRD_L refers to the **Activate to Activate Command Period (long)**. It is the minimum time interval (measured in memory clock cycles) between two activate commands to the **same bank group**. For 3DS devices, this parameter is the same as tRRD_L_slr (i.e. tRRD_L within the same logical rank) in the memory data sheet. (Identifier: MEM_DDR4_TRRD_L_CYC)

continued...

Display Name	Description
tRRD_dlr	tRRD_dlr refers to the **Activate to Activate Command Period to Different Logical Ranks**. It is the minimum time interval (measured in memory clock cycles) between two activate commands to different logical ranks within a 3DS DDR4 device. (Identifier: MEM_DDR4_TRRD_DLR_CYC)
tFAW	tFAW refers to the **four activate window time**. It describes the period of time during which only four banks can be active. For 3DS devices, this parameter is the same as tFAW_slr (i.e. tFAW within the same logical rank) in the memory data sheet. (Identifier: MEM_DDR4_TFAW_NS)
tFAW_dlr	tFAW_dlr refers to the **four activate window to different logical ranks**. It describes the period of time during which only four banks can be active across all logical ranks within a 3DS DDR4 device. (Identifier: MEM_DDR4_TFAW_DLR_CYC)
tCCD_S	tCCD_S refers to the **CAS_n-to-CAS_n delay (short)**. It is the minimum time interval between two read/write (CAS) commands to **different bank groups**. (Identifier: MEM_DDR4_TCCD_S_CYC)
tCCD_L	tCCD_L refers to the **CAS_n-to-CAS_n delay (long)**. It is the minimum time interval between two read/write (CAS) commands to the **same bank group**. (Identifier: MEM_DDR4_TCCD_L_CYC)
tWTR_S	tWTR_S or Write Timing Parameter refers to the **Write to Read period for different bank groups**. It describes the delay from start of internal write transaction to internal read command, for accesses to the different bank group. The delay is measured from the first rising memory clock edge after the last write data is received to the rising memory clock edge when a read command is received. (Identifier: MEM_DDR4_TWTR_S_CYC)
tWTR_L	tWTR_L or Write Timing Parameter refers to the **Write to Read period for the same bank group**. It describes the delay from start of internal write transaction to internal read command, for accesses to the same bank group. The delay is measured from the first rising memory clock edge after the last write data is received to the rising memory clock edge when a read command is received. (Identifier: MEM_DDR4_TWTR_L_CYC)

Table 232. Group: Mem Timing / Parameters dependent on Density and Temperature

Display Name	Description
tRFC	tRFC refers to the **Refresh Cycle Time**. It is the amount of delay after a refresh command before an activate command can be accepted by the memory. This parameter is dependent on the memory density and is necessary for proper hardware functionality. For 3DS devices, this parameter is the same as tRFC_slr (i.e. tRFC within the same logical rank) in the memory data sheet. (Identifier: MEM_DDR4_TRFC_NS)
tRFC_dlr	tRFC_dlr refers to the **Refresh Cycle Time to different logical rank**. It is the amount of delay after a refresh command to one logical rank before an activate command can be accepted by another logical rank within a 3DS DDR4 device. This parameter is dependent on the memory density and is necessary for proper hardware functionality. (Identifier: MEM_DDR4_TRFC_DLR_NS)
tREFI	tREFI refers to the **average periodic refresh interval**. It is the maximum amount of time the memory can tolerate in between each refresh command (Identifier: MEM_DDR4_TREFI_US)

7.1.6. Intel Stratix 10 EMIF IP DDR4 Parameters: Board

Table 233. Group: Board / Intersymbol Interference/Crosstalk

Display Name	Description
Use default ISI/crosstalk values	You can enable this option to use default intersymbol interference and crosstalk values for your topology. Note that the default values are not optimized for your board. *For optimal signal integrity, it is recommended that you do not enable this parameter, but instead perform I/O simulation using IBIS models and Hyperlynx*, and manually enter values based on your simulation results, instead of using the default values.* (Identifier: BOARD_DDR4_USE_DEFAULT_ISI_VALUES)
Address and command ISI/crosstalk	The address and command window reduction due to ISI and crosstalk effects. The number to be entered is the **total loss of margin on both the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR4_USER_AC_ISI_NS)
Read DQS/DQS# ISI/crosstalk	The reduction of the read data window due to ISI and crosstalk effects on the DQS/DQS# signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR4_USER_RCLK_ISI_NS)
Read DQ ISI/crosstalk	The reduction of the read data window due to ISI and crosstalk effects on the DQ signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on the setup and hold side (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR4_USER_RDATA_ISI_NS)
Write DQS/DQS# ISI/crosstalk	The reduction of the write data window due to ISI and crosstalk effects on the DQS/DQS# signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR4_USER_WCLK_ISI_NS)
Write DQ ISI/crosstalk	The reduction of the write data window due to ISI and crosstalk effects on the DQ signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_DDR4_USER_WDATA_ISI_NS)

Table 234. Group: Board / Board and Package Skews

Display Name	Description
Package deskewed with board layout (DQS group)	Enable this parameter if you are compensating for package skew on the DQ, DQS, and DM buses in the board layout. **Include package skew in calculating the following board skew parameters.** (Identifier: BOARD_DDR4_IS_SKEW_WITHIN_DQS_DESKEWED)
Maximum board skew within DQS group	The largest skew between all DQ and DM pins in a DQS group. This value affects the read capture and write margins. (Identifier: BOARD_DDR4_BRD_SKEW_WITHIN_DQS_NS)
Maximum system skew within DQS group	The largest skew between all DQ and DM pins in a DQS group. Enter combined board and package skew. This value affects the read capture and write margins. (Identifier: BOARD_DDR4_PKG_BRD_SKEW_WITHIN_DQS_NS)
Package deskewed with board layout (address/command bus)	Enable this parameter if you are compensating for package skew on the address, command, control, and memory clock buses in the board layout. **Include package skew in calculating the following board skew parameters.** (Identifier: BOARD_DDR4_IS_SKEW_WITHIN_AC_DESKEWED)

continued...

Display Name	Description
Maximum board skew within address/ command bus	The largest skew between the address and command signals. (Identifier: BOARD_DDR4_BRD_SKEW_WITHIN_AC_NS)
Maximum system skew within address/ command bus	The largest skew between the address and command signals. Enter combined board and package skew. (Identifier: BOARD_DDR4_PKG_BRD_SKEW_WITHIN_AC_NS)
Average delay difference between DQS and CK	The average delay difference between the DQS signals and the CK signal, calculated by averaging the longest and smallest DQS trace delay minus the CK trace delay. Positive values represent DQS signals that are longer than CK signals and negative values represent DQS signals that are shorter than CK signals. (Identifier: BOARD_DDR4_DQS_TO_CK_SKEW_NS)
Maximum delay difference between DIMMs/devices	The largest propagation delay on DQ signals between ranks (*applicable only when there is more than one rank*). For example: when you configure two ranks using one DIMM there is a short distance between the ranks for the same DQ pin; when you implement two ranks using two DIMMs the distance is larger. (Identifier: BOARD_DDR4_SKEW_BETWEEN_DIMMS_NS)
Maximum skew between DQS groups	The largest skew between DQS signals. (Identifier: BOARD_DDR4_SKEW_BETWEEN_DQS_NS)
Average delay difference between address/command and CK	The average delay difference between the address/command signals and the CK signal, calculated by averaging the longest and smallest address/ command signal trace delay minus the maximum CK trace delay. Positive values represent address and command signals that are longer than CK signals and negative values represent address and command signals that are shorter than CK signals. (Identifier: BOARD_DDR4_AC_TO_CK_SKEW_NS)
Maximum CK delay to DIMM/device	The delay of the longest CK trace from the FPGA to any DIMM/device. (Identifier: BOARD_DDR4_MAX_CK_DELAY_NS)
Maximum DQS delay to DIMM/device	The delay of the longest DQS trace from the FPGA to any DIMM/device (Identifier: BOARD_DDR4_MAX_DQS_DELAY_NS)

7.1.7. Intel Stratix 10 EMIF IP DDR4 Parameters: Controller

Table 235. Group: Controller / Low Power Mode

Display Name	Description
Enable Auto Power-Down	Enable this parameter to have the controller automatically place the memory device into power-down mode after a specified number of idle controller clock cycles. The idle wait time is configurable. **All ranks must be idle to enter auto power-down.** (Identifier: CTRL_DDR4_AUTO_POWER_DOWN_EN)
Auto Power-Down Cycles	Specifies the number of idle controller cycles after which the memory device is placed into power-down mode. You can configure the idle waiting time. The supported range for number of cycles is from 1 to 65534. (Identifier: CTRL_DDR4_AUTO_POWER_DOWN_CYCS)

Table 236. Group: Controller / Efficiency

Display Name	Description
Enable User Refresh Control	When enabled, user logic has complete control and is responsible for issuing adaquate refresh commands to the memory devices, via the MMR interface. This feature provides increased control over worst-case read latency and enables you to issue refresh bursts during idle periods. (Identifier: CTRL_DDR4_USER_REFRESH_EN)
Enable Auto-Precharge Control	Select this parameter to enable the auto-precharge control on the controller top level. If you assert the auto-precharge control signal while requesting a read or write burst, you can specify whether the controller should close (auto-precharge) the currently open page at the end of the read or write burst, potentially making a future access to a different page of the same bank faster. (Identifier: CTRL_DDR4_AUTO_PRECHARGE_EN)
Address Ordering	Controls the mapping between Avalon addresses and memory device addresses. **By changing the value of this parameter, you can change the mappings between the Avalon-MM address and the DRAM address.** (*CS* = chip select, *CID* = chip ID in 3DS/TSV devices, *BG* = bank group address, *Bank* = bank address, *Row* = row address, *Col* = column address) (Identifier: CTRL_DDR4_ADDR_ORDER_ENUM)
Enable Reordering	Enable this parameter to allow the controller to perform command and data reordering. **Reordering can improve efficiency by reducing bus turnaround time and row/bank switching time.** Data reordering allows the single-port memory controller to change the order of read and write commands to achieve highest efficiency. Command reordering allows the controller to issue bank management commands early based on incoming patterns, so that the desired row in memory is already open when the command reaches the memory interface. *For more information, refer to the Data Reordering topic in the EMIF Handbook.* (Identifier: CTRL_DDR4_REORDER_EN)
Starvation limit for each command	Specifies the **number of commands that can be served before a waiting command is served**. The controller employs a counter to ensure that all requests are served after a pre-defined interval -- this ensures that low priority requests are not ignored, when doing data reordering for efficiency. The valid range for this parameter is from 1 to 63. *For more information, refer to the Starvation Control topic in the EMIF Handbook.* (Identifier: CTRL_DDR4_STARVE_LIMIT)
Enable Command Priority Control	Select this parameter to enable user-requested command priority control on the controller top level. This parameter instructs the controller to treat a read or write request as high-priority. The controller attempts to fill high-priority requests sooner, to reduce latency. **Connect this interface to the conduit of your logic block that determines when the external memory interface IP treats the read or write request as a high-priority command.** (Identifier: CTRL_DDR4_USER_PRIORITY_EN)

Table 237. Group: Controller / Configuration, Status and Error Handling

Display Name	Description
Enable Memory-Mapped Configuration and Status Register (MMR) Interface	Enable this parameter to change or read memory timing parameters, memory address size, mode register settings, controller status, and request sideband operations. (Identifier: CTRL_DDR4_MMR_EN)
Enable Error Detection and Correction Logic with ECC	Enables error-correction code (ECC) for **single-bit error correction and double-bit error detection**. *ECC is implemented as soft logic.* (Identifier: CTRL_DDR4_ECC_EN)

continued...

Display Name	Description
Enable Auto Error Correction to External Memory	Specifies that the controller automatically schedule and perform a write back to the external memory when a single-bit error is detected. Regardless of whether the option is enabled or disabled, the ECC feature always corrects single-bit errors before returning the read data to user logic. (Identifier: CTRL_DDR4_ECC_AUTO_CORRECTION_EN)
Enable ctrl_ecc_readdataerror signal to indicate uncorrectable data errors	Select this option to enable the `ctrl_ecc_readdataerror` signal on the controller top level. The signal has the same timing as the read data valid signal of the Controller Avalon Memory-Mapped interface, and is asserted high to indicate that the read data returned by the Controller in the same cycle contains errors uncorrectable by the ECC logic. (Identifier: CTRL_DDR4_ECC_READDATAERROR_EN)
Export error-correction code (ECC) status ports	Enable this parameter to export ECC status ports.

Table 238. Group: Controller / Data Bus Turnaround Time

Display Name	Description
Additional read-to-write turnaround time (same rank)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from **a read to a write within the same logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR4_RD_TO_WR_SAME_CHIP_DELTA_CYCS)
Additional write-to-read turnaround time (same rank)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **write to a read within the same logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR4_WR_TO_RD_SAME_CHIP_DELTA_CYCS)
Additional read-to-read turnaround time (different ranks)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **read of one logical rank to a read of another logical rank**. This can resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR4_RD_TO_RD_DIFF_CHIP_DELTA_CYCS)
Additional read-to-write turnaround time (different ranks)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **read of one logical rank to a write of another logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR4_RD_TO_WR_DIFF_CHIP_DELTA_CYCS)
Additional write-to-write turnaround time (different ranks)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **write of one logical rank to a write of another logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR4_WR_TO_WR_DIFF_CHIP_DELTA_CYCS)
Additional write-to-read turnaround time (different ranks)	Specifies additional number of **idle controller (not DRAM)** cycles when switching the data bus from a **write of one logical rank to a read of another logical rank**. This can help resolve bus contention problems specific to your board topology. The value is added to the default which is calculated automatically. *Use the default setting unless you suspect a problem exists.* (Identifier: CTRL_DDR4_WR_TO_RD_DIFF_CHIP_DELTA_CYCS)

7.1.8. Intel Stratix 10 EMIF IP DDR4 Parameters: Diagnostics

Table 239. Group: Diagnostics / Simulation Options

Display Name	Description
Calibration mode	Specifies whether to **skip memory interface calibration** during simulation, or to **simulate the full calibration** process. Simulating the full calibration process can take hours (or even days), depending on the width and depth of the memory interface. You can achieve much faster simulation times by skipping the calibration process, but that is only expected to work when the memory model is ideal and the interconnect delays are zero. *If you enable this parameter, the interface still performs some memory initialization before starting normal operations.* Abstract PHY is supported with skip calibration. (Identifier: DIAG_DDR4_SIM_CAL_MODE_ENUM)
Abstract phy for fast simulation	Specifies that the system use Abstract PHY for simulation. **Abstract PHY replaces the PHY with a model for fast simulation and can reduce simulation time by 3-10 times.** Abstract PHY is available for certain protocols and device families, and only when you select **Skip Calibration**. (Identifier: DIAG_DDR4_ABSTRACT_PHY)
Show verbose simulation debug messages	This option allows adjusting the verbosity of the simulation output messages. (Identifier: DIAG_DDR4_SIM_VERBOSE)

Table 240. Group: Diagnostics / Calibration Debug Options

Display Name	Description
Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies the connectivity of an Avalon slave interface for use by the Quartus Prime EMIF Debug Toolkit or user core logic. If you set this parameter to "**Disabled**", no debug features are enabled. If you set this parameter to "**Export**", an Avalon slave interface named "`cal_debug`" is exported from the IP. To use this interface with the EMIF Debug Toolkit, you must instantiate and connect an EMIF debug interface IP core to it, or connect it to the `cal_debug_out` interface of another EMIF core. If you select "**Add EMIF Debug Interface**", an EMIF debug interface component containing a JTAG Avalon Master is connected to the debug port, allowing the core to be accessed by the EMIF Debug Toolkit. *Only one EMIF debug interface should be instantiated per I/O column.* You can chain additional EMIF or PHYLite cores to the first by enabling the "**Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option for all cores in the chain, and selecting "**Export**" for the "**Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option on all cores after the first. (Identifier: DIAG_DDR4_EXPORT_SEQ_AVALON_SLAVE)
Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies that the IP export an Avalon-MM master interface (`cal_debug_out`) which can connect to the `cal_debug` interface of other EMIF cores residing in the same I/O column. **This parameter applies only if the EMIF Debug Toolkit or On-Chip Debug Port is enabled.** *Refer to the Debugging Multiple EMIFs wiki page for more information about debugging multiple EMIFs.* (Identifier: DIAG_DDR4_EXPORT_SEQ_AVALON_MASTER)
First EMIF Instance in the Avalon Chain	If selected, this EMIF instance will be the head of the Avalon interface chain connected to the master. For simulation purposes it is needed to identify the first EMIF instance in the avalon Chain. (Identifier: DIAG_DDR4_EXPORT_SEQ_AVALON_HEAD_OF_CHAIN)

continued...

Display Name	Description
Interface ID	Identifies interfaces within the I/O column, for use by the EMIF Debug Toolkit and the On-Chip Debug Port. Interface IDs should be unique among EMIF cores within the same I/O column. If the **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** parameter is set to **Disabled**, the interface ID is unused. (Identifier: DIAG_DDR4_INTERFACE_ID)
Skip address/command leveling calibration	Specifies to skip the address/command leveling stage during calibration. Address/command leveling attempts to center the memory clock edge against CS# by adjusting delay elements inside the PHY, and then applying the same delay offset to the rest of the address and command pins. (Identifier: DIAG_DDR4_SKIP_CA_LEVEL)
Skip address/command deskew calibration	Specifies to skip the address/command deskew calibration stage. Address/command deskew performs per-bit deskew for the address and command pins. (Identifier: DIAG_DDR4_SKIP_CA_DESKEW)
Skip VREF calibration	Specifies to skip the VREF stage of calibration. **Enable this parameter for debug purposes only**; generally, you should include the VREF calibration stage during normal operation. (Identifier: DIAG_DDR4_SKIP_VREF_CAL)
Use Soft NIOS Processor for On-Chip Debug	Enables a soft Nios processor as a peripheral component to access the **On-Chip Debug Port**. *Only one interface in a column can activate this option.* (Identifier: DIAG_SOFT_NIOS_MODE)

Table 241. Group: Diagnostics / Example Design

Display Name	Description
Number of core clocks sharing slaves to instantiate in the example design	Specifies the number of core clock sharing slaves to instantiate in the example design. This parameter applies only if you set the "**Core clocks sharing**" parameter in the "**General**" tab to "**Master**" or "**Slave**". (Identifier: DIAG_DDR4_EX_DESIGN_NUM_OF_SLAVES)
Enable In-System-Sources-and-Probes	Enables In-System-Sources-and-Probes in the example design for *common debug signals, such as calibration status or example traffic generator per-bit status*. This parameter must be enabled if you want to do driver margining using the EMIF Debug Toolkit. (Identifier: DIAG_DDR4_EX_DESIGN_ISSP_EN)

Table 242. Group: Diagnostics / Traffic Generator (settings only applicable for example design)

Display Name	Description
Use configurable Avalon traffic generator 2.0	This option allows users to add the new configurable Avalon traffic generator to the example design. (Identifier: DIAG_DDR4_USE_TG_AVL_2)
Enable default traffic pattern (pattern configured during compile-time)	Specifies that the default traffic pattern will be enabled. If this parameter is enabled, a default traffic pattern will be run immediately every time the traffic generator comes out of reset. If this parameter is disabled, the traffic generator will not run any traffic until it is signaled to start by its Avalon configuration interface.
Enable user-configured traffic pattern (pattern configured during run-time)	Specifies that the user-defined traffic pattern will be enabled. If this parameter is enabled, the traffic generator will respond to the configuration interface and launch a user-configured traffic pattern when signaled to. If this parameter is disabled, the traffic generator will ignore commands on the configuration interface and will not run any user-defined traffic..
TG2 default traffic duration	This option allows adjusting the pattern length of default (compile-time) traffic.
TG2 Configuration Interface Mode	Specifies the connectivity of an Avalon slave interface for use by the TG Configuration Toolkit or user core logic. If you set this parameter to **Export**, an Avalon slave interface named `tg_cfg` is exported from the IP. If you select **JTAG**, a JTAG Avalon Master Endpoint is connected to the configuration interface, allowing the core to be accessed by the TG Configuration Toolkit.

Table 243. Group: Diagnostics / Performance

Display Name	Description
Enable Efficiency Monitor	Adds an Efficiency Monitor component to the Avalon-MM interface of the memory controller, allowing you to view efficiency statistics of the interface. You can access the efficiency statistics using the EMIF Debug Toolkit. (Identifier: DIAG_DDR4_EFFICIENCY_MONITOR)
Use Efficiency Monitor with Unified Toolkit	Specifies the efficiency monitor version to be used. The new efficiency monitor works with the Unified Toolkit.

Table 244. Group: Diagnostics / Miscellaneous

Display Name	Description
Export PLL lock signal	Specifies whether to export the pll_locked signal at the IP top-level to indicate status of PLL. (Identifier: DIAG_EXPORT_PLL_LOCKED)
Export Address/Command parity error indicator	Specifies whether to export the ac_parity_err interface at the IP top level to indicate if a parity error was detected on the Address/Command bus by the memory, causing ALERT_N to toggle. To enable this option, the ADDR/CMD parity latency option must not be set to Disabled.

7.1.9. Intel Stratix 10 EMIF IP DDR4 Parameters: Example Designs

Table 245. Group: Example Designs / Available Example Designs

Display Name	Description
Select design	Specifies the *creation of a full Quartus Prime project, instantiating an external memory interface and an example traffic generator, according to your parameterization*. After the design is created, you can specify the target device and pin location assignments, run a full compilation, verify timing closure, and test the interface on your board using the programming file created by the Quartus Prime assembler. The '**Generate Example Design**' button lets you generate simulation or synthesis file sets. (Identifier: EX_DESIGN_GUI_DDR4_SEL_DESIGN)

Table 246. Group: Example Designs / Example Design Files

Display Name	Description
Simulation	Specifies that the '**Generate Example Design**' button create all necessary file sets for simulation. Expect a short additional delay as the file set is created. *If you do not enable this parameter, simulation file sets are not created.* Instead, the output directory will contain the `ed_sim.qsys` file which holds Qsys details of the simulation example design, and a `make_sim_design.tcl` file with other corresponding `tcl` files. You can run `make_sim_design.tcl` from a command line to generate the simulation example design. The generated example designs for various simulators are **stored in the /sim sub-directory**. (Identifier: EX_DESIGN_GUI_DDR4_GEN_SIM)
Synthesis	Specifies that the '**Generate Example Design**' button create all necessary file sets for synthesis. Expect a short additional delay as the file set is created. *If you do not enable this parameter, synthesis file sets are not created.* Instead, the output directory will contain the `ed_synth.qsys` file which holds Qsys details of the synthesis example design, and a `make_qii_design.tcl` script with other corresponding `tcl` files. You can run `make_qii_design.tcl` from a command line to generate the synthesis example design. The generated example design is **stored in the /qii sub-directory**. (Identifier: EX_DESIGN_GUI_DDR4_GEN_SYNTH)

Table 247. Group: Example Designs / Generated HDL Format

Display Name	Description
Simulation HDL format	This option lets you choose the format of HDL in which generated simulation files are created. (Identifier: EX_DESIGN_GUI_DDR4_HDL_FORMAT)

Table 248. Group: Example Designs / Target Development Kit

Display Name	Description
Select board	Specifies that *when you select a development kit with a memory module, the generated example design contains all settings and fixed pin assignments to run on the selected board. You must select a development kit preset to generate a working example design for the specified development kit*. Any IP settings not applied directly from a development kit preset will not have guaranteed results when testing the development kit. To exclude hardware support of the example design, select '**none**' from the '**Select board**' pull down menu. When you apply a development kit preset, all IP parameters are automatically set appropriately to match the selected preset. If you want to save your current settings, you should do so before you apply the preset. You can save your settings under a different name using **File->Save as**. (Identifier: EX_DESIGN_GUI_DDR4_TARGET_DEV_KIT)

7.2. Register Map IP-XACT Support for Intel Stratix 10 EMIF DDR4 IP

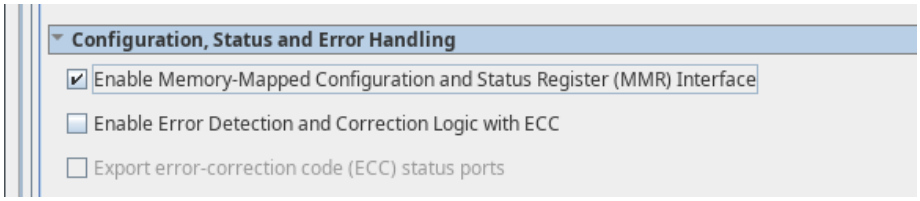
IP-XACT is an XML format that describes reusable intellectual property (IP).

When you generate an EMIF DDR4 design example from the Intel Quartus Prime software version 21.3 or later, the generated .ip file includes IP-XACT information for that IP. The generated IP-XACT information includes the register map for the DDR4 IP, Traffic Generator 2.0 (TG2), and Efficiency Monitor. The IP-XACT information for Intel Stratix 10 EMIF IP Memory-Mapped Registers (MMR) and Efficiency Monitor is included in `ed_synth_emif_fm_0.ip`, and the IP-XACT information for Traffic Generator 2.0 is included in `ed_synth_tg.ip`.

IP-XACT information is generated only with the design example. To enable generation of the IP-XACT information, follow these steps:

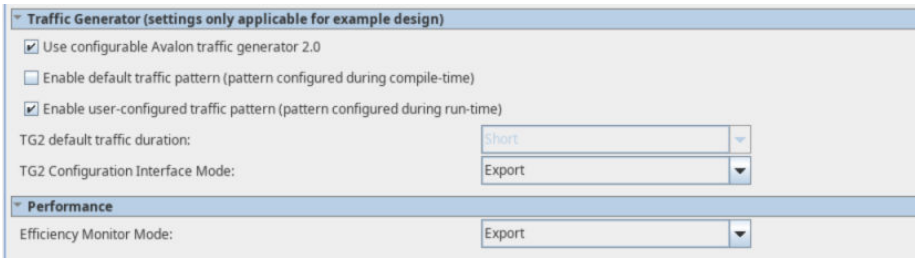
1. To enable generation of the IP-XACT information for Intel Stratix 10 IP MMR, check the **Enable Memory-Mapped Configuration and Status Register (MMR) Interface** box on the **Controller** tab of the parameter editor.

Figure 60. Enabling IP-XACT Generation for MMR Registers



2. To enable generation of IP-XACT information for TG2, check the **Use configurable Avalon traffic generator 2.0** box and set **TG2 Configuration Interface Mode** to *Export* on the **Diagnostics** tab of the parameter editor. To include IP-XACT information for the Efficiency Monitor, set the **Efficiency Monitor Mode** to *Export*.

Figure 61. Enabling IP-XACT Generation for TG2 and Efficiency Monitor



For information on the registers available for the Intel Stratix 10 EMIF IP, refer to *Intel Stratix 10 EMIF IP Memory Mapped Register (MMR) Tables* in the *End-User Signals* chapter.

For information on the registers available for Traffic Generator 2.0, refer to *Configuration and Status Registers* in the *Debugging* chapter.

For information on the registers available for the Efficiency Monitor, refer to *Control and Status Registers* in the *Debugging* chapter.

7.3. Board Skew Equations

The following table presents the underlying equations for the board skew parameters.

7.3.1. Equations for DDR4 Board Skew Parameters

Table 249. Board Skew Parameter Equations

Parameter	Description/Equation
Maximum CK delay to DIMM/device	The delay of the longest CK trace from the FPGA to any DIMM/device. $\max_r\left[\max_n\left(CK_{n\_r}PathDelay\right)\right]$ Where *n* is the number of memory clock and r is the number rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 pairs of memory clocks in each rank DIMM, the maximum CK delay is expressed by the following equation: $\max(CK_1PathDelayrank1,\ CK_2PathDelayrank1, CK_1PathDelayrank2,\ CK_2PathDelayrank2)$
Maximum DQS delay to DIMM/device	The delay of the longest DQS trace from the FPGA to the DIMM/device. $\max_r\left[\max_n\left(DQS_{n\_r}PathDelay\right)\right]$ Where *n* is the number of DQS and r is the number of rank of DIMM/device. For example in dual-rank DIMM implementation, if there are 2 DQS in each rank DIMM, the maximum DQS delay is expressed by the following equation: $\max(DQS_1PathDelayrank1,\ DQS_2PathDelayrank1,\ DQS_1PathDelayrank2,\ DQS_2PathDelayrank2)$
Average delay difference between DQS and CK	The average delay difference between the DQS signals and the CK signal, calculated by averaging the longest and smallest DQS delay minus the CK delay. Positive values represent DQS signals that are longer than CK signals and negative values represent DQS signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the DQS signals for appropriate setup and hold margins.

continued...

Parameter	Description/Equation
	$\max_r\left[\frac{\max}{n,\ m}\right]\frac{\left\{\left(DQS_{m\_r}Delay - CK_{n\_r}Delay\right)\right\} + \min_r\left[\frac{\min}{n,\ m}\right]\left\{\left(DQS_{m\_r}Delay - CK_{n\_r}Delay\right)\right.}{2}$ Where *n* is the number of memory clock, *m* is the number of DQS, and *r* is the number of rank of DIMM/device. When using discrete components, the calculation differs slightly. Find the minimum and maximum values for (DQS-CK) over all groups and then divide by 2. Calculate the (DQS-CK) for each DQS group, by using the appropriate CLK for that group. For example, in a configuration with 5 x16 components, with each component having two DQS groups: To find the minimum and maximum, calculate the minimum and maximum of (DQS0 – CK0, DQS1 – CK0, DQS2 –CK1, DQS3 – CK1, and so forth) and then divide the result by 2.
Maximum Board skew within DQS group	The largest skew between all DQ and DM pins in a DQS group. Enter your board skew only. Package skew is calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins. $\left[\underset{Max_g}{groups}\left[\max DQ_g - \min DQ_g\right]\right]$
Maximum skew between DQS groups	The largest skew between DQS signals in different DQS groups. $\left[\underset{Max_g}{groups}\left[DQS_g\right]\right] - \left[\underset{Min_g}{groups}\left[DQS_g\right]\right]$
Maximum system skew within address/command bus	$(MaxAC - MinAC)$ The largest skew between the address and command signals. Enter combined board and package skew. In the case of a component, find the maximum address/command and minimum address/command values across all component address signals.
Average delay difference between address/command and CK	A value equal to the average of the longest and smallest address/command signal delays, minus the delay of the CK signal. The value can be positive or negative. The average delay difference between the address/command and CK is expressed by the following equation: $\frac{\sum\left(\frac{n\ =\ n}{n\ =\ 1}\right)\left[\left(\frac{LongestACPathDelay + ShortestACPathDelay}{2}\right) - CK_nPathDelay\right.}{n}$ where *n* is the number of memory clocks.
Maximum delay difference between DIMMs/devices	The largest propagation delay on DQ signals between ranks. For example, in a two-rank configuration where you place DIMMs in different slots there is also a propagation delay for DQ signals going to and coming back from the furthest DIMM compared to the nearest DIMM. This parameter is applicable only when there is more than one rank.`Maxr { max n,m [(DQn_r path delay– DQn_r+1 path delay), (DQSm_r path delay–DQSm_r+1 path delay)]}` Where *n* is the number of DQ, *m* is the number of DQS and *r* is number of rank of DIMM/device .

7.4. Pin and Resource Planning

The following topics provide guidelines on pin placement for external memory interfaces.

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

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

Once all the requirements are known for your external memory interface, you can begin planning your system.

7.4.1. Interface Pins

DQS (data strobe or data clock) and DQ (data) pins are listed for EMIF supported banks in the device pin tables and are fixed at specific locations in the device. You must adhere to these pin locations to optimize routing, minimize skew, and maximize margins. Always check the device pin table for the actual locations of the DQS and DQ pins, and the EMIF pin table for location of address and control pins.

Pin tables are available here: https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html?1.

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

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

- Maximum possible interface width in any particular device is limited by the number of DQS groups available.
- Sufficient clock networks are available to the interface PLL as required by the IP.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other address and command, and clock pin placement requirements.

Note: The greater the number of banks, the greater the skew, hence Intel recommends that you always generate a test project of your desired configuration and confirm that it meets timing.

7.4.1.1. Estimating Pin Requirements

You should use the Intel Quartus Prime software for final pin fitting. However, you can estimate whether you have enough pins for your memory interface using the EMIF Device Selector on www.altera.com, or perform the following steps:

1. Determine how many read/write data pins are associated per data strobe or clock pair.
2. Calculate the number of other memory interface pins needed, including any other clocks (write clock or memory system clock), address, command, and RZQ. Refer to the External Memory Interface Pin Table to determine necessary Address/Command/Clock pins based on your desired configuration.
3. Calculate the total number of I/O banks required to implement the memory interface, given that an I/O bank supports up to 48 GPIO pins.

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

Related Information

Intel FPGA IP for External Memory Interfaces - Support Center

7.4.1.2. DIMM Options

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

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

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

Table 250. UDIMM, RDIMM, and LRDIMM Pin Options for DDR4

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)	LRDIMM Pins (Dual Rank)	LRDIMM Pins (Quad Rank)
Data	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}	72 bit DQ[71:0]= {CB[7:0], DQ[63:0]}
Data Mask	DM#/ DBI#[8:0] (1)	DM#/ DBI#[8:0](1)	DM#/ DBI#[8:0](1)	DM#/ DBI#[8:0](1)	—	—
Data Strobe	x8: DQS[8:0] and DQS#[8:0]	x8: DQS[8:0] and DQS#[8:0]	x8: DQS[8:0] and DQS#[8:0] x4: DQS[17:0] and DQS#[17:0]	x8: DQS[8:0] and DQS#[8:0] x4: DQS[17:0] and DQS#[17:0]	x4: DQS[17:0] and DQS#[17:0]	x4: DQS[17:0] and DQS#[17:0]
Address	BA[1:0], BG[1:0], A[16:0] - 4GB: A[14:0] 8GB: A[15:0] 16GB: A[16:0] (2)	BA[1:0], BG[1:0], A[16:0] - 8GB: A[14:0] 16GB: A[15:0] 32GB: A[16:0] (2)	BA[1:0], BG[1:0], x8: A[16:0] - 4GB: A[14:0] 8GB: A[15:0] 16GB: A[16:0] (2) 32GB: A[17:0] (3)	BA[1:0], BG[1:0],x8: A[16:0] x4: A[17:0] - 8GB: A[14:0] 16GB: A[15:0] 32GB: A[16:0] (2) 64GB: A[17:0] (3)	BA[1:0], BG[1:0], A[17:0] - 16GB: A[15:0] 32GB: A[16:0] (2) 64GB: A[17:0] (3)	BA[1:0], BG[1:0], A[17:0] - 32GB: A[15:0] 64GB: A[16:0] (2) 128GB: A[17:0] (3)

continued...

Pins	UDIMM Pins (Single Rank)	UDIMM Pins (Dual Rank)	RDIMM Pins (Single Rank)	RDIMM Pins (Dual Rank)	LRDIMM Pins (Dual Rank)	LRDIMM Pins (Quad Rank)
Clock	CK0/CK0#	CK0/CK0#, CK1/CK1#	CK0/CK0#	CK0/CK0#	CK0/CK0#	CK0/CK0#
Command	ODT, CS#, CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT[1:0], CS#[1:0], CKE[1:0], ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT, CS#, CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT[1:0], CS#[1:0], CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT, CS#[1:0], CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14	ODT, CS#[3:0], CKE, ACT#, RAS#/A16, CAS#/A15, WE#/A14
Parity	PAR, ALERT#	PAR, ALERT#	PAR, ALERT#	PAR, ALERT#	PAR, ALERT#	PAR, ALERT#
Other Pins	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#	SA[2:0], SDA, SCL, EVENT#, RESET#

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

7.4.1.3. Maximum Number of Interfaces

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

Unless otherwise noted, the calculation for the maximum number of interfaces is based on independent interfaces where the address or command pins are not shared.

Note: You may need to share PLL clock outputs depending on your clock network usage.

For interface information for Intel Stratix 10, consult the EMIF Device Selector on www.altera.com.

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

Related Information

- Intel FPGA IP for External Memory Interfaces - Support Center
- Intel Stratix 10 EMIF Architecture: PLL Reference Clock Networks on page 21
- External Memory Interface Device Selector
- Intel Quartus Prime Pro Edition Handbook

7.4.2. FPGA Resources

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

7.4.2.1. OCT

You require one OCT calibration block if you are using an FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design. You can select any available OCT calibration block—it need not be within the same bank or side of the device as the memory interface pins. The only requirement is that the I/O bank where you place the OCT calibration block must use the same V_{CCIO} voltage as the memory interface.

The OCT calibration block uses a single R_{ZQ} pin. The R_{ZQ} pin in Intel Stratix 10 devices can be used as a general purpose I/O pin when it is not used to support OCT, provided the signal conforms to the bank voltage requirements.

7.4.2.2. PLL

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

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

7.4.3. Pin Guidelines for Intel Stratix 10 EMIF IP

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

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

- The numeric component of the `Bank Number` value identifies the I/O column, while the letter represents the I/O bank.
- The `Index within I/O Bank` value falls within one of the following ranges: 0 to 11, 12 to 23, 24 to 35, or 36 to 47, and represents I/O lanes 1, 2, 3, and 4, respectively.
- To determine if I/O banks are adjacent, you can refer to the I/O Pin Counts tables located in the *Intel Stratix 10 General Purpose I/O User Guide*. You can always assume I/O banks are adjacent within an I/O column except in the following conditions:
  - When an I/O bank is not bonded out on the package (contains the '-' symbol in the I/O table).
  - An I/O bank does not contain 48 pins, indicating it is only partially bonded out.
- The pairing pin for an I/O pin is located in the same I/O bank. You can identify the pairing pin by adding one to its `Index within I/O Bank` number (if it is an even number), or by subtracting one from its `Index within I/O Bank` number (if it is an odd number).

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

7.4.3.1. General Guidelines

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

If you are using the hard memory controller, you should employ the relative pin locations defined in the *<variation_name>*`/altera_emif_arch_nd_`*version number*`/<`*synth*`|`*sim*`>/<`*variation_name*`>_altera_emif_arch_nd_`*version number*`_<`*unique ID*`>_readme.txt` file, which is generated with your IP.

Note:

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

Observe the following general guidelines when placing pins for your Intel Stratix 10 external memory interface:

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

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

   Note: The pin-out scheme is a hardware requirement that you must follow, and can vary according to the topology of the memory device. Some schemes require three lanes to implement address and command pins, while others require four lanes. To determine which scheme to follow, refer to the messages window during parameterization of your IP, or to the `<variation_name>/altera_emif_arch_nd_<version>/<synth|sim>/<variation_name>_altera_emif_arch_nd_<version>_<unique ID>_readme.txt` file after you have generated your IP.
7. An unused I/O lane in the address and command bank can serve to implement a data group, such as a x8 DQS group. The data group must be from the same controller as the address and command signals.
8. An I/O lane must not be used by both address and command pins and data pins.
9. Place read data groups according to the DQS grouping in the pin table and Pin Planner. Read data strobes (such as DQS and DQS#) or read clocks (such as CQ and CQ# / QK and QK#) must reside at physical pins capable of functioning as DQS/CQ and DQSn/CQn for a specific read data group size. You must place the associated read data pins (such as DQ and Q), within the same group.

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

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

10. You can implement two x4 DQS groups with a single I/O lane. The pin table specifies which pins within an I/O lane can be used for the two pairs of DQS and DQS# signals. In addition, for x4 DQS groups you must observe the following rules:
    - There must be an even number of x4 groups in an external memory interface.
    - DQS group 0 and DQS group 1 must be placed in the same I/O lane. Similarly, DQS group 2 and group 3 must be in the same I/O lane. Generally, DQS group *X* and DQS group *X*+1 must be in the same I/O lane, where *X* is an even number.
    - When placing DQ pins in x4 mode, it is important to stay within an I/O lane when swapping pin locations. In other words, you may swap DQ pins within a given DQS group or across an adjacent DQS group, so long as you are within the same I/O lane. The following table illustrates an example, where DATA_A and DATA_B are swap groups, meaning that any pin in that index can move within that range of pins.

Index Within Lane	DQS x4 Locations
11	DATA_B[3:0]
10	DATA_B[3:0]
9	DQS_Bn
8	DQS_Bp
7	DATA_B[3:0]
6	DATA_B[3:0]
5	DQS_An
4	DQS_Ap
3	DATA_A[3:0]
2	DATA_A[3:0]
1	DATA_A[3:0]
0	DATA_A[3:0]

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

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

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

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

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

14. Be aware that for DDR4 interfaces clocked at 1333 MHz, total I/O bank usage is limited as follows:

Package	Total I/O 48 banks	Maximum number of I/O 48 banks that can be used for 1333 MHz	Remaining I/O 48 bank usage for EMIF or general-purpose I/O
1760	14	12	Do not use.
2397B	14	12	Do not use.
2912E	24	20	Do not use.

Note:

1. x4 mode does not support DM/DBI, or Intel Stratix 10 EMIF IP for HPS.
2. If you are using an Intel Stratix 10 EMIF IP-based RLDRAM 3 external memory interface, you should ensure that all the pins in a DQS group (that is, DQ, DM, DK, and QK) are placed in the same I/O bank. This requirement facilitates timing closure and is necessary for successful compilation of your design.

I/O Banks Selection

- For each memory interface, select adjacent I/O banks. To determine whether I/O banks are adjacent, refer to the I/O Pin Counts tables located in the *Intel Stratix 10 General Purpose I/O User Guide*. You can always assume I/O banks are adjacent within an I/O column except in the following conditions:
  - When an I/O bank is not bonded out on the package (contains the '-' symbol in the I/O table).
  - An I/O bank does not contain 48 pins, indicating that it is only partially bonded out.
- A memory interface can only span across I/O banks in the same I/O column.
- The number of I/O banks that you require depends on the memory interface width.
- In some device packages, the number of I/O pins in some LVDS I/O banks is less than 48 pins.

Address/Command Pins Location

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

CK Pins Assignment

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

- If the number of I/O banks is odd, assign one CK pin to the middle I/O bank.
- If the number of I/O banks is even, assign the CK pin to either of the middle two I/O banks.

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

PLL Reference Clock Pin Placement

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

- If you are sharing the PLL reference clock pin between several interfaces, the I/O banks must be adjacent. (That is, the banks must contain the same column number and letter before or after the respective I/O bank letter.)

The Intel Stratix 10 external memory interface IP does not support PLL cascading.

RZQ Pin Placement

You may place the R_{ZQ} pin in any I/O bank in an I/O column with the correct V_{CCIO} and V_{CCPT} for the memory interface I/O standard in use. However, the recommended location is in the address/command I/O bank, for greater flexibility during debug if a narrower interface project is required for testing.

DQ and DQS Pins Assignment

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

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

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

Sharing an I/O Bank Across Multiple Interfaces

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

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

7.4.3.2. x4 DIMM Implementation

DIMMS using a x4 DQS configuration require remapping of the DQS signals to achieve compatibility between the EMIF IP and the JEDEC standard DIMM socket connections.

The necessary remapping is shown in the table below. You can implement this DQS remapping in either RTL logic or in your schematic wiring connections.

Table 251. Mapping of DQS Signals Between DIMM and the EMIF IP

DIMM			Intel Quartus Prime EMIF IP	
DQS0	DQ[3:0]		DQS0	DQ[3:0]
DQS9	DQ[7:4]		DQS1	DQ[7:4]
DQS1	DQ[11:8]		DQS2	DQ[11:8]
DQS10	DQ[15:12]		DQS3	DQ[15:12]
DQS2	DQ[19:16]		DQS4	DQ[19:16]
DQS11	DQ[23:20]		DQS5	DQ[23:20]
DQS3	DQ[27:24]		DQS6	DQ[27:24]
DQS12	DQ[31:28]		DQS7	DQ[31:28]
DQS4	DQ[35:32]		DQS8	DQ[35:32]
DQS13	DQ[39:36]		DQS9	DQ[39:36]
DQS5	DQ[43:40]		DQS10	DQ[43:40]
DQS14	DQ[47:44]		DQS11	DQ[47:44]
DQS6	DQ[51:48]		DQS12	DQ[51:48]
DQS15	DQ[55:52]		DQS13	DQ[55:52]

continued...

DIMM			Intel Quartus Prime EMIF IP	
DQS7	DQ[59:56]		DQS14	DQ[59:56]
DQS16	DQ[63:60]		DQS15	DQ[63:60]
DQS8	DQ[67:64]		DQS16	DQ[67:64]
DQS17	DQ[71:68]		DQS17	DQ[71:68]

Data Bus Connection Mapping Flow

1. Connect all FPGA DQ pins accordingly to DIMM DQ pins. No remapping is required.
2. DQS/DQS*n* remapping is required either on the board schematics or in the RTL code.
3. An example mapping is shown below, with reference to the above table values:

```
FPGA (DQS0) to DIMM (DQS0)
FPGA (DQS1) to DIMM (DQS9)
FPGA (DQS2) to DIMM (DQS1)
...
FPGA (DQS16) to DIMM (DQS8)
FPGA (DQS17) to DIMM (DQS17)
```

When designing a board to support x4 DQS groups, Intel recommends that you make it compatible for x8 mode, for the following reasons:

- Provides the flexibility of x4 and x8 DIMM support.
- Allows use of x8 DQS group connectivity rules.
- Allows use of x8 timing rules for matching. Intel strongly recommends adhering to x4/x8 interoperability rules when designing a DIMM interface, even if the primary use case is to support x4 DIMMs only, because doing so facilitates debug and future migration capabilities. Regardless, the rules for length matching for two nibbles in a x4 interface must match those of the signals for a corresponding x8 interface, as the data terminations are turned on and off at the same time for both x4 DQS groups in an I/O lane. If the two x4 DQS groups were to have significantly different trace delays, it could adversely affect signal integrity.

About Pinout and Schematic Reviewing

When viewing x4 DQS mode in the Pin Planner, the 4 DQ pins do not have to be placed in the same colour-coded x4 group with the associated DQS/DQS*n* pins. This might look odd, but is not incorrect. The x4 DQS pins can be used as the strobe for any DQ pins placed within a x8 DQS group in an I/O lane.

Necessary checks to perform if the DQS groups are remapped in the RTL code

1. In the Pin Planner, view x8 DQS groups and check the following:
   a. Check that DQ[7:0] is in x8 group, DQ[15:8] is in another DQS group, and so forth.
   b. Check that DSQ0 and DQS9 are in the DQS group with DQ[7:0], DQS1 and DQS10 are in the DQS group with DQ[15:8], and so forth. This is the *DIMM* numbering convention column shown in the table at the beginning of this topic.
2. In the Pin Planner, view x4 DQS groups and check the following:

a. Check that all the DQS signals are on pins marked S and Sbar.

3. On the schematic, check the following DIMM connections:
   a. Check that DQS*x* on the DIMM maps to the DQS*x* on the FPGA pinout (for values of *x* from 0 to 17).
   b. Check that DQ*y* on the DIMM maps to the DQ*y* on the FPGA pinout. Note that there is scope for swapping pins within the x4/x8 DQS group to optimize the PCB layout.

Necessary checks to perform if the DQS groups are remapped on the schematic

1. In the Pin Planner, view x8 DQS groups and check the following:
   a. Check that DQ[7:0] is in x8 group, DQ[15:8] is in another DQS group, and so forth.
   b. Check that DSQ0 and DQS1 are in the DQS group with DQ[7:0], DQS2 and DQS3 are in the DQS group with DQ[15:8], and so forth. This is the *Intel Quartus Prime EMIF IP* mapping shown in the table at the beginning of this topic.
2. In the Pin Planner, view x4 DQS groups and check the following:
   a. Check that all the DQS signals are on pins marked S and Sbar.
3. On the schematic, check the following DIMM connections:
   a. Referring to the table above, check that DQS has the remapping between the FPGA (Intel Quartus Prime EMIF IP) and DIMM pinout (*DIMM*).
   b. Check that DQ*y* on the DIMM maps to the DQ*y* on the FPGA pinout. Note that there is scope for swapping pins within the x4/x8 DQS group to optimize the PCB layout.

7.4.3.3. Command and Address Signals

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

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

Although DDR4 operates in fundamentally the same way as other SDRAM, there are no longer dedicated pins for RAS#, CAS#, and WE#, as those are now shared with higher-order address pins. DDR4 still has CS#, CKE, ODT, and RESET# pins, similar to DDR3. DDR4 introduces some additional pins, including the ACT# (activate) pin and BG (bank group) pins. Depending on the memory format and the functions enabled, the following pins might also exist in DDR4: PAR (address command parity) pin and the ALERT# pin (1.2V I/O standard).

7.4.3.4. Clock Signals

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

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

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

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

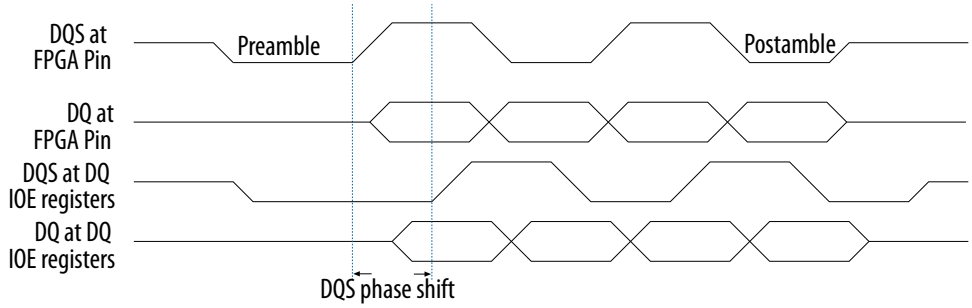
7.4.3.5. Data, Data Strobes, DM/DBI, and Optional ECC Signals

DDR3 and DDR4 SDRAM use bidirectional differential data strobes. Differential DQS operation enables improved system timing due to reduced crosstalk and less simultaneous switching noise on the strobe output drivers. The DQ pins are also bidirectional.

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

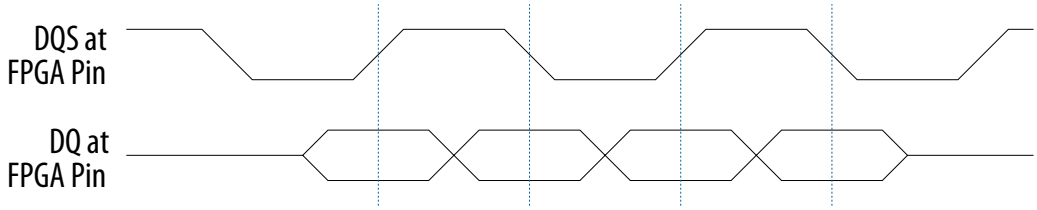
The DQ signals are edge-aligned with the DQS signal during a read from the memory and are center-aligned with the DQS signal during a write to the memory. The memory controller shifts the DQ signals by –90 degrees during a write operation to center align the DQ and DQS signals. The PHY IP delays the DQS signal during a read, so that the DQ and DQS signals are center aligned at the capture register. Intel devices use a phase-locked loop (PLL) to center-align the DQS signal with respect to the DQ signals during writes and Intel devices use dedicated DQS phase-shift circuitry to shift the incoming DQS signal during reads. The following figure shows an example where the DQS signal is shifted by 90 degrees for a read from the DDR3 SDRAM.

Figure 62. Edge-aligned DQ and DQS Relationship During a SDRAM Read in Burst-of-Four Mode



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

Figure 63. DQ and DQS Relationship During a SDRAM Write in Burst-of-Four Mode



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

The DQS signal is generated on the positive edge of the system clock to meet the t_{DQSS} requirement. DQ and DM signals use a clock shifted –90 degrees from the system clock, so that the DQS edges are centered on the DQ or DM signals when they arrive at the DDR3 SDRAM. The DQS, DQ, and DM board trace lengths need to be tightly matched (within 20 ps).

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

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

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

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

7.4.3.6. `alert_n` Pin Termination Recommendation

The `alert_n` signal requires an external pull-up resistor to 1.2V using a typical pull-up resistor value of 10,000 ohms.

7.4.4. Resource Sharing Guidelines (Multiple Interfaces)

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

PLL Reference Clock Pin

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

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

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

Note: You can place the `pll_ref_clk` pin in the address and command I/O bank or in a data I/O bank, there is no impact on timing. However, for greatest flexibility during debug (such as when creating designs with narrower interfaces), the recommended placement is in the address and command I/O bank.

Core Clock Network

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

Observe the following guidelines for sharing the core clock network:

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

I/O Bank

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

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

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

Hard Nios Processor

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

7.5. DDR4 Board Design Guidelines

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

The following areas are discussed:

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

I/O Standards

DDR4 SDRAM interface signals use one of the following JEDEC I/O signaling standards:

- SSTL-12—for address and command pins.
- POD-12—for DQ, DQS, and DBIn.

You do not have to assign the I/O standard to each pin, as that is done automatically by the IP during generation.

Termination Schemes

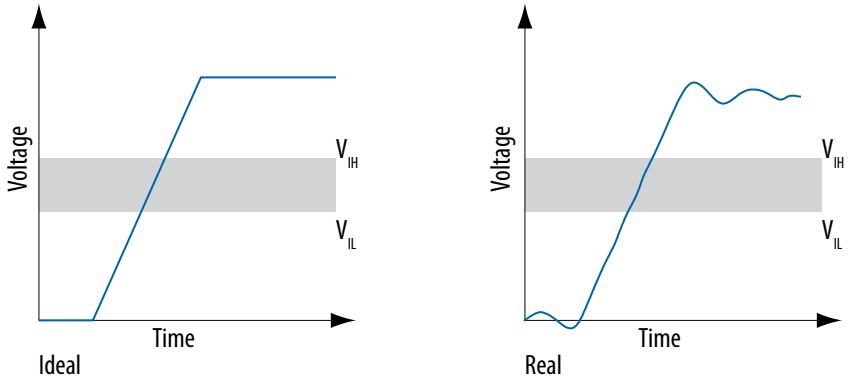
It is important to understand the trade-offs between different types of termination schemes, the effects of output drive strengths, and different loading types, so that you can swiftly navigate through the multiple combinations and choose the best possible settings for your designs.

The following key factors affect signal quality at the receiver:

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

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

Figure 64. Ideal and Real Signal at the Receiver



Related Information

JEDEC.org

7.5.1. Terminations and Slew Rates with Intel Stratix 10 Devices

The following topics describe termination and slew rate considerations for Intel Stratix 10 devices.

7.5.1.1. Dynamic On-Chip Termination (OCT) in Intel Stratix 10 Devices

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

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

Check the FPGA I/O tab of the parameter editor to determine the I/O standards and termination values supported for data, address and command, and memory clock signals.

Related Information

Choosing Terminations on Intel Stratix 10 Devices on page 178

7.5.1.2. Dynamic On-Die Termination (ODT) in DDR4

In DDR4, in addition to the Rtt_nom and Rtt_wr values, which are applied during read and write respectively, a third option called Rtt_park is available. When Rtt_park is enabled, a selected termination value is set in the DRAM when ODT is driven low.

Rtt_nom and Rtt_wr work the same as in DDR3, which is described in *Dynamic ODT for DDR3*.

Refer to the DDR4 JEDEC specification or your memory vendor data sheet for details about available termination values and functional description for dynamic ODT in DDR4 devices.

For DDR4 LRDIMM, if SPD byte 152 calls for different values of `Rtt_Park` to be used for package ranks 0 and 1 versus package ranks 2 and 3, set the value to the larger of the two impedance settings.

7.5.1.3. Choosing Terminations on Intel Stratix 10 Devices

To determine optimal on-chip termination (OCT) and on-die termination (ODT) values for best signal integrity, you should simulate your memory interface in HyperLynx or a similar tool.

If the optimal OCT and ODT termination values as determined by simulation are not available in the list of available values in the parameter editor, select the closest available termination values for OCT and ODT.

For information about available ODT choices, refer to your memory vendor data sheet.

Related Information

Dynamic On-Chip Termination (OCT) in Intel Stratix 10 Devices on page 178

7.5.1.4. On-Chip Termination Recommendations for Intel Stratix 10 Devices

- A value of 34 to 40 ohms is a good starting point for output mode drive strength.
- Input mode (parallel termination) for Data and Data Strobe signals: A value of 40 or 60 ohms is a good starting point for FPGA side input termination.

7.5.1.5. Slew Rates

For optimum timing margins and best signal integrity for the address, command, and memory clock signals, you should generally use fast slew rates and external terminations.

In board simulation, fast slew rates may show a perceived signal integrity problem, such as reflections or a nonmonotonic waveform in the SSTL I/O switching region. Such indications may cause you to consider using slow slew rate options for either the address and command signals or the memory clock, or both.

If you set the **FPGA I/O tab parameter options ➤ Address/Command ➤ Slew Rate** and **Memory Clock ➤ Slew Rate** parameters to different values, a warning message appears: .

```
Warning: .emif_0: When the address/command signals and the memory clock signals
do not use the same slew rate setting, signals using the "Slow" setting are
delayed relative to signals using "Fast" setting. For accurate timing analysis,
you must perform I/O simulation and manually include the delay as board skew.
To avoid the issue, use the same slew rate setting for both address/command
signals and memory clock signals whenever possible.
```

Note: The warning message applies only to board-level simulation, and does not require any delay adjustments in the PCB design or Board tab parameter settings.

Due to limitations of the IBIS model correlation tolerance and the accuracy of the board simulation model, it is possible for signal integrity problems to appear when using fast slew rate during simulation but not occur during operation on hardware. If you observe a signal integrity problem during simulation with a fast slew rate, use an oscilloscope to view the signal at that point in hardware, to verify whether the problem exists on hardware, or only in simulation.

If the signal integrity problem exists on hardware as well as in simulation, using different slew rates for the address and command signals and the clock remains a valid approach, and the address and command calibration stage will help to improve the address and command to clock setup and hold time margins.

7.5.2. Channel Signal Integrity Measurement

As external memory interface data rates increase, so does the importance of proper channel signal integrity measurement.By measuring the actual channel loss during the layout process and including that data in your parameterization, a realistic assessment of margins is achieved.

7.5.2.1. Importance of Accurate Channel Signal Integrity Information

Default values for channel loss (or eye reduction) can be used when calculating timing margins, however those default values may not accurately reflect the channel loss in your system.If the channel loss in your system is different than the default values, the calculated timing margins vary accordingly.

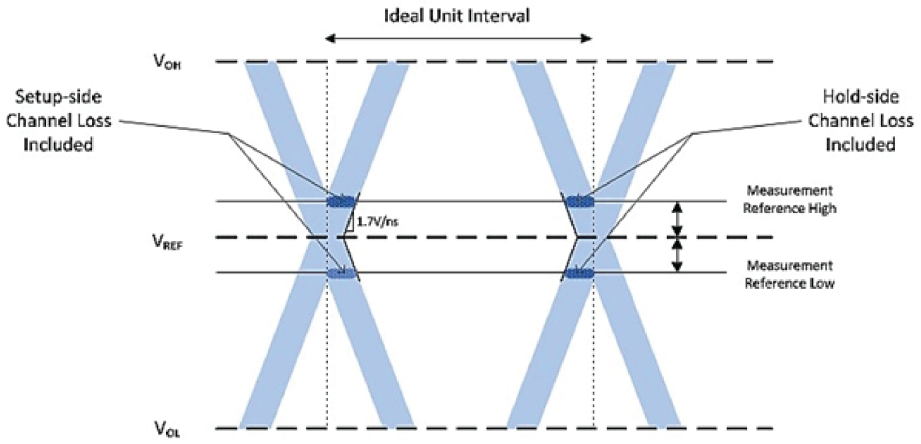
If your actual channel loss is greater than the default channel loss, and if you rely on default values, the available timing margins for the entire system are lower than the values calculated during compilation. By relying on default values that do not accurately reflect your system, you may be lead to believe that you have good timing margin, while in reality, your design may require changes to achieve good channel signal integrity.

7.5.2.2. Understanding Channel Signal Integrity Measurement

To measure channel signal integrity you need to measure the channel loss for various signals.For a particular signal or signal trace, channel loss is defined as loss of the eye width at +/- V_{IH}(ac and dc) +/- V_{IL}(ac and dc). V_{IH}/V_{IL} above or below V_{REF} is used to align with various requirements of the timing model for memory interfaces.

The example below shows a reference eye diagram where the channel loss on the setup- or leading-side of the eye is equal to the channel loss on the hold- or lagging-side of the eye; however, it does not necessarily have to be that way. Because the calibrating PHY calibrates to the center of the read and write eye, the Board Settings tab has parameters for the total extra channel loss for Write DQ and Read DQ. For address and command signals which are not-calibrated, the Board Settings tab allows you to enter setup- and hold-side channel losses that are not equal, allowing the Intel Quartus Prime software to place the clock statically within the center of the address and command eye.

Figure 65. Equal Setup and Hold-side Losses



7.5.2.3. How to Enter Calculated Channel Signal Integrity Values

You should enter calculated channel loss values in the **Channel Signal Integrity** section of the **Board** (or **Board Timing**) tab of the parameter editor.

For Intel Stratix 10 external memory interfaces, the default channel loss displayed in the parameter editor is based on the selected configuration (different values for single rank versus dual rank), and on internal Intel reference boards. You should replace the default value with the value that you calculate.

7.5.2.4. Guidelines for Calculating DDR4 Channel Signal Integrity

Address and Command ISI and Crosstalk

Simulate the address/command and control signals and capture eye at the DRAM pins, using the memory clock as the trigger for the memory interface's address/command and control signals. Measure the setup and hold channel losses at the voltage thresholds mentioned in the memory vendor's data sheet. For optimal address/command signal integrity, you should simulate both slow and fast slew rate settings.

Address and command channel loss = Measured loss on the setup side + measured loss on the hold side.

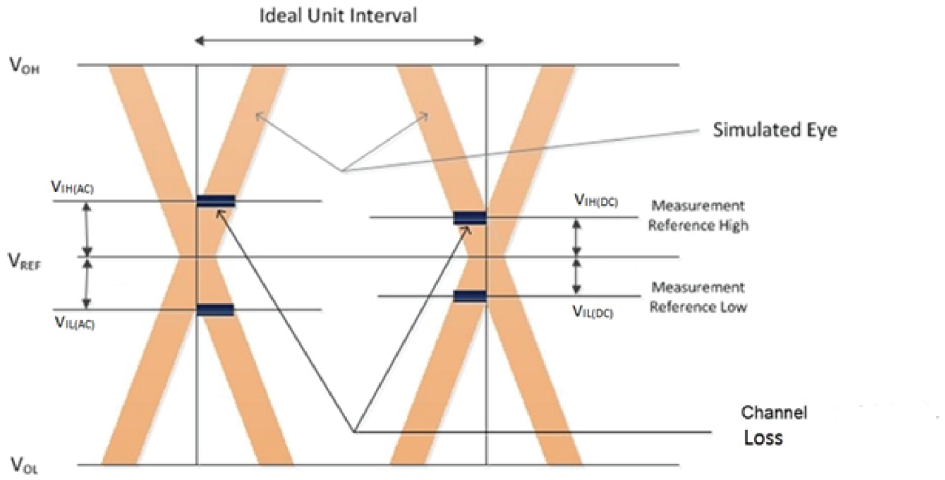
$V_{REF} = V_{DD}/2 = 0.60$ V for address/command for DDR4.

You should select the V_{IH} and V_{IL} voltage levels appropriately for the DDR4 memory device that you are using. Check with your memory vendor for the correct voltage levels, as the levels may vary for different speed grades of device.

The following figure illustrates a DDR4-1200 example, where $V_{IH(AC)}$/ $V_{IL(AC)}$ is +/- 100 mV and $V_{IH(DC)}$/ $V_{IL(DC)}$ is +/- 45 mV.

Select the $V_{IH(AC)}$, $V_{IL(AC)}$, $V_{IH(DC)}$, and $V_{IL(DC)}$for the speed grade of DDR4 memory device from the memory vendor's data sheet.

Figure 66.



Write DQ ISI and Crosstalk

Simulate the write DQ signals and capture eye at the DRAM pins, using DQ Strobe (DQS) as a trigger for the DQ signals of the memory interface simulation. Measure the setup and hold channel losses at the V_{IH} and V_{IL} mentioned in the memory vendor's data sheet

Write Channel Loss = Measured Loss on the Setup side + Measured Loss on the Hold side.

or

Write Channel Loss = UI – (Eye opening at V_{IH} or V_{IL}).

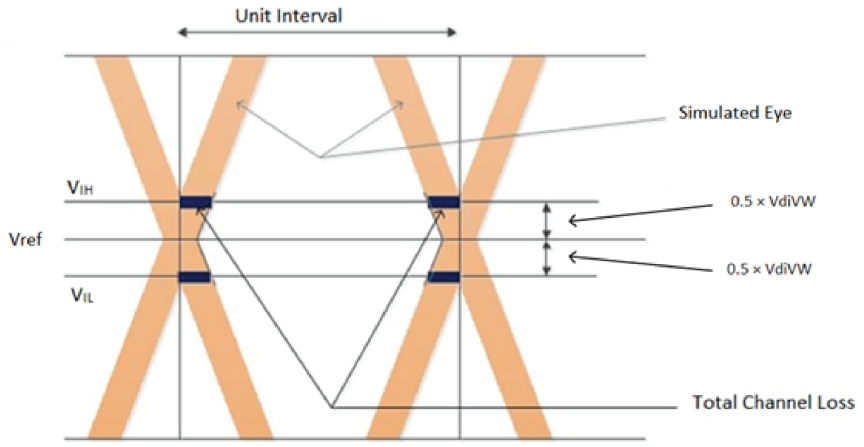
V_{REF} = Voltage level where the eye opening is highest.

V_{IH} = V_{REF} + (0.5 x VdiVW).

V_{IL} = V_{REF} - (0.5 x VdiVW).

Where VdiVW varies by frequency of operation; you can find the VdiVW value in your memory vendor's data sheet.

Figure 67.



Read DQ ISI and Crosstalk

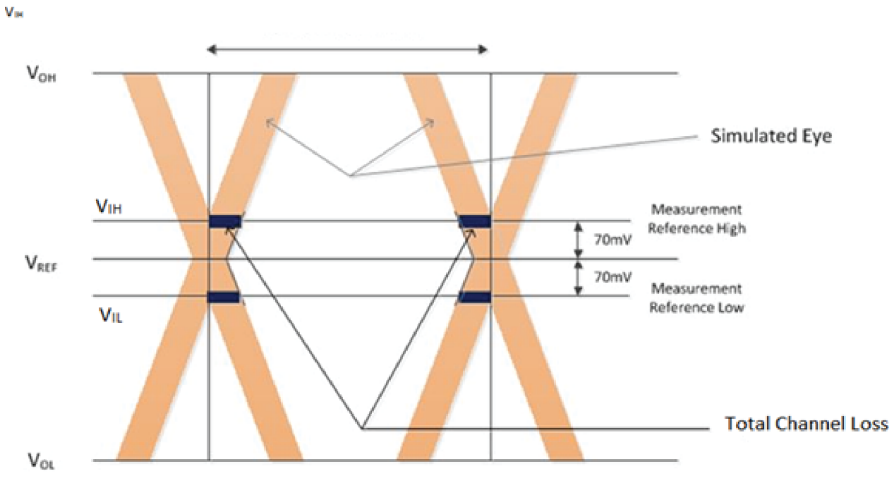
Simulate read DQ signals and capture eye at the FPGA die. Do not measure at the pin, because you might see unwanted reflections that could create a false representation of the eye opening at the input buffer of the FPGA. Use DQ Strobe (DQS) as a trigger for the DQ signals of your memory interface simulation. Measure the eye opening at +/- 45 mV (V_{IH}/V_{IL}) with respect to V_{REF}.

Read Channel Loss = (UI) - (Eye opening at +/- 45 mV with respect to V_{REF}.)

UI = Unit interval. For example, if you are running your interface at 800 Mhz, the effective data is 1600 Mbps, giving a unit interval of 1/1600 = 625 ps.

V_{REF} = Voltage level where the eye opening is highest.

Figure 68.

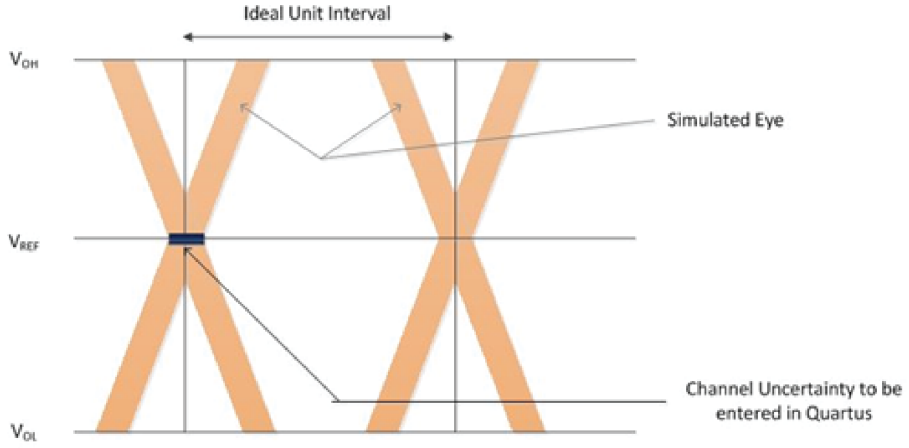


Write/Read DQS ISI and Crosstalk

Simulate write and read DQS and capture eye. Measure the uncertainty at V_{REF}.

V_{REF} = Voltage level where the eye opening is the highest.

Figure 69.

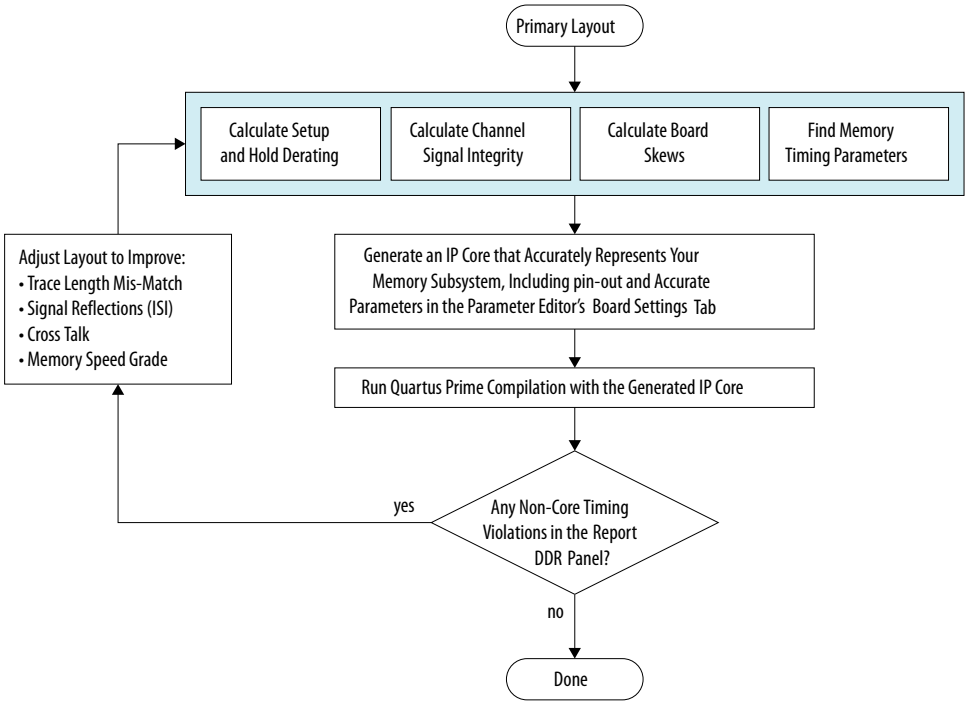


7.5.3. Layout Approach

For all practical purposes, you can regard the Timing Analyzer report on your memory interface as definitive for a given set of memory and board timing parameters.

You can find timing information under **Report DDR** in the Timing Analyzer and on the **Timing Analysis** tab in the parameter editor.

The following flowchart illustrates the recommended process to follow during the board design phase, to determine timing margin and make iterative improvements to your design.



Board Skew

For information on calculating board skew parameters, refer to *Board Skew Equations*, in this chapter.

The Board Skew Parameter Tool is an interactive tool that can help you calculate board skew parameters if you know the absolute delay values for all the memory related traces.

Memory Timing Parameters

For information on the memory timing parameters to be entered into the parameter editor, refer to the datasheet for your external memory device.

Related Information

Board Skew Parameter Tool

7.5.4. Design Layout Guidelines

The general layout guidelines in the following topic apply to DDR3 and DDR4 SDRAM interfaces.

These guidelines help you plan your board layout, but are not meant as strict rules that you must adhere to. Intel recommends that you perform your own board-level simulations to ensure that the layout you choose for your board allows you to achieve your desired performance.

For more information about how the memory manufacturers route these address and control signals on their DIMMs, refer to the Cadence PCB browser from the Cadence website, at www.cadence.com. You can find the various JEDEC example DIMM layouts on the JEDEC website, at www.jedec.org.

For assistance in calculating board skew parameters, refer to the board skew calculator tool, which you can find at the Intel website.

Note:

1. The following layout guidelines include several +/- length based rules. These length based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristic of the interface. They do not include any margin for crosstalk.
2. To ensure reliable timing closure to and from the periphery of the device, you should register signals to and from the periphery before you connect any further logic.

Intel recommends that you get accurate time base skew numbers for your design when you simulate the specific implementation.

Related Information

- JEDEC.org
- https://www.cadence.com/
- Board Skew Parameter Tool
- https://eda.sw.siemens.com/

7.5.4.1. General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Intel recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

Table 252. General Layout Guidelines

Parameter	Guidelines
Impedance	• All unused via pads must be removed, because they cause unwanted capacitance. • Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.
Decoupling Parameter	• Use 0.1 uF in 0402 size to minimize inductance • Make VTT voltage decoupling close to termination resistors • Connect decoupling caps between VTT and ground • Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin • Verify the capacitive decoupling using the Intel Power Distribution Network Design Tool
Power	• Route GND and V_{CC} as planes • Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation • Route VTT as islands or 250-mil (6.35-mm) power traces • Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces
General Routing	All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Intel recommends that signals from the same net group always be routed on the same layer. • Use 45° angles (*not* 90° corners) • Avoid T-Junctions for critical nets or clocks • Avoid T-junctions greater than 250 mils (6.35 mm) • Disallow signals across split planes • Restrict routing other signals close to system reset signals • Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks

Related Information

Power Distribution Network

7.5.4.2. Layout Guidelines

The following table lists layout guidelines.

Unless otherwise specified, the guidelines in the following table apply to the following topologies:

- DIMM—UDIMM topology
- DIMM—RDIMM topology
- DIMM—LRDIMM topology
- Not all versions of the Intel Quartus Prime software support LRDIMM.
- Discrete components laid out in UDIMM topology
- Discrete components laid out in RDIMM topology

These guidelines are recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface.

For supported frequencies and topologies, refer to the *External Memory Interface Spec Estimator* https://www.intel.com/content/www/us/en/programmable/support/support-resources/external-memory.html.

For frequencies greater than 800 MHz, when you are calculating the delay associated with a trace, you must take the FPGA package delays into consideration.

Table 253. Layout Guidelines *(1)*

Parameter	Guidelines
Decoupling Parameter	• Make VTT voltage decoupling close to the components and pull-up resistors. • Connect decoupling caps between VTT and VDD using a 0.1 uF cap for every other VTT pin. • Use a 0.1 uF cap and 0.01 uF cap for every VDDQ pin.
Maximum Trace Length	• Even though there are no hard requirements for minimum trace length, you need to simulate the trace to ensure the signal integrity. Shorter routes result in better timing. • For DIMM topology only: — Maximum trace length for all signals from FPGA to the first DIMM slot is 4.5 inches. — Maximum trace length for all signals from DIMM slot to DIMM slot is 0.425 inches. • For discrete components only: — Maximum trace length for address, command, control, and clock from FPGA to the first component must not be more than 7 inches. — Maximum trace length for DQ, DQS, DQS#, and DM from FPGA to the first component is 5 inches.
General Routing	• Route over appropriate VCC and GND planes. • Keep signal routing layers close to GND and power planes.
Spacing Guidelines	• Avoid routing two signal layers next to each other. Always make sure that the signals related to memory interface are routed between appropriate GND or power layers. • For DQ/DQS/DM traces: Maintain at least 3H spacing between the edges (air-gap) for these traces. (Where H is the vertical distance to the closest return path for that particular trace.) • For Address/Command/Control traces: Maintain at least 3H spacing between the edges (air-gap) these traces. (Where H is the vertical distance to the closest return path for that particular trace.) • For Clock traces: Maintain at least 5H spacing between two clock pair or a clock pair and any other memory interface trace. (Where H is the vertical distance to the closest return path for that particular trace.)
Clock Routing	• Route clocks on inner layers with outer-layer run lengths held to under 500 mils (12.7 mm). • Route clock signals in a daisy chain topology from the first SDRAM to the last SDRAM. The maximum length of the first SDRAM to the last SDRAM must not exceed 0.69 tCK for DDR3 and 1.5 tCK for DDR4. For different DIMM configurations, check the appropriate JEDEC specification. • These signals should maintain the following spacings: • Clocks should maintain a length-matching between clock pairs of ±5 ps. • Clocks should maintain a length-matching between positive (p) and negative (n) signals of ±2 ps, routed in parallel. • Space between different pairs should be at least two times the trace width of the differential pair to minimize loss and maximize interconnect density. • To avoid mismatched transmission line to via, Intel recommends that you use Ground Signal Signal Ground (GSSG) topology for your clock pattern—GND|CLKP|CKLN|GND. • Route all addresses and commands to match the clock signals to within ±20 ps to each discrete memory component. Refer to the following figure.

continued...

Parameter	Guidelines
Address and Command Routing	• Route address and command signals in a daisy chain topology from the first SDRAM to the last SDRAM. The maximum length of the first SDRAM to the last SDRAM must not be more than 0.69 tCK for DDR3 and 1.5 tCK for DDR4. For different DIMM configurations, check the appropriate JEDEC specifications. • UDIMMs are more susceptible to cross-talk and are generally noisier than buffered DIMMs. Therefore, route address and command signals of UDIMMs on a different layer than data signals (DQ) and data mask signals (DM) and with greater spacing. • Do not route differential clock (CK) and clock enable (CKE) signals close to address signals. • Route all addresses and commands to match the clock signals to within ±20 ps to each discrete memory component. Refer to the following figure.
DQ, DM, and DQS Routing Rules	• All the trace length matching requirements are from the FPGA package ball to the SDRAM package ball, which means you must consider trace mismatching on different DIMM raw cards. • Match in length all DQ, DQS, and DM signals within a given byte-lane group with a maximum deviation of ±10 ps. • Ensure to route all DQ, DQS, and DM signals within a given byte-lane group on the same layer to avoid layer to layer transmission velocity differences, which otherwise increase the skew within the group. • Do not count on FPGAs to deskew for more than 20 ps of DQ group skew. The skew algorithm only removes the following possible uncertainties: — Minimum and maximum die IOE skew or delay mismatch — Minimum and maximum device package skew or mismatch — Board delay mismatch of 20 ps — Memory component DQ skew mismatch — Increasing any of these four parameters runs the risk of the deskew algorithm limiting, failing to correct for the total observed system skew. If the algorithm cannot compensate without limiting the correction, timing analysis shows reduced margins. • For memory interfaces with leveling, the timing between the DQS and clock signals on each device calibrates dynamically to meet tDQSS. To make sure the skew is not too large for the leveling circuit's capability, follow these rules: — Propagation delay of clock signal must not be shorter than propagation delay of DQS signal at every device: (CKi) – DQSi > 0; 0 < *i* < number of components – 1 . For DIMMs, ensure that the CK trace is longer than the longest DQS trace at the DIMM connector. — Total skew of CLK and DQS signal between groups is less than one clock cycle: (CKi+ DQSi) max – (CKi+ DQSi) min < 1 × tCK(If you are using a DIMM topology, your delay and skew must take into consideration values for the actual DIMM.)

continued...

Parameter	Guidelines
Spacing Guidelines	• Avoid routing two signal layers next to each other. Always ensure that the signals related to the memory interface are routed between appropriate GND or power layers. • For DQ/DQS/DM traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace. • For Address/Command/Control traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace. • For Clock traces: Maintain at least 5H spacing between two clock pairs or a clock pair and any other memory interface trace, where H is the vertical distance to the closest return path for that particular trace.
Intel Quartus Prime Software Settings for Board Layout	• To perform timing analyses on board and I/O buffers, use a third-party simulation tool to simulate all timing information such as skew, ISI, crosstalk, and type the simulation result into the Board Settings tab in the parameter editor. • Do not use advanced I/O timing model (AIOT) or board trace model unless you do not have access to any third party tool. AIOT provides reasonable accuracy but tools like HyperLynx provide better results.
Notes to Table: 1. For point-to-point and DIMM interface designs, refer to the Micron website, www.micron.com.	

For DDR4 interfaces clocked at 1333 MHz, total I/O bank usage is limited as follows

Package	Total I/O 48 banks	Maximum number of I/O 48 banks that can be used for 1333 MHz	Remaining I/O 48 bank usage for EMIF or general-purpose I/O
1760	14	12	Do not use.
2397B	14	12	Do not use.
2912E	24	20	Do not use.

Related Information

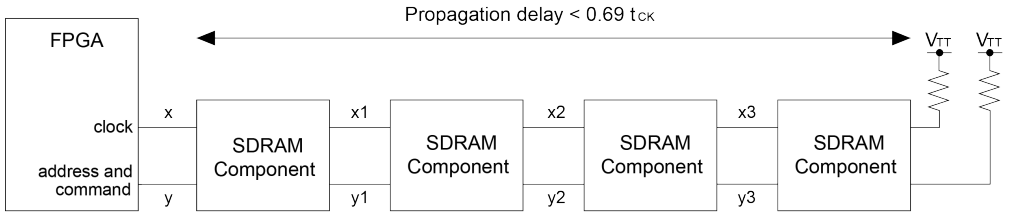
- Package Deskew on page 255
- External Memory Interface Spec Estimator
- www.micron.com

7.5.4.3. Length Matching Rules

The following topics provide guidance on length matching for different types of SDRAM signals.

Route all addresses and commands to match the clock signals to within ±20 ps to each discrete memory component. The following figure shows the component routing guidelines for address and command signals.

Figure 70. SDRAM Component Address and Command Routing Guidelines



If using discrete components:
x = y ± 20 ps
x + x1 = y + y1 ± 20 ps
x + x1 + x2 = y + y1 + y2 ± 20 ps
x + x1 + x2 + x3 = y + y1 + y2 +y3 ± 20 ps

If using a DIMM topology:
x=y +/- 20 ps

The `alert_n` signal requires an external pull-up resistor to 1.2V using a typical pull-up resistor value of 10,000 ohms.

The timing between the DQS and clock signals on each device calibrates dynamically to meet tDQSS. The following figure shows the delay requirements to align DQS and clock signals. To ensure that the skew is not too large for the leveling circuit's capability, follow these rules:

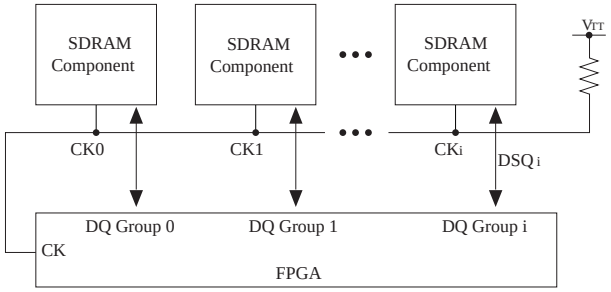
- Propagation delay of clock signal must not be shorter than propagation delay of DQS signal at every device:

```
CKi  – DQSi > 0; 0 < i < number of components – 1
```

- Total skew of CLK and DQS signal between groups is less than one clock cycle:

```
(CKi + DQSi) max – (CKi + DQSi) min < 1 × tCK
```

Figure 71. Delaying DQS Signal to Align DQS and Clock



CKi = Clock signal propagation delay to device i
DQSi = DQ/DQS signals propagation delay to group i

Clk pair matching—If you are using a DIMM (UDIMM, RDIMM, or LRDIMM) topology, match the trace lengths up to the DIMM connector. If you are using discrete components, match the lengths for all the memory components connected in the fly-by chain.

DQ group length matching—If you are using a DIMM (UDIMM, RDIMM, or LRDIMM) topology, apply the DQ group trace matching rules described in the guideline table earlier up to the DIMM connector. If you are using discrete components, match the lengths up to the respective memory components.

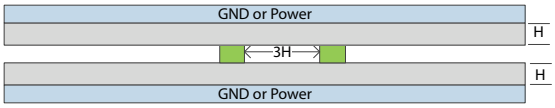
When you are using DIMMs, it is assumed that lengths are tightly matched within the DIMM itself. You should check that appropriate traces are length-matched within the DIMM.

7.5.4.4. Spacing Guidelines

This topic provides recommendations for minimum spacing between board traces for various signal traces.

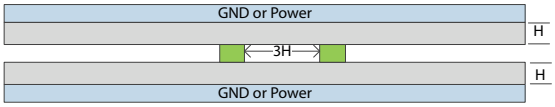
Spacing Guidelines for DQ, DQS, and DM Traces

Maintain a minimum of 3H spacing between the edges (air-gap) of these traces. (Where *H* is the vertical distance to the closest return path for that particular trace.)



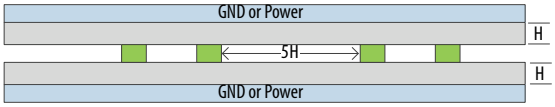
Spacing Guidelines for Address and Command and Control Traces

Maintain at least 3H spacing between the edges (air-gap) of these traces. (Where *H* is the vertical distance to the closest return path for that particular trace.)



Spacing Guidelines for Clock Traces

Maintain at least 5H spacing between two clock pair or a clock pair and any other memory interface trace. (Where *H* is the vertical distance to the closest return path for that particular trace.)



7.5.4.5. Layout Guidelines for DDR3 and DDR4 SDRAM Wide Interface (>72 bits)

The following topics discuss different ways to lay out a wider DDR3 or DDR4 SDRAM interface to the FPGA. Choose the topology based on board trace simulation and the timing budget of your system.

The EMIF IP supports up to a 144-bit wide DDR3 interface. You can use discrete components or DIMMs to implement a wide interface (any interface wider than 72 bits). Intel recommends using leveling when you implement a wide interface with DDR3 components.

When you lay out for a wider interface, all rules and constraints discussed in the previous sections still apply. The DQS, DQ, and DM signals are point-to-point, and all the same rules discussed in *Design Layout Guidelines* apply.

The main challenge for the design of the fly-by network topology for the clock, command, and address signals is to avoid signal integrity issues, and to make sure you route the DQS, DQ, and DM signals with the chosen topology.

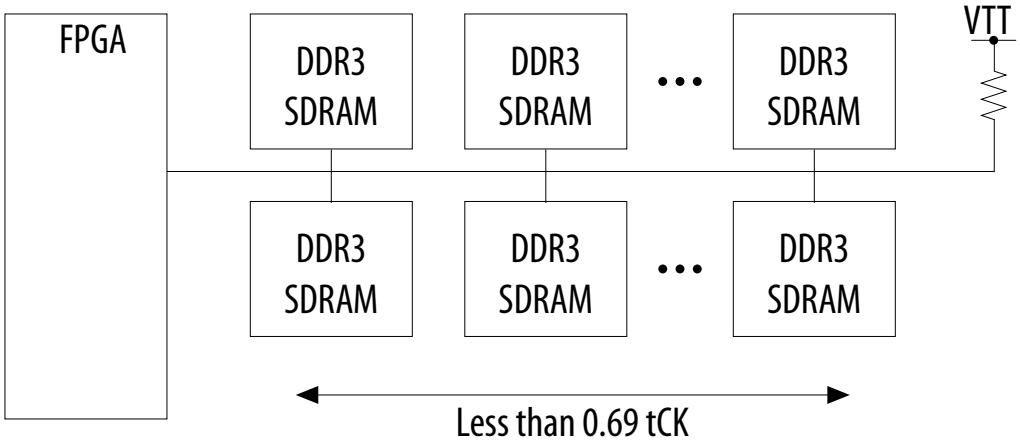
7.5.4.6. Fly-By Network Design for Clock, Command, and Address Signals

The EMIF IP requires the flight-time skew between the first SDRAM component and the last SDRAM component to be less than 0.69 tCK for memory clocks. This constraint limits the number of components you can have for each fly-by network.

If you design with discrete components, you can choose to use one or more fly-by networks for the clock, command, and address signals.

The following figure shows an example of a single fly-by network topology.

Figure 72. Single Fly-By Network Topology

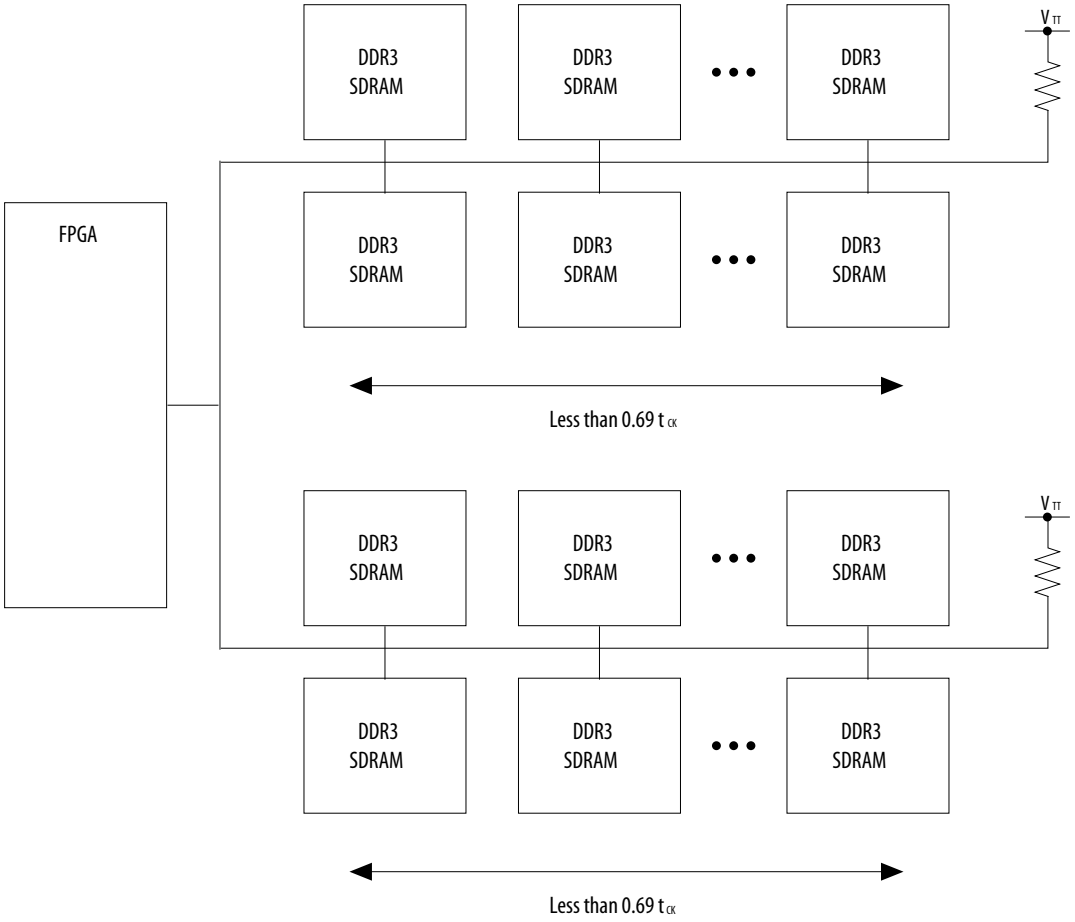


Every SDRAM component connected to the signal is a small load that causes discontinuity and degrades the signal. When using a single fly-by network topology, to minimize signal distortion, follow these guidelines:

- Use ×16 device instead ×4 or ×8 to minimize the number of devices connected to the trace.
- Keep the stubs as short as possible.
- Even with added loads from additional components, keep the total trace length short; keep the distance between the FPGA and the first SDRAM component less than 5 inches.
- Simulate clock signals to ensure a decent waveform.

The following figure shows an example of a double fly-by network topology. This topology is not rigid but you can use it as an alternative option. The advantage of using this topology is that you can have more SDRAM components in a system without violating the 0.69 tCK rule. However, as the signals branch out, the components still create discontinuity.

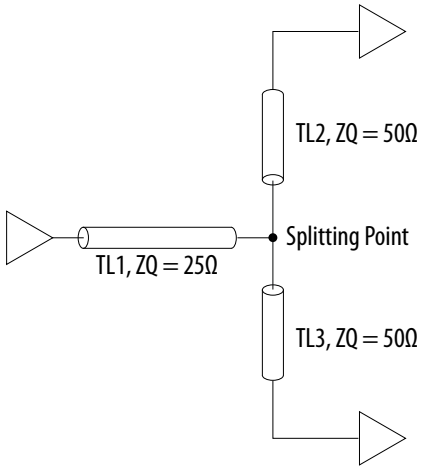
Figure 73. Double Fly-By Network Topology



You must perform simulations to find the location of the split, and the best impedance for the traces before and after the split.

The following figure shows a way to minimize the discontinuity effect. In this example, keep TL2 and TL3 matches in length. Keep TL1 longer than TL2 and TL3, so that it is easier to route all the signals during layout.

Figure 74. Minimizing Discontinuity Effect



You can also consider using a DIMM on each branch to replace the components. Because the trace impedance on the DIMM card is 40-ohm to 60-ohm, perform a board trace simulation to control the reflection to within the level your system can tolerate.

Using the fly-by daisy chain topology increases the complexity of the datapath and controller design to achieve leveling, but also greatly improves performance and eases board layout for SDRAM implementations.

You can also use the SDRAM components without leveling in a design if it may result in a more optimal solution, or use with devices that support the required electrical interface standard, but do not support the required read and write leveling functionality.

7.5.4.7. Clamshell Topology

In a DDR4 clamshell topology, SDRAM is arranged in two layers along either side of the chip, with individual memory devices opposite one another. This configuration allows for a smaller footprint than with fly-by topology, where memory devices are arranged on a single layer.

The small footprint of the clamshell topology requires less board space than fly-by topology. However, the close proximity of the memory devices in clamshell topology increases the complexity of the required device routing to prevent signal integrity problems.

Clamshell topology uses Address Mirroring to minimize undesired effects such as cross-talk, by splitting the chip select signal for each rank:

- A chip select that accesses the top layer of components, which have not been mirrored.
- A chip select that accesses the bottom layer of components, which have been mirrored.

The total number of chip selects required is double the interface's rank — for example, a single-rank memory interface requires two chip selects. The two chip selects are required for proper calibration of the interface, as a way of accounting for address

mirroring. Because the I/O columns have 4 chip-select pins, an external memory interface for a clamshell memory topology has a maximum of 2 ranks, in contrast with the fly-by topology which supports up to 4 ranks.

The JEDEC specification JESD21-C defines address mirroring for DDR4 as shown in the table below.

Table 254. Address Mirroring

Memory Controller Pin	DRAM Pin (Non-Mirrored)	DRAM Pin (Mirrored)
A3	A3	A4
A4	A4	A3
A5	A5	A6
A6	A6	A5
A7	A7	A8
A8	A8	A7
A11	A11	A13
A13	A13	A11
BA0	BA0	BA1
BA1	BA1	BA0
BG0 (1)	BG0	BG1
BG1 (1)	BG1	BG0
(1) BG0 and BG1 can be mirrored only when pin BG1 is present on the memory device.		

Enabling Clamshell Topology in Your External Memory Interface

1. Configure a single memory interface according to your requirements.
2. Select **Use clamshell layout** on the **General** tab in the parameter editor.
3. Set the number of chip-select pins equal to the number of ranks.

Note: Do not select the **Address Mirror** option in the parameter editor. Choosing a clamshell layout is sufficient to invoke address mirroring to configure the device.

Mapping

Table 255. Single Rank

Rank	Top/Bottom of Memory Device	CS Pin on Memory Device	CS Pin on FPGA
0	Top	CS0	CS0
0	Bottom	CS0	CS1

Table 256. Dual Rank

Rank	Top/Bottom of Memory Device	CS Pin on Memory Device	CS Pin on FPGA
0	Top	CS0	CS0
0	Bottom	CS0	CS2
1	Top	CS1	CS1
1	Bottom	CS1	CS3

Note: The single-rank clamshell and dual-rank clamshell pinouts are not interoperable.

7.5.4.8. Additional Layout Guidelines for DDR4 Twin-die Devices

Twin-die DDR4 memory devices have increased capacitive loading on the address, command, and memory clock signals, which can affect the signal integrity in a fly-by topology.

To ensure a good PCB layout, you should perform board-level simulations to optimize the fly-by topology, trace impedance, and terminations. The following techniques may help you improve signal integrity:

- Fly-by component placement: Compact layouts such as clamshell topologies tend to cause worse reflections. To reduce reflections at the first DRAM, add some additional signal routing between the first and second DRAMs, relative to the other fly-by routing lengths.
- PCB trace impedance: You may reduce reflections by increasing the trace impedance from the first to the last DRAM. However, be aware that thinner traces may cause issues with PCB fabrication.
- Board simulation models: Verify the IBIS model correlation accuracy with your memory vendor and determine whether package loss is modeled. HSPICE simulation models might be more accurate.
- Terminations: Experiment with different values of the parallel termination to V_{tt}.

If you encounter memory test errors during hardware testing and suspect problems with address and command signal integrity, you can confirm the address and command signal integrity as follows:

- Probe the `alert_n` signal with an oscilloscope and look for a falling edge after the memory has calibrated. A parity error on the address and command signals causes `alert_n` to pulse low.

7.5.5. Package Deskew

Trace lengths inside the device package are not uniform for all package pins. The nonuniformity of package traces can affect system timing for high frequencies. A package deskew option is available in the Intel Quartus Prime software.

If you do not enable the package deskew option, the Intel Quartus Prime software uses the package delay numbers to adjust skews on the appropriate signals; you do not need to adjust for package delays on the board traces. If you do enable the package deskew option, the Intel Quartus Prime software does not use the package delay numbers for timing analysis, and you must deskew the package delays with the board traces for the appropriate signals for your design.

Related Information

Layout Guidelines on page 245

7.5.5.1. DQ/DQS/DM Deskew

To get the package delay information, follow these steps:

1. Select the **FPGA DQ/DQS Package Skews Deskewed on Board** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Compile your design.
5. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

7.5.5.2. Address and Command Deskew

Deskew address and command delays as follows:

1. Select the **FPGA Address/Command Package Skews Deskewed on Board** checkbox on the **Board Settings** tab of the parameter editor.
2. Generate your IP.
3. Instantiate your IP in the project.
4. Compile your design.
5. Refer to the **All Package Pins** compilation report, or find the pin delays displayed in the `<core_name>.pin` file.

7.5.5.3. Package Deskew Recommendations for Intel Stratix 10 Devices

The following table shows package deskew recommendations for Intel Stratix 10 devices.

As operating frequencies increase, it becomes increasingly critical to perform package deskew. The frequencies listed in the table are the *minimum* frequencies for which you must perform package deskew.

If you plan to use a listed protocol at the specified frequency or higher, you must perform package deskew.

Protocol	Minimum Frequency (MHz) for Which to Perform Package Deskew		
	Single Rank	Dual Rank	Quad Rank
DDR4	933	800	667
DDR3	933	800	667
QDR IV	933	Not applicable	Not applicable
RLDRAM 3	933	667	Not applicable
QDR II, II+, II+ Xtreme	Not required	Not applicable	Not applicable

7.5.5.4. Deskew Example

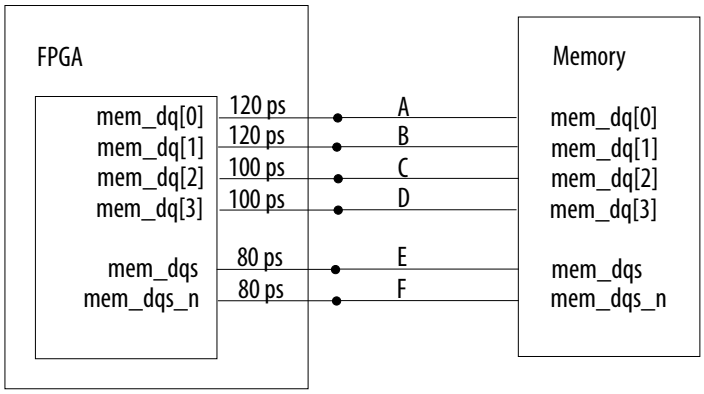
Consider an example where you want to deskew an interface with 4 DQ pins, 1 DQS pin, and 1 DQSn pin.

Let's assume an operating frequency of 667 MHz, and the package lengths for the pins reported in the **.pin** file as follows:

```
dq[0] = 120 ps
dq[1] = 120 ps
dq[2] = 100 ps
dq[3] = 100 ps
dqs   = 80 ps
dqs_n = 80 ps
```

The following figure illustrates this example.

Figure 75. Deskew Example

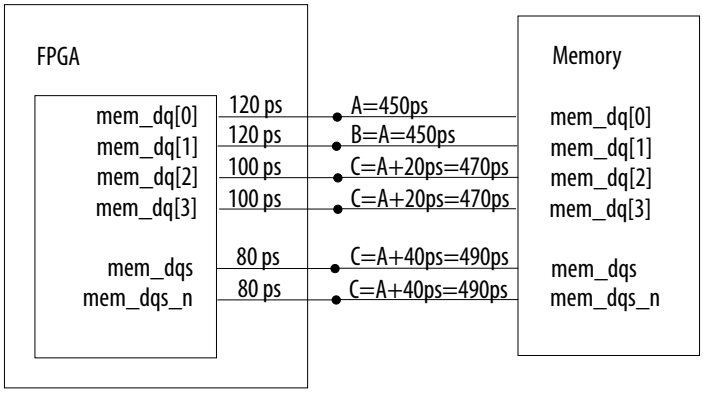


When you perform length matching for all the traces in the DQS group, you must take package delays into consideration. Because the package delays of traces A and B are 40 ps longer than the package delays of traces E and F, you would need to make the board traces for E and F 40 ps longer than the board traces for A and B.

A similar methodology would apply to traces C and D, which should be 20 ps longer than the lengths of traces A and B.

The following figure shows this scenario with the length of trace A at 450 ps.

Figure 76. Deskew Example with Trace Delay Calculations



When you enter the board skews into the Board Settings tab of the DDR3 parameter editor, you should calculate the board skew parameters as the sums of board delay and corresponding package delay. If a pin does not have a package delay (such as address and command pins), you should use the board delay only.

The example of the preceding figure shows an ideal case where board skews are perfectly matched. In reality, you should allow plus or minus 10 ps of skew mismatch within a DQS group (DQ/DQS/DM).

7.5.5.5. Package Migration

Package delays can be different for the same pin in different packages. If you want to use multiple migratable packages in your system, you should compensate for package skew as described in this topic.The information in this topic applies to Intel Stratix 10 devices.

Scenario 1

Your PCB is designed for multiple migratable devices, but you have only one device with which to go to production.

Assume two migratable packages, device A and device B, and that you want to go to production with device A. Follow these steps:

1. Perform package deskew for device A.
2. Compile your design for device A, with the **Package Skew** option enabled.
3. Note the skews in the `<core_name>.pin` file for device A. Deskew these package skews with board trace lengths as described in the preceding examples.
4. Recompile your design for device A.
5. For device B, open the parameter editor and deselect the **Package Deskew** option.
6. Calculate board skew parameters, only taking into account the board traces for device B, and enter that value into the parameter editor for device B.
7. Regenerate the IP and recompile the design for device B.
8. Verify that timing requirements are met for both device A and device B.

Scenario 2

Your PCB is designed for multiple migratable devices, and you want to go to production with all of them.

Assume you have device A and device B, and plan to use both devices in production. Follow these steps:

1. Do not perform any package deskew compensation for either device.
2. Compile a Quartus Prime design for device A with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
3. Verify that the **Report DDR** timing report meets your timing requirements.
4. Compile a Quartus Prime design for device B with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
5. Verify that the **Report DDR** timing report meets your timing requirements.

8. Intel Stratix 10 EMIF IP for QDR II/II+/II+ Xtreme

This chapter contains IP parameter descriptions, board skew equations, pin planning information, and board design guidance for Intel Stratix 10 external memory interfaces for QDR II/II+/II+ Xtreme.

8.1. Parameter Descriptions

The following topics describe the parameters available on each tab of the IP parameter editor, which you can use to configure your IP.

8.1.1. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: General

Table 257. Group: General / Interface

Display Name	Description
Configuration	Specifies the configuration of the memory interface. The available options depend on the protocol and the targeted FPGA product. (Identifier: PHY_QDR2_CONFIG_ENUM)

Table 258. Group: General / Clocks

Display Name	Description
Memory clock frequency	Specifies the **operating frequency** of the memory interface in MHz. If you change the memory frequency, you should update the memory latency parameters on the **Memory** tab and the memory timing parameters on the **Mem Timing** tab. (Identifier: PHY_QDR2_MEM_CLK_FREQ_MHZ)
Use recommended PLL reference clock frequency	Specifies that the PLL reference clock frequency is automatically calculated for best performance. *If you want to specify a different PLL reference clock frequency, uncheck the check box for this parameter.* (Identifier: PHY_QDR2_DEFAULT_REF_CLK_FREQ)
PLL reference clock frequency	This parameter tells the IP what PLL reference clock frequency the user will supply. Users must select a valid PLL reference clock frequency from the list. The values in the list can change when the memory interface frequency changes and/or the clock rate of user logic changes. It is recommended to use the fastest possible PLL reference clock frequency because it leads to better jitter performance. Selection is required only if the user does not check the "Use recommended PLL reference clock frequency" option. (Identifier: PHY_QDR2_USER_REF_CLK_FREQ_MHZ)
PLL reference clock jitter	Specifies the **peak-to-peak jitter** on the PLL reference clock source. The clock source of the PLL reference clock must meet or exceed the following jitter requirements: 10ps peak to peak, or 1.42ps RMS at 1e-12 BER, 1.22ps at 1e-16 BER. (Identifier: PHY_QDR2_REF_CLK_JITTER_PS)

continued...

Display Name	Description
Clock rate of user logic	Specifies the relationship between the user logic clock frequency and the memory clock frequency. For example, if the memory clock sent from the FPGA to the memory device is toggling at 800MHz, a quarter-rate interface means that the user logic in the FPGA runs at 200MHz. The list of available options is dependent on the memory protocol and device family. (Identifier: PHY_QDR2_RATE_ENUM)
Core clocks sharing	When a design contains multiple interfaces of the same protocol, rate, frequency, and PLL reference clock source, they can share a common set of core clock domains. By sharing core clock domains, they **reduce clock network usage and avoid clock synchronization logic between the interfaces**. To share core clocks, denote one of the interfaces as **"Master"**, and the remaining interfaces as **"Slave"**. In the RTL, connect the `clks_sharing_master_out` signal from the master interface to the `clks_sharing_slave_in` signal of all the slave interfaces. Both master and slave interfaces still expose their own output clock ports in the RTL (for example, `emif_usr_clk`, `afi_clk`), but the physical signals are equivalent, hence it does not matter whether a clock port from a master or a slave is used. *As the combined width of all interfaces sharing the same core clock increases, you may encounter timing closure difficulty for transfers between the FPGA core and the periphery.* (Identifier: PHY_QDR2_CORE_CLKS_SHARING_ENUM)
Export clks_sharing_slave_out to facilitate multi-slave connectivity	When more than one slave exist, you can either connect the clks_sharing_master_out interface from the master to the clks_sharing_slave_in interface of all the slaves (i.e. one-to-many topology), OR, you can connect the clks_sharing_master_out interface to one slave, and connect the clks_sharing_slave_out interface of that slave to the next slave (i.e. daisy-chain topology). Both approaches produce the same result. The daisy-chain approach may be easier to achieve in the Platform Designer tool, whereas the one-to-many approach may be more intuitive. (Identifier: PHY_QDR2_CORE_CLKS_SHARING_EXPOSE_SLAVE_OUT)
Specify additional core clocks based on existing PLL	Displays additional parameters allowing you to create additional output clocks based on the existing PLL. This parameter **provides an alternative clock-generation mechanism for when your design exhausts available PLL resources**. The additional output clocks that you create can be fed into the core. Clock signals created with this parameter are synchronous to each other, but asynchronous to the memory interface core clock domains (such as `emif_usr_clk` or `afi_clk`). *You must follow proper clock-domain-crossing techniques when transferring data between clock domains.* (Identifier: PLL_ADD_EXTRA_CLKS)

Table 259. Group: General / Clocks / Additional Core Clocks

Display Name	Description
Number of additional core clocks	Specifies the number of additional output clocks to create from the PLL. (Identifier: PLL_USER_NUM_OF_EXTRA_CLKS)

Table 260. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_0

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_5)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_5)

Table 261. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_1

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_6)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_6)

Table 262. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_2

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_7)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_7)

Table 263. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_3

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_8)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_8)

8.1.2. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: FPGA I/O

You should use Hyperlynx* or similar simulators to determine the best settings for your board. Refer to the EMIF Simulation Guidance wiki page for additional information.

Table 264. Group: FPGA I/O / FPGA I/O Settings

Display Name	Description
Voltage	The voltage level for the I/O pins driving the signals between the memory device and the FPGA memory interface. (Identifier: PHY_QDR2_IO_VOLTAGE)
Use default I/O settings	Specifies that a legal set of I/O settings are automatically selected. The default I/O settings are not necessarily optimized for a specific board. *To achieve optimal signal integrity, perform I/O simulations with IBIS models and enter the I/O settings manually, based on simulation results*. (Identifier: PHY_QDR2_DEFAULT_IO)

Table 265. Group: FPGA I/O / FPGA I/O Settings / Address/Command

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the address/command pins of the memory interface. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_QDR2_USER_AC_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_QDR2_USER_AC_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_QDR2_USER_AC_SLEW_RATE_ENUM)

Table 266. Group: FPGA I/O / FPGA I/O Settings / Memory Clock

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the memory clock pins. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_QDR2_USER_CK_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_QDR2_USER_CK_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_QDR2_USER_CK_SLEW_RATE_ENUM)

Table 267. Group: FPGA I/O / FPGA I/O Settings / Data Bus

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the data and data clock/strobe pins of the memory interface. The selected I/O standard option configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_QDR2_USER_DATA_IO_STD_ENUM)
Output mode	This parameter allows you to change the output current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_QDR2_USER_DATA_OUT_MODE_ENUM)
Input mode	This parameter allows you to change the input termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_QDR2_USER_DATA_IN_MODE_ENUM)

Table 268. Group: FPGA I/O / FPGA I/O Settings / PHY Inputs

Display Name	Description
PLL reference clock I/O standard	Specifies the I/O standard for the PLL reference clock of the memory interface. (Identifier: PHY_QDR2_USER_PLL_REF_CLK_IO_STD_ENUM)
RZQ I/O standard	Specifies the I/O standard for the RZQ pin used in the memory interface. (Identifier: PHY_QDR2_USER_RZQ_IO_STD_ENUM)

8.1.3. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Memory

Table 269. Group: Memory / Topology

Display Name	Description
Data width per device	Number of D and Q pins per QDR II device. (Identifier: MEM_QDR2_DATA_PER_DEVICE)
Enable BWS# pins	Indicates whether the interface uses the BWS#(**Byte Write Select**) pins. If enabled, 1 BWS# pin for every 9 D pins will be added. (Identifier: MEM_QDR2_BWS_EN)
Enable width expansion	Indicates whether to combine two memory devices to double the data bus width. With two devices, the interface supports a width expansion configuration up to 72-bits. For width expansion configuration, the address and control signals are routed to 2 devices. (Identifier: MEM_QDR2_WIDTH_EXPANDED)
Address width	Number of address pins. (Identifier: MEM_QDR2_ADDR_WIDTH)
Burst length	Burst length of the memory device. (Identifier: MEM_QDR2_BL)

8.1.4. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Mem Timing

These parameters should be read from the table in the datasheet associated with the speed bin of the memory device (not necessarily the frequency at which the interface is running).

Table 270. Group: Mem Timing

Display Name	Description
Speed bin	The speed grade of the memory device used. This parameter refers to the maximum rate at which the memory device is specified to run. (Identifier: MEM_QDR2_SPEEDBIN_ENUM)
tRL	tRL refers to the **QDR memory specific read latency**. This parameter describes the length of time after a Read command has been registered on the rising edge of the Write Clock (K) at the QDR memory before the first piece of read data (Q) can be expected at the output of the memory. It is measured in Write Clock (K) cycles. **The Read Latency is specific to a QDR memory device and cannot be modified to a different value.** The Read Latency (tRL) can have the following values: 1.5, 2, 2,5 clk cycles. (Identifier: MEM_QDR2_TRL_CYC)
tSA	tSA refers to the **setup time for the address and command bus (A) before the rising edge of the clock (K)**. The address and command bus must be stable for at least tSA before the rising edge of K. (Identifier: MEM_QDR2_TSA_NS)
tHA	tHA refers to the **hold time after the rising edge of the clock (K) to the address and command control bus (A)**. The address and command control bus must remain stable for at least tHA after the rising edge of K. (Identifier: MEM_QDR2_THA_NS)
tSD	tSD refers to the **setup time for the data bus (D) before the rising edge of the clock (K)**. The data bus must be stable for at least tSD before the rising edge of K. (Identifier: MEM_QDR2_TSD_NS)

continued...

Display Name	Description
tHD	tHD refers to the **hold time after the rising edge of the clock (K) to the data bus (D)**. The data bus must remain stable for at least tHD after the rising edge of K. (Identifier: MEM_QDR2_THD_NS)
tCQD	tCQD refers to the maximum time expected between an echo clock edge and valid data on the Read Data bus (Q). (Identifier: MEM_QDR2_TCQD_NS)
tCQDOH	tCQDOH refers to the minimum time expected between the echo clock (CQ or CQ#) edge and the last of the valid Read data (Q). (Identifier: MEM_QDR2_TCQDOH_NS)
Internal Jitter	QDRII internal jitter. (Identifier: MEM_QDR2_INTERNAL_JITTER_NS)
tCQH	tCQH describes the time period during which the echo clock (CQ, #CQ) is considered logically high. (Identifier: MEM_QDR2_TCQH_NS)
tCCQO	tCCQO describes the **skew between the rising edge of the C clock to the rising edge of the echo clock (CQ)** in QDRII memory devices. (Identifier: MEM_QDR2_TCCQO_NS)

8.1.5. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Board

Table 271. Group: Board / Intersymbol Interference/Crosstalk

Display Name	Description
Use default ISI/crosstalk values	You can enable this option to use default intersymbol interference and crosstalk values for your topology. Note that the default values are not optimized for your board. *For optimal signal integrity, it is recommended that you do not enable this parameter, but instead perform I/O simulation using IBIS models and Hyperlynx)*, and manually enter values based on your simulation results, instead of using the default values.* (Identifier: BOARD_QDR2_USE_DEFAULT_ISI_VALUES)
Address and command ISI/crosstalk	The address and command window reduction due to ISI and crosstalk effects. The number to be entered is the **total loss of margin on both the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR2_USER_AC_ISI_NS)
CQ/CQ# ISI/crosstalk	CQ/CQ# ISI/crosstalk describes the reduction of the read data window due to intersymbol interference and crosstalk effects on the CQ/CQ# signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR2_USER_RCLK_ISI_NS)
Read Q ISI/crosstalk	Read Q ISI/crosstalk describes the reduction of the read data window due to intersymbol interference and crosstalk effects on the CQ/CQ# signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR2_USER_RDATA_ISI_NS)
K/K# ISI/crosstalk	K/K# ISI/crosstalk describes the reduction of the write data window due to intersymbol interference and crosstalk effects on the K/K# signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the**

continued...

Display Name	Description
	setup side + measured loss on the hold side). *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR2_USER_WCLK_ISI_NS)
Write D ISI/crosstalk	Write D ISI/crosstalk describes the reduction of the write data window due to intersymbol interference and crosstalk effects on the signal when driven by driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR2_USER_WDATA_ISI_NS)

Table 272. Group: Board / Board and Package Skews

Display Name	Description
Package deskewed with board layout (Q group)	If you are compensating for package skew on the Q bus in the board layout (hence checking the box here), please **include package skew in calculating the following board skew parameters.** (Identifier: BOARD_QDR2_IS_SKEW_WITHIN_Q_DESKEWED)
Maximum board skew within Q group	This parameter describes the largest skew between all Q signals in a Q group. Q pins drive the data signals from the memory to the FPGA when the read operation is active. **Users should enter their board skew only.** Package skew will be calculated automatically, based on the memory interface configuration, and added to this value. **This value affects the read capture and write margins.** (Identifier: BOARD_QDR2_BRD_SKEW_WITHIN_Q_NS)
Maximum system skew within Q group	The largest skew between all Q pins in a Q group. Enter combined board and package skew. This value affects the read capture and write margins. (Identifier: BOARD_QDR2_PKG_BRD_SKEW_WITHIN_Q_NS)
Package deskewed with board layout (D group)	If you are compensating for package skew on the D and BWS# signals in the board layout (hence checking the box here), please **include package skew in calculating the following board skew parameters.** (Identifier: BOARD_QDR2_IS_SKEW_WITHIN_D_DESKEWED)
Maximum board skew within D group	This parameter refers to the largest skew between all D and BWS# signals in a D group. D pins are used for driving data signals to the memory device during a write operation. BWS# pins are used as Byte Write Select signals to control which byte(s) are written to the memory during a write operation. **Users should enter their board skew only.** Package skew will be calculated automatically, based on the memory interface configuration, and added to this value. **This value affects the read capture and write margins.** (Identifier: BOARD_QDR2_BRD_SKEW_WITHIN_D_NS)
Maximum system skew within D group	The largest skew between all D and BWS# pins in a D group. Enter combined board and package skew. This value affects the read capture and write margins. (Identifier: BOARD_QDR2_PKG_BRD_SKEW_WITHIN_D_NS)
Package deskewed with board layout (address/command bus)	Enable this parameter if you are compensating for package skew on the address, command, control, and memory clock buses in the board layout. **Include package skew in calculating the following board skew parameters.** (Identifier: BOARD_QDR2_IS_SKEW_WITHIN_AC_DESKEWED)
Maximum board skew within address/ command bus	The largest skew between the address and command signals. Enter the board skew only; package skew is calculated automatically, based on the memory interface configuration, and added to this value. (Identifier: BOARD_QDR2_BRD_SKEW_WITHIN_AC_NS)

continued...

Display Name	Description
Maximum system skew within address/ command bus	Maximum system skew within address/command bus refers to the largest skew between the address and command signals. (Identifier: BOARD_QDR2_PKG_BRD_SKEW_WITHIN_AC_NS)
Average delay difference between address/command and K	This parameter refers to the average delay difference between the Address/ Command signals and the K signal, calculated by averaging the longest and smallest Address/Command trace delay minus the maximum K trace delay. Positive values represent address and command signals that are longer than K signals and negative values represent address and command signals that are shorter than K signals. (Identifier: BOARD_QDR2_AC_TO_K_SKEW_NS)
Maximum K delay to device	The maximum K delay to device refers to the delay of the longest K trace from the FPGA to any device (Identifier: BOARD_QDR2_MAX_K_DELAY_NS)

8.1.6. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Controller

Table 273. Group: Controller

Display Name	Description
Maximum Avalon-MM burst length	Specifies the maximum burst length on the Avalon-MM bus. This will be used to configure the FIFOs to be able to manage the maximum data burst. **More core logic will be required for an increase in FIFO length.** (Identifier: CTRL_QDR2_AVL_MAX_BURST_COUNT)
Generate power-of-2 data bus widths for Qsys	**If enabled, the Avalon data bus width is rounded down to the nearest power-of-2**. The width of the symbols within the data bus is also rounded down to the nearest power-of-2. You should only enable this option if you know you will be connecting the memory interface to Qsys interconnect components that require the data bus and symbol width to be a power-of-2. **If this option is enabled, you cannot utilize the full density of the memory device.** For example, in x36 data width upon selecting this parameter, will define the Avalon data bus to 256-bit. This will ignore the upper 4-bit of data width. (Identifier: CTRL_QDR2_AVL_ENABLE_POWER_OF_TWO_BUS)

8.1.7. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Diagnostics

Table 274. Group: Diagnostics / Simulation Options

Display Name	Description
Calibration mode	Specifies whether to **skip memory interface calibration** during simulation, or to **simulate the full calibration** process. Simulating the full calibration process can take hours (or even days), depending on the width and depth of the memory interface. You can achieve much faster simulation times by skipping the calibration process, but that is only expected to work when the memory model is ideal and the interconnect delays are zero. *If you enable this parameter, the interface still performs some memory initialization before starting normal operations.* Abstract PHY is supported with skip calibration.

continued...

Display Name	Description
	(Identifier: DIAG_QDR2_SIM_CAL_MODE_ENUM)
Abstract phy for fast simulation	Specifies that the system use Abstract PHY for simulation. **Abstract PHY replaces the PHY with a model for fast simulation and can reduce simulation time by 3-10 times.** Abstract PHY is available for certain protocols and device families, and only when you select **Skip Calibration**. (Identifier: DIAG_QDR2_ABSTRACT_PHY)
Show verbose simulation debug messages	This option allows adjusting the verbosity of the simulation output messages. (Identifier: DIAG_QDR2_SIM_VERBOSE)

Table 275. Group: Diagnostics / Calibration Debug Options

Display Name	Description
Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies the connectivity of an Avalon slave interface for use by the Quartus Prime EMIF Debug Toolkit or user core logic. If you set this parameter to "**Disabled**", no debug features are enabled. If you set this parameter to "**Export**", an Avalon slave interface named "`cal_debug`" is exported from the IP. To use this interface with the EMIF Debug Toolkit, you must instantiate and connect an EMIF debug interface IP core to it, or connect it to the `cal_debug_out` interface of another EMIF core. If you select "**Add EMIF Debug Interface**", an EMIF debug interface component containing a JTAG Avalon Master is connected to the debug port, allowing the core to be accessed by the EMIF Debug Toolkit. *Only one EMIF debug interface should be instantiated per I/O column.* You can chain additional EMIF or PHYLite cores to the first by enabling the "**Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option for all cores in the chain, and selecting "**Export**" for the "**Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option on all cores after the first. (Identifier: DIAG_QDR2_EXPORT_SEQ_AVALON_SLAVE)
Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies that the IP export an Avalon-MM master interface (`cal_debug_out`) which can connect to the `cal_debug` interface of other EMIF cores residing in the same I/O column. **This parameter applies only if the EMIF Debug Toolkit or On-Chip Debug Port is enabled.** *Refer to the Debugging Multiple EMIFs wiki page for more information about debugging multiple EMIFs.* (Identifier: DIAG_QDR2_EXPORT_SEQ_AVALON_MASTER)
First EMIF Instance in the Avalon Chain	If selected, this EMIF instance will be the head of the Avalon interface chain connected to the master. For simulation purposes it is needed to identify the first EMIF instance in the avalon Chain. (Identifier: DIAG_QDR2_EXPORT_SEQ_AVALON_HEAD_OF_CHAIN)
Interface ID	Identifies interfaces within the I/O column, for use by the EMIF Debug Toolkit and the On-Chip Debug Port. Interface IDs should be unique among EMIF cores within the same I/O column. If the **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** parameter is set to **Disabled**, the interface ID is unused. (Identifier: DIAG_QDR2_INTERFACE_ID)
Use Soft NIOS Processor for On-Chip Debug	Enables a soft Nios processor as a peripheral component to access the **On-Chip Debug Port**. *Only one interface in a column can activate this option.* (Identifier: DIAG_SOFT_NIOS_MODE)

Table 276. Group: Diagnostics / Example Design

Display Name	Description
Number of core clocks sharing slaves to instantiate in the example design	Specifies the number of core clock sharing slaves to instantiate in the example design. This parameter applies only if you set the "**Core clocks sharing**" parameter in the "**General**" tab to "**Master**" or "**Slave**". (Identifier: DIAG_QDR2_EX_DESIGN_NUM_OF_SLAVES)
Enable In-System-Sources-and-Probes	Enables In-System-Sources-and-Probes in the example design for *common debug signals, such as calibration status or example traffic generator per-bit status*. This parameter must be enabled if you want to do driver margining using the EMIF Debug Toolkit. (Identifier: DIAG_QDR2_EX_DESIGN_ISSP_EN)

Table 278. Group: Diagnostics / Performance

Display Name	Description
Efficiency Monitor Mode	Adds an Efficiency Monitor component to the Avalon-MM interface of the memory controller, allowing you to view efficiency statistics of the interface. You can access the efficiency statistics using the EMIF Efficiency Monitor Toolkit. (Identifier: DIAG_QDR2_EFFICIENCY_MONITOR)

Table 279. Group: Diagnostics / Miscellaneous

Display Name	Description
Export PLL lock signal	Specifies whether to export the pll_locked signal at the IP top-level to indicate status of PLL. (Identifier: DIAG_EXPORT_PLL_LOCKED)

8.1.8. Intel Stratix 10 EMIF IP QDR II/II+/II+ Xtreme Parameters: Example Designs

Table 280. Group: Example Designs / Available Example Designs

Display Name	Description
Select design	Specifies the *creation of a full Quartus Prime project, instantiating an external memory interface and an example traffic generator, according to your parameterization*. After the design is created, you can specify the target device and pin location assignments, run a full compilation, verify timing closure, and test the interface on your board using the programming file created by the Quartus Prime assembler. The '**Generate Example Design**' button lets you generate simulation or synthesis file sets. (Identifier: EX_DESIGN_GUI_QDR2_SEL_DESIGN)

Table 281. Group: Example Designs / Example Design Files

Display Name	Description
Simulation	Specifies that the '**Generate Example Design**' button create all necessary file sets for simulation. Expect a short additional delay as the file set is created. *If you do not enable this parameter, simulation file sets are not created.* Instead, the output directory will contain the `ed_sim.qsys` file which holds Qsys details of the simulation example design, and a `make_sim_design.tcl` file with other corresponding `tcl` files. You can run `make_sim_design.tcl` from a command line to generate the

continued...

Display Name	Description
	simulation example design. The generated example designs for various simulators are **stored in the /sim sub-directory**. (Identifier: EX_DESIGN_GUI_QDR2_GEN_SIM)
Synthesis	Specifies that the '**Generate Example Design**' button create all necessary file sets for synthesis. Expect a short additional delay as the file set is created. *If you do not enable this parameter, synthesis file sets are not created.* Instead, the output directory will contain the `ed_synth.qsys` file which holds Qsys details of the synthesis example design, and a `make_qii_design.tcl` script with other corresponding `tcl` files. You can run `make_qii_design.tcl` from a command line to generate the synthesis example design. The generated example design is **stored in the /qii sub-directory**. (Identifier: EX_DESIGN_GUI_QDR2_GEN_SYNTH)

Table 282. Group: Example Designs / Generated HDL Format

Display Name	Description
Simulation HDL format	This option lets you choose the format of HDL in which generated simulation files are created. (Identifier: EX_DESIGN_GUI_QDR2_HDL_FORMAT)

Table 283. Group: Example Designs / Target Development Kit

Display Name	Description
Select board	Specifies that *when you select a development kit with a memory module, the generated example design contains all settings and fixed pin assignments to run on the selected board. You must select a development kit preset to generate a working example design for the specified development kit*. Any IP settings not applied directly from a development kit preset will not have guaranteed results when testing the development kit. To exclude hardware support of the example design, select '**none**' from the '**Select board**' pull down menu. When you apply a development kit preset, all IP parameters are automatically set appropriately to match the selected preset. If you want to save your current settings, you should do so before you apply the preset. You can save your settings under a different name using **File->Save as**. Current presets for development kit do not cover QDRII. (Identifier: EX_DESIGN_GUI_QDR2_TARGET_DEV_KIT)

8.2. Board Skew Equations

The following table presents the underlying equations for the board skew parameters.

8.2.1. Equations for QDRII, QDRII+, and QDRII+ Xtreme Board Skew Parameters

Table 284. Board Skew Parameter Equations

Parameter	Description/Equation
Maximum system skew within address/command bus	$(MaxAC - MinAC)$

continued...

Parameter	Description/Equation
	The largest skew between the address and command signals. Enter combined board and package skew.
Average delay difference between address/command and K	The average delay difference between the address and command signals and the K signal, calculated by averaging the longest and smallest Address/Command signal delay minus the K delay. Positive values represent address and command signals that are longer than K signals and negative values represent address and command signals that are shorter than K signals. The Quartus Prime software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins. $\frac{\sum\left(\frac{n=n}{n=1}\right)\left[\left(\frac{LongestACPathDelay + ShortestACPathDelay}{2}\right) - K_n PathDelay\right]}{n}$ where *n* is the number of K clocks.
Maximum board skew within Q group	The largest skew between all Q pins in a Q group. Enter your board skew only. Package skew is calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins. $^{groups}Max_g\left[\max Q_g - \min Q_g\right]$ where *g* is the number of Q group.
Maximum board skew within D group	The largest skew between all D and BWS# pins in a D group. Enter your board skew only. Package skew is calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins. $^{groups}Max_g\left[\max D_g - \min D_g\right]$ where *g* is the number of D group.
Maximum K delay to device	$\max_n(K_n PathDelay)$ where *n* is the number of K clocks.

8.3. Pin and Resource Planning

The following topics provide guidelines on pin placement for external memory interfaces.

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

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

Once all the requirements are known for your external memory interface, you can begin planning your system.

8.3.1. Interface Pins

DQS (data strobe or data clock) and DQ (data) pins are listed for EMIF supported banks in the device pin tables and are fixed at specific locations in the device. You must adhere to these pin locations to optimize routing, minimize skew, and maximize margins. Always check the device pin table for the actual locations of the DQS and DQ pins, and the EMIF pin table for location of address and control pins.

Pin tables are available here: https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html?1.

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

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

- Maximum possible interface width in any particular device is limited by the number of DQS groups available.
- Sufficient clock networks are available to the interface PLL as required by the IP.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other address and command, and clock pin placement requirements.

Note: The greater the number of banks, the greater the skew, hence Intel recommends that you always generate a test project of your desired configuration and confirm that it meets timing.

8.3.1.1. Estimating Pin Requirements

You should use the Intel Quartus Prime software for final pin fitting. However, you can estimate whether you have enough pins for your memory interface using the EMIF Device Selector on www.altera.com, or perform the following steps:

1. Determine how many read/write data pins are associated per data strobe or clock pair.
2. Calculate the number of other memory interface pins needed, including any other clocks (write clock or memory system clock), address, command, and RZQ. Refer to the External Memory Interface Pin Table to determine necessary Address/Command/Clock pins based on your desired configuration.
3. Calculate the total number of I/O banks required to implement the memory interface, given that an I/O bank supports up to 48 GPIO pins.

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

Related Information

Intel FPGA IP for External Memory Interfaces - Support Center

8.3.1.2. Maximum Number of Interfaces

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

Unless otherwise noted, the calculation for the maximum number of interfaces is based on independent interfaces where the address or command pins are not shared.

Note: You may need to share PLL clock outputs depending on your clock network usage.

For interface information for Intel Stratix 10, consult the EMIF Device Selector on www.altera.com.

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

Related Information

- Intel FPGA IP for External Memory Interfaces - Support Center
- Intel Stratix 10 EMIF Architecture: PLL Reference Clock Networks on page 21
- External Memory Interface Device Selector
- Intel Quartus Prime Pro Edition Handbook

8.3.1.3. FPGA Resources

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

8.3.1.4. OCT

You require one OCT calibration block if you are using an FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design. You can select any available OCT calibration block—it need not be within the same bank or side of the device as the memory interface pins. The only requirement is that the I/O bank where you place the OCT calibration block must use the same V_{CCIO} voltage as the memory interface.

The OCT calibration block uses a single R_{ZQ} pin. The R_{ZQ} pin in Intel Stratix 10 devices can be used as a general purpose I/O pin when it is not used to support OCT, provided the signal conforms to the bank voltage requirements.

8.3.1.5. PLL

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

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

8.3.1.6. Pin Guidelines for Intel Stratix 10 EMIF IP

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

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

- The numeric component of the `Bank Number` value identifies the I/O column, while the letter represents the I/O bank.
- The `Index within I/O Bank` value falls within one of the following ranges: 0 to 11, 12 to 23, 24 to 35, or 36 to 47, and represents I/O lanes 1, 2, 3, and 4, respectively.
- To determine if I/O banks are adjacent, you can refer to the I/O Pin Counts tables located in the *Intel Stratix 10 General Purpose I/O User Guide*. You can always assume I/O banks are adjacent within an I/O column except in the following conditions:
  - When an I/O bank is not bonded out on the package (contains the '-' symbol in the I/O table).
  - An I/O bank does not contain 48 pins, indicating it is only partially bonded out.
- The pairing pin for an I/O pin is located in the same I/O bank. You can identify the pairing pin by adding one to its `Index within I/O Bank` number (if it is an even number), or by subtracting one from its `Index within I/O Bank` number (if it is an odd number).

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

8.3.1.6.1. General Guidelines

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

If you are using the hard memory controller, you should employ the relative pin locations defined in the *<variation_name>*/altera_emif_arch_nd_*version number*/*<synth|sim>*/*<variation_name>*_altera_emif_arch_nd_*version number*_*<unique ID>*_readme.txt file, which is generated with your IP.

Note:

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

Observe the following general guidelines when placing pins for your Intel Stratix 10 external memory interface:

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

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

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

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

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

   b. QDR-IV requires that the polarity of all QKB/QKB# pins be swapped with respect to the polarity of the differential buffer inputs on the FPGA to ensure correct data capture on port B. All QKB pins on the memory device must be connected to the negative pins of the input buffers on the FPGA side, and all QKB# pins on the memory device must be connected to the positive pins of the input buffers on the FPGA side. Notice that the port names at the top-level of the IP already reflect this swap (that is, `mem_qkb` is assigned to the negative buffer leg, and `mem_qkb_n` is assigned to the positive buffer leg).
10. You can implement two x4 DQS groups with a single I/O lane. The pin table specifies which pins within an I/O lane can be used for the two pairs of DQS and DQS# signals. In addition, for x4 DQS groups you must observe the following rules:
    - There must be an even number of x4 groups in an external memory interface.
    - DQS group 0 and DQS group 1 must be placed in the same I/O lane. Similarly, DQS group 2 and group 3 must be in the same I/O lane. Generally, DQS group *X* and DQS group *X*+1 must be in the same I/O lane, where *X* is an even number.
11. You should place the write data groups according to the DQS grouping in the pin table and Pin Planner. Output-only data clocks for QDR II, QDR II+, and QDR II+ Extreme, and RLDRAM 3 protocols need not be placed on DQS/DQSn pins, but must be placed on a differential pin pair. They must be placed in the same I/O bank as the corresponding DQS group.

    Note: For RLDRAM 3, x36 device, `DQ[8:0]` and `DQ[26:18]` are referenced to `DK0/DK0#`, and `DQ[17:9]` and `DQ[35:27]` are referenced to `DK1/DK1#`.
12. For protocols and topologies with bidirectional data pins where a write data group consists of multiple read data groups, you should place the data groups and their respective write and read clock in the same bank to improve I/O timing.

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

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

Note:

1. x4 mode does not support DM/DBI, or Intel Stratix 10 EMIF IP for HPS.
2. If you are using an Intel Stratix 10 EMIF IP-based RLDRAM 3 external memory interface, you should ensure that all the pins in a DQS group (that is, DQ, DM, DK, and QK) are placed in the same I/O bank. This requirement facilitates timing closure and is necessary for successful compilation of your design.

I/O Banks Selection

- For each memory interface, select consecutive I/O banks. (That is, select banks that contain the same column number and letter before or after the respective I/O bank letter.)
- A memory interface can only span across I/O banks in the same I/O column.
- The number of I/O banks that you require depends on the memory interface width.
- In some device packages, the number of I/O pins in some LVDS I/O banks is less that 48 pins.

Address/Command Pins Location

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

CK Pins Assignment

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

- If the number of I/O banks is odd, assign one CK pin to the middle I/O bank.
- If the number of I/O banks is even, assign the CK pin to either of the middle two I/O banks.

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

PLL Reference Clock Pin Placement

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

- If you are sharing the PLL reference clock pin between several interfaces, the I/O banks must be adjacent. (That is, the banks must contain the same column number and letter before or after the respective I/O bank letter.)

The Intel Stratix 10 external memory interface IP does not support PLL cascading.

RZQ Pin Placement

You may place the R_{ZQ} pin in any I/O bank in an I/O column with the correct V_{CCIO} and V_{CCPT} for the memory interface I/O standard in use. However, the recommended location is in the address/command I/O bank, for greater flexibility during debug if a narrower interface project is required for testing.

DQ and DQS Pins Assignment

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

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

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

Sharing an I/O Bank Across Multiple Interfaces

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

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

8.3.1.6.2. QDR II, QDR II+ and QDR II+ Xtreme SRAM Command Signals

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

8.3.1.6.3. QDR II, QDR II+ and QDR II+ Xtreme SRAM Address Signals

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

8.3.1.6.4. QDR II, QDR II+, and QDR II+ Xtreme SRAM Clock Signals

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

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

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

The positive input clock, K, is the logical complement of the negative input clock, K#. Similarly, C and CQ are complements of C# and CQ#, respectively. With these complementary clocks, the rising edges of each clock leg latch the DDR data.

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

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

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

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

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

8.3.1.6.5. QDR II, QDR II+ and QDR II+ Xtreme SRAM Data, BWS, and QVLD Signals

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

At the pin, the read data is edge-aligned with the CQ and CQ# clocks while the write data is center-aligned with the K and K# clocks (see the following figures).

Figure 77. Edge-aligned CQ and Q Relationship During QDR II+ SRAM Read

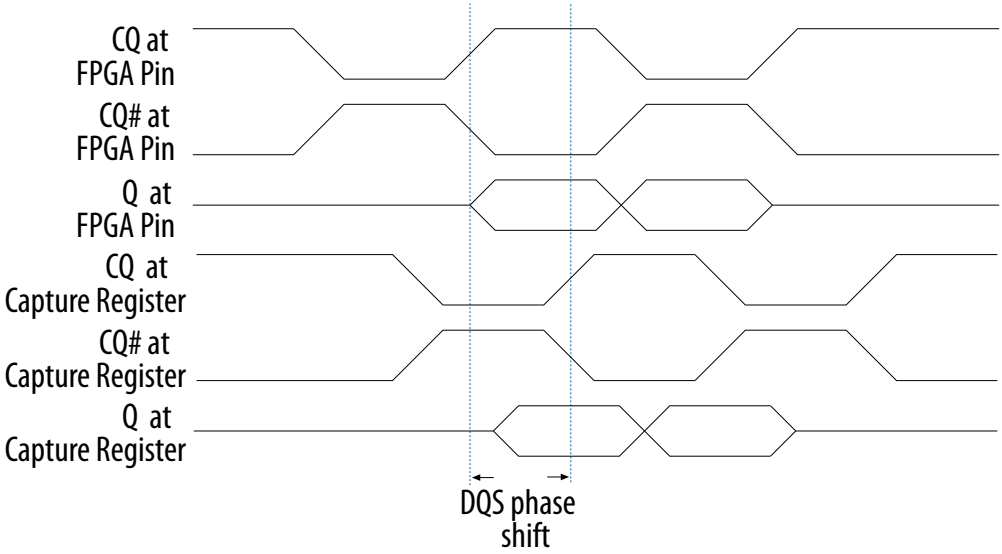
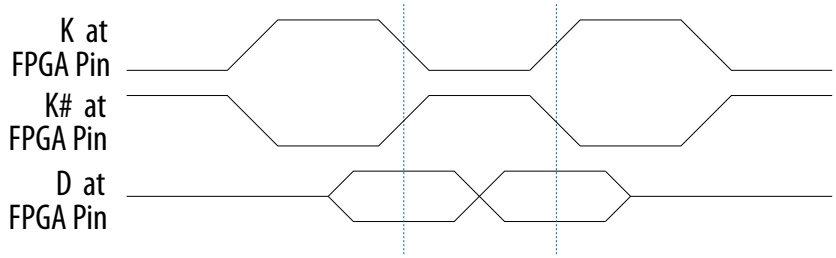


Figure 78. Center-aligned K and D Relationship During QDR II+ SRAM Write



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

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

Note: The Intel FPGA external memory interface IP does not use the QVLD signal.

8.3.1.6.6. Resource Sharing Guidelines (Multiple Interfaces)

In the external memory interface IP, different external memory interfaces can share PLL reference clock pins, core clock networks, I/O banks, and hard Nios processors. Each I/O bank has DLL and PLL resources, therefore these do not need to be shared.

The Intel Quartus Prime Fitter automatically merges DLL and PLL resources when a bank is shared by different external memory interfaces, and duplicates them for a multi-I/O-bank external memory interface.

PLL Reference Clock Pin

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

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

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

Note: You can place the `pll_ref_clk` pin in the address and command I/O bank or in a data I/O bank, there is no impact on timing. However, for greatest flexibility during debug (such as when creating designs with narrower interfaces), the recommended placement is in the address and command I/O bank.

Core Clock Network

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

Observe the following guidelines for sharing the core clock network:

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

I/O Bank

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

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

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

Hard Nios Processor

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

8.4. QDR II/II+/II+ Xtreme Board Design Guidelines

The following topics provide guidelines for you to improve your system's signal integrity and layout guidelines to help successfully implement a QDR II, QDR II+, or QDR II+ Xtreme SRAM interface in your system.

Note: In the following topics, QDR II SRAM refers to QDR II, QDR II+, and QDR II+ Xtreme SRAM unless stated otherwise.

The following topics focus on the following key factors that affect signal integrity:

- I/O standards
- QDR II SRAM configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

I/O Standards

QDR II SRAM interface signals use one of the following JEDEC I/O signaling standards:

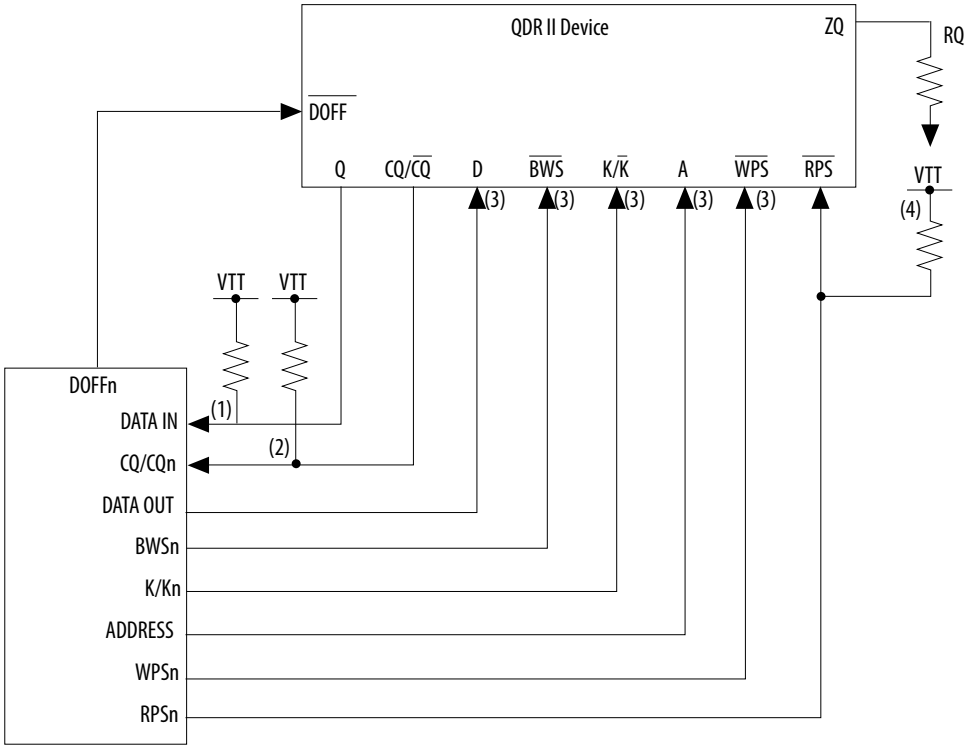
- HSTL-15—provides the advantages of lower power and lower emissions.
- HSTL-18—provides increased noise immunity with slightly greater output voltage swings.

8.4.1. QDR II SRAM Configurations

The QDR II SRAM Controller for Intel Stratix 10 EMIF IP supports interfaces with a single device, and two devices in a width expansion configuration up to maximum width of 72 bits.

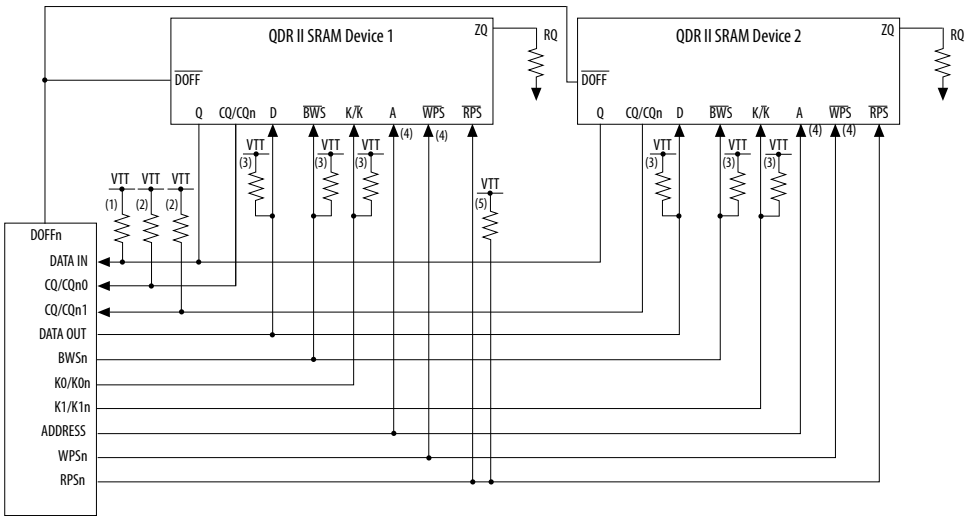
The following figure shows the main signal connections between the FPGA and a single QDR II SRAM component.

Figure 79. Configuration With A Single QDR II SRAM Component



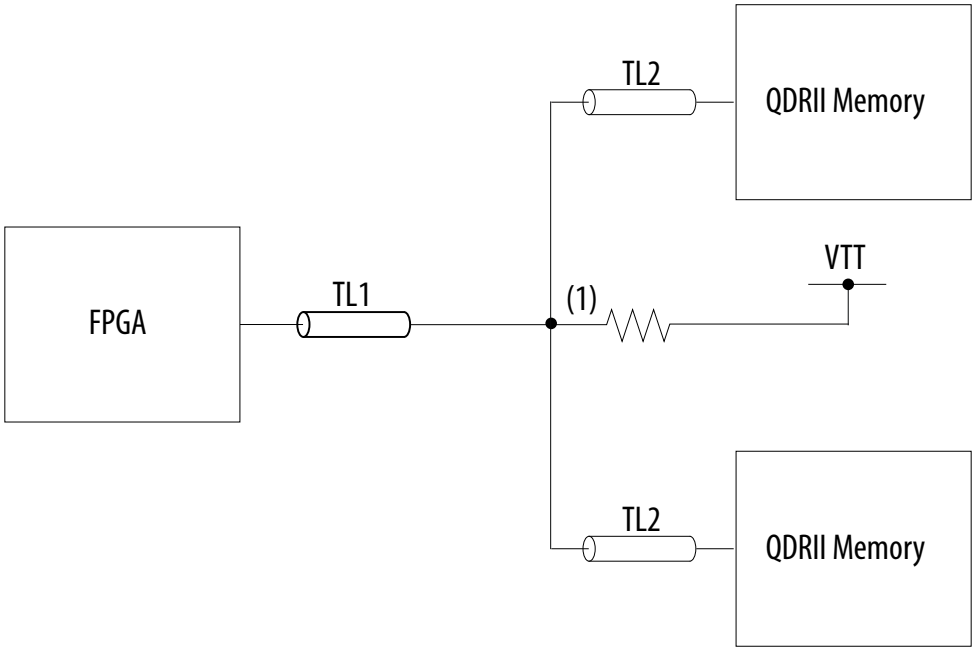
The following figure shows the main signal connections between the FPGA and two QDR II SRAM components in a width expansion configuration.

Figure 80. Configuration With Two QDR II SRAM Components In A Width Expansion Configuration



The following figure shows the detailed balanced topology recommended for the address and command signals in the width expansion configuration.

Figure 81. External Parallel Termination for Balanced Topology



8.4.2. General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Intel recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

Table 285. General Layout Guidelines

Parameter	Guidelines
Impedance	• All unused via pads must be removed, because they cause unwanted capacitance. • Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.
Decoupling Parameter	• Use 0.1 uF in 0402 size to minimize inductance • Make VTT voltage decoupling close to termination resistors • Connect decoupling caps between VTT and ground • Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin • Verify the capacitive decoupling using the Intel Power Distribution Network Design Tool
Power	• Route GND and V_{CC} as planes • Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation • Route VTT as islands or 250-mil (6.35-mm) power traces • Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces
General Routing	All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Intel recommends that signals from the same net group always be routed on the same layer. • Use 45° angles (*not* 90° corners) • Avoid T-Junctions for critical nets or clocks • Avoid T-junctions greater than 250 mils (6.35 mm) • Disallow signals across split planes • Restrict routing other signals close to system reset signals • Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks

Related Information

Power Distribution Network

8.4.3. QDR II Layout Guidelines

The following table summarizes QDR II and QDR II+ SRAM general routing layout guidelines.

Note:

1. The following layout guidelines include several +/- length based rules. These length based guidelines are for first order timing approximations if you cannot simulate the actual delay characteristics of your PCB implementation. They do not include any margin for crosstalk.
2. Intel recommends that you get accurate time base skew numbers when you simulate your specific implementation.
3. To reliably close timing to and from the periphery of the device, signals to and from the periphery should be registered before any further logic is connected.

Table 286. QDR II and QDR II+ SRAM Layout Guidelines

Parameter	Guidelines
General Routing	• If signals of the same net group must be routed on different layers with the same impedance characteristic, you must simulate your worst case PCB trace tolerances to ascertain actual propagation delay differences. Typical layer to layer trace delay variations are of 15 ps/inch order. • Avoid T-junctions greater than 150 ps.
Clock Routing	• Route clocks on inner layers with outer-layer run lengths held to under 150 ps. • These signals should maintain a 10-mil (0.254 mm) spacing from other nets. • Clocks should maintain a length-matching between clock pairs of ±5 ps. • Complementary clocks should maintain a length-matching between P and N signals of ±2 ps. • Keep the distance from the pin on the QDR II SRAM component to stub termination resistor (V_{TT}) to less than 50 ps for the K, K# clocks. • Keep the distance from the pin on the QDR II SRAM component to fly-by termination resistor (V_{TT}) to less than 100 ps for the K, K# clocks. • Keep the distance from the pin on the FPGA component to stub termination resistor (V_{TT}) to less than 50 ps for the echo clocks, CQ, CQ#, if they require an external discrete termination. • Keep the distance from the pin on the FPGA component to fly-by termination resistor (V_{TT}) to less than 100 ps for the echo clocks, CQ, CQ#, if they require an external discrete termination.
External Memory Routing Rules	• Keep the distance from the pin on the QDR II SRAM component to stub termination resistor (V_{TT}) to less than 50 ps for the write data, byte write select and address/command signal groups. • Keep the distance from the pin on the QDR II SRAM component to fly-by termination resistor (V_{TT}) to less than 100 ps for the write data, byte write select and address/command signal groups. • Keep the distance from the pin on the FPGA to stub termination resistor (V_{TT}) to less than 50 ps for the read data signal group. • Keep the distance from the pin on the FPGA to fly-by termination resistor (V_{TT}) to less than 100 ps for the read data signal group. • Parallelism rules for the QDR II SRAM data/address/command groups are as follows: — 4 mils for parallel runs < 0.1 inch (approximately 1× spacing relative to plane distance). — 5 mils for parallel runs < 0.5 inch (approximately 1× spacing relative to plane distance). — 10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance). — 15 mils for parallel runs between 1.0 and 6.0 inch (approximately 3× spacing relative to plane distance).
Maximum Trace Length	• Keep the maximum trace length of all signals from the FPGA to the QDR II SRAM components to 6 inches.

Related Information

Power Distribution Network

8.4.4. QDR II SRAM Layout Approach

Using the layout guidelines in the above table, Intel recommends the following layout approach:

1. Route the K/K# clocks and set the clocks as the target trace propagation delays for the output signal group.
2. Route the write data output signal group (`write data`, `byte write select`), ideally on the same layer as the K/K# clocks, to within ±10 ps skew of the K/K# traces.
3. Route the address/control output signal group (`address`, RPS, WPS), ideally on the same layer as the K/K# clocks, to within ±20 ps skew of the K/K# traces.
4. Route the CQ/CQ# clocks and set the clocks as the target trace propagation delays for the input signal group.
5. Route the read data output signal group (`read data`), ideally on the same layer as the CQ/CQ# clocks, to within ±10 ps skew of the CQ/CQ# traces.
6. The output and input groups do not need to have the same propagation delays, but they must have all the signals matched closely within the respective groups.

Note: Intel recommends that you create your project with a fully implemented external memory interface, and observe the interface timing margins to determine the actual margins for your design.

Although the recommendations in this section are based on simulations, you can apply the same general principles when determining the best termination scheme, drive strength setting, and loading style to any board designs. Even armed with this knowledge, it is still critical that you perform simulations, either using IBIS or HSPICE models, to determine the quality of signal integrity on your designs.

8.4.5. Package Deskew

You should follow Intel's package deskew guidance.

Related Information

Package Deskew

8.4.6. Package Migration

Package delays can be different for the same pin in different packages. If you want to use multiple migratable packages in your system, you should compensate for package skew as described in this topic.The information in this topic applies to Intel Stratix 10 devices.

Scenario 1

Your PCB is designed for multiple migratable devices, but you have only one device with which to go to production.

Assume two migratable packages, device A and device B, and that you want to go to production with device A. Follow these steps:

1. Perform package deskew for device A.
2. Compile your design for device A, with the **Package Skew** option enabled.
3. Note the skews in the `<core_name>.pin` file for device A. Deskew these package skews with board trace lengths as described in the preceding examples.

4. Recompile your design for device A.
5. For device B, open the parameter editor and deselect the **Package Deskew** option.
6. Calculate board skew parameters, only taking into account the board traces for device B, and enter that value into the parameter editor for device B.
7. Regenerate the IP and recompile the design for device B.
8. Verify that timing requirements are met for both device A and device B.

Scenario 2

Your PCB is designed for multiple migratable devices, and you want to go to production with all of them.

Assume you have device A and device B, and plan to use both devices in production. Follow these steps:

1. Do not perform any package deskew compensation for either device.
2. Compile a Quartus Prime design for device A with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
3. Verify that the **Report DDR** timing report meets your timing requirements.
4. Compile a Quartus Prime design for device B with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
5. Verify that the **Report DDR** timing report meets your timing requirements.

8.4.7. Slew Rates

For optimum timing margins and best signal integrity for the address, command, and memory clock signals, you should generally use fast slew rates and external terminations.

In board simulation, fast slew rates may show a perceived signal integrity problem, such as reflections or a nonmonotonic waveform in the SSTL I/O switching region. Such indications may cause you to consider using slow slew rate options for either the address and command signals or the memory clock, or both.

If you set the **FPGA I/O tab parameter options ➤ Address/Command ➤ Slew Rate** and **Memory Clock ➤ Slew Rate** parameters to different values, a warning message appears: .

```
Warning: .emif_0: When the address/command signals and the memory clock signals
do not use the same slew rate setting, signals using the "Slow" setting are
delayed relative to signals using "Fast" setting. For accurate timing analysis,
you must perform I/O simulation and manually include the delay as board skew.
To avoid the issue, use the same slew rate setting for both address/command
signals and memory clock signals whenever possible.
```

Note: The warning message applies only to board-level simulation, and does not require any delay adjustments in the PCB design or Board tab parameter settings.

Due to limitations of the IBIS model correlation tolerance and the accuracy of the board simulation model, it is possible for signal integrity problems to appear when using fast slew rate during simulation but not occur during operation on hardware. If

you observe a signal integrity problem during simulation with a fast slew rate, use an oscilloscope to view the signal at that point in hardware, to verify whether the problem exists on hardware, or only in simulation.

If the signal integrity problem exists on hardware as well as in simulation, using different slew rates for the address and command signals and the clock remains a valid approach, and the address and command calibration stage will help to improve the address and command to clock setup and hold time margins.

9. Intel Stratix 10 EMIF IP for QDR-IV

This chapter contains IP parameter descriptions, board skew equations, pin planning information, and board design guidance for Intel Stratix 10 external memory interfaces for QDR-IV.

9.1. Parameter Descriptions

The following topics describe the parameters available on each tab of the IP parameter editor, which you can use to configure your IP.

9.1.1. Intel Stratix 10 EMIF IP QDR-IV Parameters: General

Table 287. Group: General / Interface

Display Name	Description
Configuration	Specifies the configuration of the memory interface. The available options depend on the protocol and the targeted FPGA product. (Identifier: PHY_QDR4_CONFIG_ENUM)

Table 288. Group: General / Clocks

Display Name	Description
Memory clock frequency	Specifies the **operating frequency** of the memory interface in MHz. If you change the memory frequency, you should update the memory latency parameters on the **Memory** tab and the memory timing parameters on the **Mem Timing** tab. (Identifier: PHY_QDR4_MEM_CLK_FREQ_MHZ)
Use recommended PLL reference clock frequency	Specifies that the PLL reference clock frequency is automatically calculated for best performance. *If you want to specify a different PLL reference clock frequency, uncheck the check box for this parameter.* (Identifier: PHY_QDR4_DEFAULT_REF_CLK_FREQ)
PLL reference clock frequency	This parameter tells the IP what PLL reference clock frequency the user will supply. Users must select a valid PLL reference clock frequency from the list. The values in the list can change when the memory interface frequency changes and/or the clock rate of user logic changes. It is recommended to use the fastest possible PLL reference clock frequency because it leads to better jitter performance. Selection is required only if the user does not check the "Use recommended PLL reference clock frequency" option. (Identifier: PHY_QDR4_USER_REF_CLK_FREQ_MHZ)
PLL reference clock jitter	Specifies the **peak-to-peak jitter** on the PLL reference clock source. The clock source of the PLL reference clock must meet or exceed the following jitter requirements: 10ps peak to peak, or 1.42ps RMS at 1e-12 BER, 1.22ps at 1e-16 BER. (Identifier: PHY_QDR4_REF_CLK_JITTER_PS)

continued...

Display Name	Description
Clock rate of user logic	Specifies the relationship between the user logic clock frequency and the memory clock frequency. For example, if the memory clock sent from the FPGA to the memory device is toggling at 800MHz, a quarter-rate interface means that the user logic in the FPGA runs at 200MHz. The list of available options is dependent on the memory protocol and device family. (Identifier: PHY_QDR4_RATE_ENUM)
Core clocks sharing	When a design contains multiple interfaces of the same protocol, rate, frequency, and PLL reference clock source, they can share a common set of core clock domains. By sharing core clock domains, they **reduce clock network usage and avoid clock synchronization logic between the interfaces**. To share core clocks, denote one of the interfaces as **"Master"**, and the remaining interfaces as **"Slave"**. In the RTL, connect the `clks_sharing_master_out` signal from the master interface to the `clks_sharing_slave_in` signal of all the slave interfaces. Both master and slave interfaces still expose their own output clock ports in the RTL (for example, `emif_usr_clk`, `afi_clk`), but the physical signals are equivalent, hence it does not matter whether a clock port from a master or a slave is used. *As the combined width of all interfaces sharing the same core clock increases, you may encounter timing closure difficulty for transfers between the FPGA core and the periphery.* (Identifier: PHY_QDR4_CORE_CLKS_SHARING_ENUM)
Export clks_sharing_slave_out to facilitate multi-slave connectivity	When more than one slave exist, you can either connect the clks_sharing_master_out interface from the master to the clks_sharing_slave_in interface of all the slaves (i.e. one-to-many topology), OR, you can connect the clks_sharing_master_out interface to one slave, and connect the clks_sharing_slave_out interface of that slave to the next slave (i.e. daisy-chain topology). Both approaches produce the same result. The daisy-chain approach may be easier to achieve in the Platform Designer tool, whereas the one-to-many approach may be more intuitive. (Identifier: PHY_QDR4_CORE_CLKS_SHARING_EXPOSE_SLAVE_OUT)
Specify additional core clocks based on existing PLL	Displays additional parameters allowing you to create additional output clocks based on the existing PLL. This parameter **provides an alternative clock-generation mechanism for when your design exhausts available PLL resources**. The additional output clocks that you create can be fed into the core. Clock signals created with this parameter are synchronous to each other, but asynchronous to the memory interface core clock domains (such as `emif_usr_clk` or `afi_clk`). *You must follow proper clock-domain-crossing techniques when transferring data between clock domains.* (Identifier: PLL_ADD_EXTRA_CLKS)

Table 289. Group: General / Clocks / Additional Core Clocks

Display Name	Description
Number of additional core clocks	Specifies the number of additional output clocks to create from the PLL. (Identifier: PLL_USER_NUM_OF_EXTRA_CLKS)

Table 290. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_0

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_5)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_5)

Table 291. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_1

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_6)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_6)

Table 292. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_2

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_7)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_7)

Table 293. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_3

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_8)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_8)

9.1.2. Intel Stratix 10 EMIF IP QDR-IV Parameters: FPGA I/O

You should use Hyperlynx* or similar simulators to determine the best settings for your board. Refer to the EMIF Simulation Guidance wiki page for additional information.

Table 294. Group: FPGA I/O / FPGA I/O Settings

Display Name	Description
Voltage	The voltage level for the I/O pins driving the signals between the memory device and the FPGA memory interface. (Identifier: PHY_QDR4_IO_VOLTAGE)
Use default I/O settings	Specifies that a legal set of I/O settings are automatically selected. The default I/O settings are not necessarily optimized for a specific board. *To achieve optimal signal integrity, perform I/O simulations with IBIS models and enter the I/O settings manually, based on simulation results*. (Identifier: PHY_QDR4_DEFAULT_IO)

Table 295. Group: FPGA I/O / FPGA I/O Settings / Address/Command

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the address/command pins of the memory interface. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_QDR4_USER_AC_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_QDR4_USER_AC_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_QDR4_USER_AC_SLEW_RATE_ENUM)

Table 296. Group: FPGA I/O / FPGA I/O Settings / Memory Clock

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the memory clock pins. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_QDR4_USER_CK_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_QDR4_USER_CK_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_QDR4_USER_CK_SLEW_RATE_ENUM)

Table 297. Group: FPGA I/O / FPGA I/O Settings / Data Bus

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the data and data clock/strobe pins of the memory interface. The selected I/O standard option configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_QDR4_USER_DATA_IO_STD_ENUM)
Output mode	This parameter allows you to change the output current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_QDR4_USER_DATA_OUT_MODE_ENUM)
Input mode	This parameter allows you to change the input termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_QDR4_USER_DATA_IN_MODE_ENUM)
Use recommended initial Vrefin	Specifies that the initial Vrefin setting is calculated automatically, to a reasonable value based on termination settings. (Identifier: PHY_QDR4_USER_AUTO_STARTING_VREFIN_EN)
Initial Vrefin	Specifies the **initial value for the reference voltage on the data pins(Vrefin)**. This value is entered as a percentage of the supply voltage level on the I/O pins. The specified value serves as a starting point and may be overridden by calibration to provide better timing margins. If you choose to **skip Vref calibration** (**Diagnostics** tab), this is the value that is used as the Vref for the interface. (Identifier: PHY_QDR4_USER_STARTING_VREFIN)

Table 298. Group: FPGA I/O / FPGA I/O Settings / PHY Inputs

Display Name	Description
PLL reference clock I/O standard	Specifies the I/O standard for the PLL reference clock of the memory interface. (Identifier: PHY_QDR4_USER_PLL_REF_CLK_IO_STD_ENUM)
RZQ I/O standard	Specifies the I/O standard for the RZQ pin used in the memory interface. (Identifier: PHY_QDR4_USER_RZQ_IO_STD_ENUM)

9.1.3. Intel Stratix 10 EMIF IP QDR-IV Parameters: Memory

Table 299. Group: Memory / Topology

Display Name	Description
DQ width per device	Specifies number of DQ pins per port per QDR IV device. Available widths for DQ are x18 and x36. (Identifier: MEM_QDR4_DQ_PER_PORT_PER_DEVICE)
Enable width expansion	Indicates whether to combine two memory devices to double the data bus width. With two devices, the interface supports a width expansion configuration up to 72-bits. For width expansion configuration, the address and control signals are routed to 2 devices. (Identifier: MEM_QDR4_WIDTH_EXPANDED)
Address width	Number of address pins. (Identifier: MEM_QDR4_ADDR_WIDTH)
Memory Type	The QDR-IV family includes two members: MEM_XP: QDR-IV Xtreme Performance (XP) with a Maximum Clock Frequency of 1066MHz MEM_HP: QDR-IV High Performance (HP) with a Maximum Clock Frequency of 667MHz. (Identifier: MEM_QDR4_MEM_TYPE_ENUM)

Table 300. Group: Memory / Configuration Register Settings

Display Name	Description
Address bus inversion	Enable address bus inversion. AINV are all active high at memory device. (Identifier: MEM_QDR4_ADDR_INV_ENA)
Data bus inversion	Enable data bus inversion for DQ pins. DINVA[1:0] and DINVB[1:0] are all active high. When set to 1, the corresponding bus is inverted at memory device. If the data inversion feature is programmed to be OFF, then the DINVA/DINVB output bits will always be driven to 0. (Identifier: MEM_QDR4_DATA_INV_ENA)
Use address parity bit	Indicates whether to use an extra address parity bit and enable address parity error detection. (Identifier: MEM_QDR4_USE_ADDR_PARITY)
ODT (Clock)	Determines the configuration register setting that controls the clock ODT setting. (Identifier: MEM_QDR4_CK_ODT_MODE_ENUM)
ODT (Address/Command)	Determines the configuration register setting that controls the address/command ODT setting. (Identifier: MEM_QDR4_AC_ODT_MODE_ENUM)
ODT (Data)	Determines the configuration register setting that controls the data ODT setting. (Identifier: MEM_QDR4_DATA_ODT_MODE_ENUM)
Output drive (pull-up)	Determines the configuration register setting that controls the pull-up output drive setting. (Identifier: MEM_QDR4_PU_OUTPUT_DRIVE_MODE_ENUM)
Output drive (pull-down)	Determines the configuration register setting that controls the pull-down output drive setting. (Identifier: MEM_QDR4_PD_OUTPUT_DRIVE_MODE_ENUM)

9.1.4. Intel Stratix 10 EMIF IP QDR-IV Parameters: Mem Timing

These parameters should be read from the table in the datasheet associated with the speed bin of the memory device (not necessarily the frequency at which the interface is running).

Table 301. Group: Mem Timing

Display Name	Description
Speed bin	The speed grade of the memory device used. This parameter refers to the maximum rate at which the memory device is specified to run. (Identifier: MEM_QDR4_SPEEDBIN_ENUM)
tISH	tISH provides the **setup/hold window requirement for the entire data bus (DK or DINV) in all the data groups with respect to the DK clock**. After deskew calibration, this parameter describes the intersection window for all the individual data bus signals setup/hold margins. (Identifier: MEM_QDR4_TISH_PS)
tQKQ_max	tQKQ_max describes the **maximum skew** between the **read strobe (QK)** clock edge to the **data bus (DQ/DINV)** edge. (Identifier: MEM_QDR4_TQKQ_MAX_PS)
tQH	tQH specifies the **output hold time for the DQ/DINV in relation to QK**. (Identifier: MEM_QDR4_TQH_CYC)
tCKDK_max	tCKDK_max refers to the **maximum skew** from the **memory clock (CK)** to the **write strobe (DK)**. (Identifier: MEM_QDR4_TCKDK_MAX_PS)
tCKDK_min	tCKDK_min refers to the **minimum skew** from the **memory clock (CK)** to the **write strobe (DK)**. (Identifier: MEM_QDR4_TCKDK_MIN_PS)
tCKQK_max	tCKQK_max refers to the **maximum skew** from the **memory clock (CK)** to the **read strobe (QK)**. (Identifier: MEM_QDR4_TCKQK_MAX_PS)
tASH	tASH provides the **setup/hold window requirement for the address bus in relation to the CK clock**. Because the individual signals in the address bus may not be perfectly aligned with each other, this parameter describes the intersection window for all the individual address signals setup/hold margins. (Identifier: MEM_QDR4_TASH_PS)
tCSH	tCSH provides the **setup/hold window requirement for the control bus (LD#, RW#) in relation to the CK clock**. Because the individual signals in the control bus may not be perfectly aligned with each other, this parameter describes the intersection window for all the individual control signals setup/hold margins. (Identifier: MEM_QDR4_TCSH_PS)

9.1.5. Intel Stratix 10 EMIF IP QDR-IV Parameters: Board

Table 302. Group: Board / Intersymbol Interference/Crosstalk

Display Name	Description
Use default ISI/crosstalk values	You can enable this option to use default intersymbol interference and crosstalk values for your topology. Note that the default values are not optimized for your board. *For optimal signal integrity, it is recommended that you do not enable this parameter, but instead perform I/O simulation*

continued...

Display Name	Description
	using IBIS models and Hyperlynx, and manually enter values based on your simulation results, instead of using the default values.* (Identifier: BOARD_QDR4_USE_DEFAULT_ISI_VALUES)
Address and command ISI/crosstalk	The address and command window reduction due to ISI and crosstalk effects. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR4_USER_AC_ISI_NS)
QK/QK# ISI/crosstalk	QK/QK# ISI/crosstalk describes the reduction of the read data window due to intersymbol interference and crosstalk effects on the QK/QK# signal when driven by the memory device during a read. The number to be entered in the Quartus Prime software is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR4_USER_RCLK_ISI_NS)
Read DQ ISI/crosstalk	The reduction of the read data window due to ISI and crosstalk effects on the DQ signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR4_USER_RDATA_ISI_NS)
DK/DK# ISI/crosstalk	DK/DK# ISI/crosstalk describes the reduction of the write data window due to intersymbol interference and crosstalk effects on the DK/DK# signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR4_USER_WCLK_ISI_NS)
Write DQ ISI/crosstalk	The reduction of the write data window due to intersymbol interference and crosstalk effects on the DQ signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_QDR4_USER_WDATA_ISI_NS)

Table 303. Group: Board / Board and Package Skews

Display Name	Description
Package deskewed with board layout (QK group)	If you are compensating for package skew on the QK bus in the board layout (hence checking the box here), please **include package skew in calculating the following board skew parameters.** (Identifier: BOARD_QDR4_IS_SKEW_WITHIN_QK_DESKEWED)
Maximum board skew within QK group	The largest skew between all DQ and DM pins in a QK group. Enter your board skew only. Package skew will be calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins. (Identifier: BOARD_QDR4_BRD_SKEW_WITHIN_QK_NS)
Maximum system skew within QK group	Maximum system skew within QK group refers to the largest skew between all DQ and DM pins in a QK group. This value can affect the read capture and write margins. (Identifier: BOARD_QDR4_PKG_BRD_SKEW_WITHIN_QK_NS)
Package deskewed with board layout (address/command bus)	Enable this parameter if you are compensating for package skew on the address, command, control, and memory clock buses in the board layout. **Include package skew in calculating the following board skew parameters.** (Identifier: BOARD_QDR4_IS_SKEW_WITHIN_AC_DESKEWED)

continued...

Display Name	Description
Maximum board skew within address/ command bus	The largest skew between the address and command signals. Enter the board skew only; package skew is calculated automatically, based on the memory interface configuration, and added to this value. (Identifier: BOARD_QDR4_BRD_SKEW_WITHIN_AC_NS)
Maximum system skew within address/ command bus	Maximum system skew within address/command bus refers to the largest skew between the address and command signals. (Identifier: BOARD_QDR4_PKG_BRD_SKEW_WITHIN_AC_NS)
Average delay difference between DK and CK	This parameter describes the average delay difference between the DK signals and the CK signal, calculated by averaging the longest and smallest DK trace delay minus the CK trace delay. Positive values represent DK signals that are longer than CK signals and negative values represent DK signals that are shorter than CK signals. (Identifier: BOARD_QDR4_DK_TO_CK_SKEW_NS)
Maximum delay difference between devices	This parameter describes the largest propagation delay on the DQ signals between ranks. For example, in a two-rank configuration where devices are placed in series, there is an extra propagation delay for DQ signals going to and coming back from the furthest device compared to the nearest device. *This parameter is only applicable when there is more than one rank.* (Identifier: BOARD_QDR4_SKEW_BETWEEN_DIMMS_NS)
Maximum skew between DK groups	This parameter describes the largest skew between DK signals in different DK groups. (Identifier: BOARD_QDR4_SKEW_BETWEEN_DK_NS)
Average delay difference between address/command and CK	The average delay difference between the address/command signals and the CK signal, calculated by averaging the longest and smallest address/ command signal trace delay minus the maximum CK trace delay. Positive values represent address and command signals that are longer than CK signals and negative values represent address and command signals that are shorter than CK signals. (Identifier: BOARD_QDR4_AC_TO_CK_SKEW_NS)
Maximum CK delay to device	The maximum CK delay to device refers to the delay of the longest CK trace from the FPGA to any device. (Identifier: BOARD_QDR4_MAX_CK_DELAY_NS)
Maximum DK delay to device	The maximum DK delay to device refers to the delay of the longest DK trace from the FPGA to any device. (Identifier: BOARD_QDR4_MAX_DK_DELAY_NS)

9.1.6. Intel Stratix 10 EMIF IP QDR-IV Parameters: Controller

Table 304. Group: Controller

Display Name	Description
Maximum Avalon-MM burst length	Specifies the maximum burst length on the Avalon-MM bus. This will be used to configure the FIFOs to be able to manage the maximum data burst. **More core logic will be required for an increase in FIFO length.** (Identifier: CTRL_QDR4_AVL_MAX_BURST_COUNT)
Generate power-of-2 data bus widths for Qsys	**If enabled, the Avalon data bus width is rounded down to the nearest power-of-2.** The width of the symbols within the data bus is also rounded down to the nearest power-of-2. You should only enable this option if you know you will be connecting the memory interface to Qsys interconnect components that require the data bus and symbol width to be a power-of-2. **If this option is enabled, you cannot utilize the full density of the memory device.** For example, in x36 data width upon selecting this parameter, will define the Avalon data bus to 256-bit. This will ignore the upper 4-bit of data width.

continued...

Display Name	Description
	(Identifier: CTRL_QDR4_AVL_ENABLE_POWER_OF_TWO_BUS)
Additional read-after-write turnaround time	Specifies an additional number of idle memory cycles when switching the data bus (of a single port) from a write to a read. (Identifier: CTRL_QDR4_ADD_RAW_TURNAROUND_DELAY_CYC)
Additional write-after-read turnaround time	Specifies an additional number of idle memory cycles when switching the data bus (of a single port) from a read to a write. (Identifier: CTRL_QDR4_ADD_WAR_TURNAROUND_DELAY_CYC)

9.1.7. Intel Stratix 10 EMIF IP QDR-IV Parameters: Diagnostics

Table 305. Group: Diagnostics / Simulation Options

Display Name	Description
Calibration mode	Specifies whether to **skip memory interface calibration** during simulation, or to **simulate the full calibration** process. Simulating the full calibration process can take hours (or even days), depending on the width and depth of the memory interface. You can achieve much faster simulation times by skipping the calibration process, but that is only expected to work when the memory model is ideal and the interconnect delays are zero. *If you enable this parameter, the interface still performs some memory initialization before starting normal operations.* Abstract PHY is supported with skip calibration. (Identifier: DIAG_QDR4_SIM_CAL_MODE_ENUM)
Abstract phy for fast simulation	Specifies that the system use Abstract PHY for simulation. **Abstract PHY replaces the PHY with a model for fast simulation and can reduce simulation time by 3-10 times.** Abstract PHY is available for certain protocols and device families, and only when you select **Skip Calibration**. (Identifier: DIAG_QDR4_ABSTRACT_PHY)
Show verbose simulation debug messages	This option allows adjusting the verbosity of the simulation output messages. (Identifier: DIAG_QDR4_SIM_VERBOSE)

Table 306. Group: Diagnostics / Calibration Debug Options

Display Name	Description
Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies the connectivity of an Avalon slave interface for use by the Quartus Prime EMIF Debug Toolkit or user core logic. If you set this parameter to "**Disabled**", no debug features are enabled. If you set this parameter to "**Export**", an Avalon slave interface named "`cal_debug`" is exported from the IP. To use this interface with the EMIF Debug Toolkit, you must instantiate and connect an EMIF debug interface IP core to it, or connect it to the `cal_debug_out` interface of another EMIF core. If you select "**Add EMIF Debug Interface**", an EMIF debug interface component containing a JTAG Avalon Master is connected to the debug port, allowing the core to be accessed by the EMIF Debug Toolkit. *Only one EMIF debug interface should be instantiated per I/O column.* You can chain additional EMIF or PHYLite cores to the first by enabling the "**Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option for all cores in the chain, and selecting "**Export**" for the "**Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option on all cores after the first. (Identifier: DIAG_QDR4_EXPORT_SEQ_AVALON_SLAVE)
Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies that the IP export an Avalon-MM master interface (`cal_debug_out`) which can connect to the `cal_debug` interface of other EMIF cores residing in the same I/O column. **This parameter applies only**

continued...

Display Name	Description
	if the EMIF Debug Toolkit or On-Chip Debug Port is enabled. *Refer to the Debugging Multiple EMIFs wiki page for more information about debugging multiple EMIFs.* (Identifier: DIAG_QDR4_EXPORT_SEQ_AVALON_MASTER)
First EMIF Instance in the Avalon Chain	If selected, this EMIF instance will be the head of the Avalon interface chain connected to the master. For simulation purposes it is needed to identify the first EMIF instance in the avalon Chain. (Identifier: DIAG_QDR4_EXPORT_SEQ_AVALON_HEAD_OF_CHAIN)
Interface ID	Identifies interfaces within the I/O column, for use by the EMIF Debug Toolkit and the On-Chip Debug Port. Interface IDs should be unique among EMIF cores within the same I/O column. If the **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** parameter is set to **Disabled**, the interface ID is unused. (Identifier: DIAG_QDR4_INTERFACE_ID)
Skip VREF_in calibration	Specifies to skip the VREF stage of calibration. **Enable this parameter for debug purposes only**; generally, you should include the VREF calibration stage during normal operation. (Identifier: DIAG_QDR4_SKIP_VREF_CAL)
Use Soft NIOS Processor for On-Chip Debug	Enables a soft Nios processor as a peripheral component to access the **On-Chip Debug Port**. *Only one interface in a column can activate this option.* (Identifier: DIAG_SOFT_NIOS_MODE)

Table 307. Group: Diagnostics / Example Design

Display Name	Description
Number of core clocks sharing slaves to instantiate in the example design	Specifies the number of core clock sharing slaves to instantiate in the example design. This parameter applies only if you set the "**Core clocks sharing**" parameter in the "**General**" tab to "**Master**" or "**Slave**". (Identifier: DIAG_QDR4_EX_DESIGN_NUM_OF_SLAVES)
Enable In-System-Sources-and-Probes	Enables In-System-Sources-and-Probes in the example design for *common debug signals, such as calibration status or example traffic generator per-bit status*. This parameter must be enabled if you want to do driver margining using the EMIF Debug Toolkit. (Identifier: DIAG_QDR4_EX_DESIGN_ISSP_EN)

Table 309. Group: Diagnostics / Performance

Display Name	Description
Efficiency Monitor Mode	Adds an Efficiency Monitor component to the Avalon-MM interface of the memory controller, allowing you to view efficiency statistics of the interface. You can access the efficiency statistics using the EMIF Efficiency Monitor Toolkit. (Identifier: DIAG_QDR4_EFFICIENCY_MONITOR)

Table 310. Group: Diagnostics / Miscellaneous

Display Name	Description
Export PLL lock signal	Specifies whether to export the pll_locked signal at the IP top-level to indicate status of PLL. (Identifier: DIAG_EXPORT_PLL_LOCKED)

9.1.8. Intel Stratix 10 EMIF IP QDR-IV Parameters: Example Designs

Table 311. Group: Example Designs / Available Example Designs

Display Name	Description
Select design	Specifies the *creation of a full Quartus Prime project, instantiating an external memory interface and an example traffic generator, according to your parameterization*. After the design is created, you can specify the target device and pin location assignments, run a full compilation, verify timing closure, and test the interface on your board using the programming file created by the Quartus Prime assembler. The '**Generate Example Design**' button lets you generate simulation or synthesis file sets. (Identifier: EX_DESIGN_GUI_QDR4_SEL_DESIGN)

Table 312. Group: Example Designs / Example Design Files

Display Name	Description
Simulation	Specifies that the '**Generate Example Design**' button create all necessary file sets for simulation. Expect a short additional delay as the file set is created. *If you do not enable this parameter, simulation file sets are not created.* Instead, the output directory will contain the `ed_sim.qsys` file which holds Qsys details of the simulation example design, and a `make_sim_design.tcl` file with other corresponding `tcl` files. You can run `make_sim_design.tcl` from a command line to generate the simulation example design. The generated example designs for various simulators are **stored in the /sim sub-directory**. (Identifier: EX_DESIGN_GUI_QDR4_GEN_SIM)
Synthesis	Specifies that the '**Generate Example Design**' button create all necessary file sets for synthesis. Expect a short additional delay as the file set is created. *If you do not enable this parameter, synthesis file sets are not created.* Instead, the output directory will contain the `ed_synth.qsys` file which holds Qsys details of the synthesis example design, and a `make_qii_design.tcl` script with other corresponding `tcl` files. You can run `make_qii_design.tcl` from a command line to generate the synthesis example design. The generated example design is **stored in the /qii sub-directory**. (Identifier: EX_DESIGN_GUI_QDR4_GEN_SYNTH)

Table 313. Group: Example Designs / Generated HDL Format

Display Name	Description
Simulation HDL format	This option lets you choose the format of HDL in which generated simulation files are created. (Identifier: EX_DESIGN_GUI_QDR4_HDL_FORMAT)

Table 314. Group: Example Designs / Target Development Kit

Display Name	Description
Select board	Specifies that *when you select a development kit with a memory module, the generated example design contains all settings and fixed pin assignments to run on the selected board. You must select a development kit preset to generate a working example design for the specified development kit*. Any IP settings not applied directly from a development kit preset will not have guaranteed results when testing the development kit. To exclude hardware support of the example design, select '**none**' from the '**Select board**' pull down menu. When you apply a development kit preset, all IP parameters are automatically set appropriately to match the selected preset. If you want to save your current settings, you should do so before you apply the preset. You can save your settings under a different name using **File->Save as**. Current presets for development kit do not support QDR-IV. (Identifier: EX_DESIGN_GUI_QDR4_TARGET_DEV_KIT)

9.2. Board Skew Equations

The following table presents the underlying equations for the board skew parameters.

9.2.1. Equations for QDR-IV Board Skew Parameters

Table 315. Board Skew Parameter Equations

Parameter	Description/Equation
Maximum system skew within address/command bus	The largest skew between the address and command signals. Enter combined board and package skew. $(MaxAC - MinAC)$
Average delay difference between address/command and CK	The average delay difference between the address and command signals and the CK signal, calculated by averaging the longest and smallest Address/Command signal delay minus the CK delay. Positive values represent address and command signals that are longer than CK signals and negative values represent address and command signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins. $\frac{\sum\left(\frac{n=n}{n=1}\right)\left[\left(\frac{LongestACPathDelay + ShortestACPathDelay}{2}\right) - CK_nPathDelay\right]}{n}$ where *n* is the number of memory clocks.
Maximum System skew within QK group	The largest skew between all DQ and DM pins in a QK group. Enter combined board and package skew. This value affects the read capture and write margins. $\max_n(\max DQ_n - \min DQ_n)$ Where *n* includes both DQa and DQb
Maximum CK delay to device	The delay of the longest CK trace from the FPGA to any device. $[\max_n(CK_nPathDelay)$ where *n* is the number of memory clocks.
Maximum DK delay to device	The delay of the longest DK trace from the FPGA to any device. $\max_n(DK_nPathDelay)$ where *n* is the number of DK.
Average delay difference between DK and CK	The average delay difference between the DK signals and the CK signal, calculated by averaging the longest and smallest DK delay minus the CK delay. Positive values represent DK signals that are longer than CK signals and negative values represent DK signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the DK signals to have appropriate setup and hold margins. $\frac{\min_{n,m}\left(CK_nPathDelay - DK_mPathDelay + \max_{n,m}\left(CK_nPathDelay - DK_mPathDelay\right)\right.}{2}$ where *n* is the number of memory clocks and *m* is the number of DK.
Maximum skew between DK groups	The largest skew between DK signals in different DK groups. $\max_n(\max DK_n - \min DK_n)$ where *n* is the number of DK. Where *n* includes both DQa and DQb.

9.3. Pin and Resource Planning

The following topics provide guidelines on pin placement for external memory interfaces.

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

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

Once all the requirements are known for your external memory interface, you can begin planning your system.

9.3.1. Interface Pins

DQS (data strobe or data clock) and DQ (data) pins are listed for EMIF supported banks in the device pin tables and are fixed at specific locations in the device. You must adhere to these pin locations to optimize routing, minimize skew, and maximize margins. Always check the device pin table for the actual locations of the DQS and DQ pins, and the EMIF pin table for location of address and control pins.

Pin tables are available here: https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html?1.

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

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

- Maximum possible interface width in any particular device is limited by the number of DQS groups available.
- Sufficient clock networks are available to the interface PLL as required by the IP.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other address and command, and clock pin placement requirements.

Note: The greater the number of banks, the greater the skew, hence Intel recommends that you always generate a test project of your desired configuration and confirm that it meets timing.

9.3.1.1. Estimating Pin Requirements

You should use the Intel Quartus Prime software for final pin fitting. However, you can estimate whether you have enough pins for your memory interface using the EMIF Device Selector on www.altera.com, or perform the following steps:

1. Determine how many read/write data pins are associated per data strobe or clock pair.
2. Calculate the number of other memory interface pins needed, including any other clocks (write clock or memory system clock), address, command, and RZQ. Refer to the External Memory Interface Pin Table to determine necessary Address/Command/Clock pins based on your desired configuration.
3. Calculate the total number of I/O banks required to implement the memory interface, given that an I/O bank supports up to 48 GPIO pins.

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

Related Information

Intel FPGA IP for External Memory Interfaces - Support Center

9.3.1.2. Maximum Number of Interfaces

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

Unless otherwise noted, the calculation for the maximum number of interfaces is based on independent interfaces where the address or command pins are not shared.

Note: You may need to share PLL clock outputs depending on your clock network usage.

For interface information for Intel Stratix 10, consult the EMIF Device Selector on www.altera.com.

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

Related Information

- Intel FPGA IP for External Memory Interfaces - Support Center
- Intel Stratix 10 EMIF Architecture: PLL Reference Clock Networks on page 21
- External Memory Interface Device Selector
- Intel Quartus Prime Pro Edition Handbook

9.3.1.3. FPGA Resources

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

9.3.1.4. OCT

You require one OCT calibration block if you are using an FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design. You can select any available OCT calibration block—it need not be within the same bank or side of the device as the memory interface pins. The only requirement is that the I/O bank where you place the OCT calibration block must use the same V_{CCIO} voltage as the memory interface.

The OCT calibration block uses a single R_{ZQ} pin. The R_{ZQ} pin in Intel Stratix 10 devices can be used as a general purpose I/O pin when it is not used to support OCT, provided the signal conforms to the bank voltage requirements.

9.3.1.5. PLL

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

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

9.3.1.6. Pin Guidelines for Intel Stratix 10 EMIF IP

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

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

- The numeric component of the `Bank Number` value identifies the I/O column, while the letter represents the I/O bank.
- The `Index within I/O Bank` value falls within one of the following ranges: 0 to 11, 12 to 23, 24 to 35, or 36 to 47, and represents I/O lanes 1, 2, 3, and 4, respectively.
- To determine if I/O banks are adjacent, you can refer to the I/O Pin Counts tables located in the *Intel Stratix 10 General Purpose I/O User Guide*. You can always assume I/O banks are adjacent within an I/O column except in the following conditions:
  - When an I/O bank is not bonded out on the package (contains the '-' symbol in the I/O table).
  - An I/O bank does not contain 48 pins, indicating it is only partially bonded out.
- The pairing pin for an I/O pin is located in the same I/O bank. You can identify the pairing pin by adding one to its `Index within I/O Bank` number (if it is an even number), or by subtracting one from its `Index within I/O Bank` number (if it is an odd number).

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

9.3.1.6.1. General Guidelines

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

If you are using the hard memory controller, you should employ the relative pin locations defined in the *<variation_name>*`/altera_emif_arch_nd_`*version number*`/`*<synth|sim>*`/`*<variation_name>*`_altera_emif_arch_nd_`*version number*`_`*<unique ID>*`_readme.txt` file, which is generated with your IP.

Note:

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

Observe the following general guidelines when placing pins for your Intel Stratix 10 external memory interface:

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

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

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

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

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

   b. QDR-IV requires that the polarity of all QKB/QKB# pins be swapped with respect to the polarity of the differential buffer inputs on the FPGA to ensure correct data capture on port B. All QKB pins on the memory device must be connected to the negative pins of the input buffers on the FPGA side, and all QKB# pins on the memory device must be connected to the positive pins of the input buffers on the FPGA side. Notice that the port names at the top-level of the IP already reflect this swap (that is, `mem_qkb` is assigned to the negative buffer leg, and `mem_qkb_n` is assigned to the positive buffer leg).
10. You can implement two x4 DQS groups with a single I/O lane. The pin table specifies which pins within an I/O lane can be used for the two pairs of DQS and DQS# signals. In addition, for x4 DQS groups you must observe the following rules:
    - There must be an even number of x4 groups in an external memory interface.
    - DQS group 0 and DQS group 1 must be placed in the same I/O lane. Similarly, DQS group 2 and group 3 must be in the same I/O lane. Generally, DQS group *X* and DQS group *X*+1 must be in the same I/O lane, where *X* is an even number.
11. You should place the write data groups according to the DQS grouping in the pin table and Pin Planner. Output-only data clocks for QDR II, QDR II+, and QDR II+ Extreme, and RLDRAM 3 protocols need not be placed on DQS/DQSn pins, but must be placed on a differential pin pair. They must be placed in the same I/O bank as the corresponding DQS group.

    Note: For RLDRAM 3, x36 device, `DQ[8:0]` and `DQ[26:18]` are referenced to `DK0/DK0#`, and `DQ[17:9]` and `DQ[35:27]` are referenced to `DK1/DK1#`.
12. For protocols and topologies with bidirectional data pins where a write data group consists of multiple read data groups, you should place the data groups and their respective write and read clock in the same bank to improve I/O timing.

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

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

Note:

1. x4 mode does not support DM/DBI, or Intel Stratix 10 EMIF IP for HPS.
2. If you are using an Intel Stratix 10 EMIF IP-based RLDRAM 3 external memory interface, you should ensure that all the pins in a DQS group (that is, DQ, DM, DK, and QK) are placed in the same I/O bank. This requirement facilitates timing closure and is necessary for successful compilation of your design.

I/O Banks Selection

- For each memory interface, select consecutive I/O banks. (That is, select banks that contain the same column number and letter before or after the respective I/O bank letter.)
- A memory interface can only span across I/O banks in the same I/O column.
- The number of I/O banks that you require depends on the memory interface width.
- In some device packages, the number of I/O pins in some LVDS I/O banks is less that 48 pins.

Address/Command Pins Location

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

CK Pins Assignment

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

- If the number of I/O banks is odd, assign one CK pin to the middle I/O bank.
- If the number of I/O banks is even, assign the CK pin to either of the middle two I/O banks.

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

PLL Reference Clock Pin Placement

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

- If you are sharing the PLL reference clock pin between several interfaces, the I/O banks must be adjacent. (That is, the banks must contain the same column number and letter before or after the respective I/O bank letter.)

The Intel Stratix 10 external memory interface IP does not support PLL cascading.

RZQ Pin Placement

You may place the R_{ZQ} pin in any I/O bank in an I/O column with the correct V_{CCIO} and V_{CCPT} for the memory interface I/O standard in use. However, the recommended location is in the address/command I/O bank, for greater flexibility during debug if a narrower interface project is required for testing.

DQ and DQS Pins Assignment

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

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

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

Sharing an I/O Bank Across Multiple Interfaces

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

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

9.3.1.6.2. QDR IV SRAM Commands and Addresses, AP, and AINV Signals

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

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

The above features are available as **Option Control** under **Configuration Register Settings** in the parameter editor. The commands and addresses must meet the memory address and command setup (`tAS`, `tCS`) and hold (`tAH`, `tCH`) time requirements.

9.3.1.6.3. QDR IV SRAM Clock Signals

QDR IV SRAM devices have three pairs of differential clocks.

The three QDR IV differential clocks are as follows:

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

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

DKx and DKx# samples the DQx inputs on both rising and falling edges. Similarly, QKx and QKx# samples the DQx outputs on both rising and falling edges.

QDR IV SRAM devices employ two sets of free running differential clocks to accompany the data. The DKx and DKx# clocks are the differential input data clocks used during writes. The QKx and QKx# clocks are the output data clocks used during reads. Each pair of DKx and DKx#, or QKx and QKx# clocks are associated with either 9 or 18 data bits.

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

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

- For ×18 data bus width configuration, there are 9 data bits associated with each pair of write and read clocks. So, there are two pairs of DKx and DKx# pins and two pairs of QKx or QKx# pins.
- For ×36 data bus width configuration, there are 18 data bits associated with each pair of write and read clocks. So, there are two pairs of DKx and DKx# pins and two pairs of QKx or QKx# pins.

There are tCKDK timing requirements for skew between CK and DKx or CK# and DKx# .Similarly, there are tCKQK timing requirements for skew between CK and QKx or CK# and QKx# .

9.3.1.6.4. QDR IV SRAM Data, DINV, and QVLD Signals

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

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

Figure 82. Edge-Aligned DQ and QK Relationship During Read

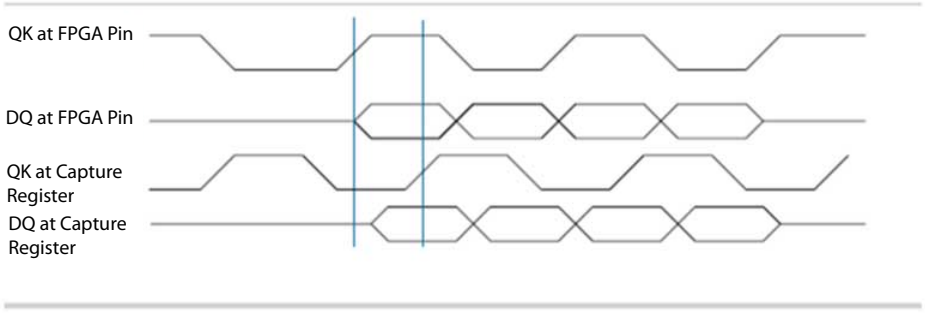
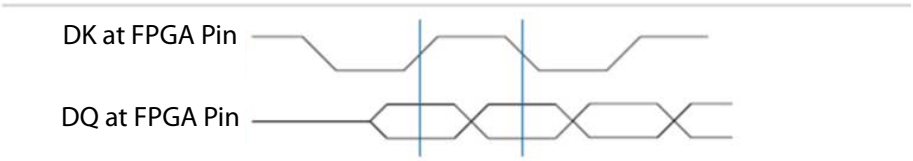


Figure 83. Center-Aligned DQ and DK Relationship During Write



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

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

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

To enable the data pin inversion feature, click **Configuration Register Settings ➤ Option Control** in the parameter editor.

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

Note: The Intel ZFPGA external memory interface IP does not use the QVLD signal.

9.3.1.7. Resource Sharing Guidelines (Multiple Interfaces)

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

PLL Reference Clock Pin

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

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

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

Note: You can place the `pll_ref_clk` pin in the address and command I/O bank or in a data I/O bank, there is no impact on timing. However, for greatest flexibility during debug (such as when creating designs with narrower interfaces), the recommended placement is in the address and command I/O bank.

Core Clock Network

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

Observe the following guidelines for sharing the core clock network:

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

I/O Bank

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

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

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

Hard Nios Processor

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

9.4. QDR-IV Board Design Guidelines

The following topics provide guidelines for you to improve your system's signal integrity and layout guidelines to help successfully implement a QDR-IV SRAM interface in your system.

The following topics focus on the following key factors that affect signal integrity:

- I/O standards
- QDR-IV SRAM configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

I/O Standards

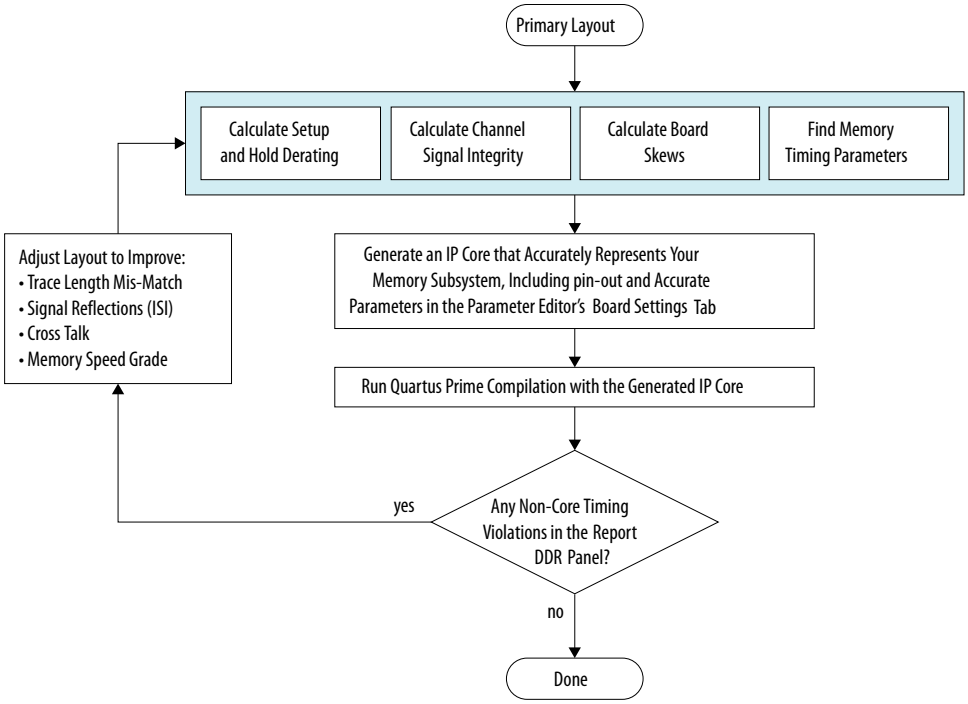
QDR-IV SRAM interface signals use one of the following JEDEC I/O signaling standards:

- HSTL-15—provides the advantages of lower power and lower emissions.
- HSTL-18—provides increased noise immunity with slightly greater output voltage swings.

9.4.1. QDR-IV Layout Approach

For all practical purposes, you can regard the Timing Analyzer report on your memory interface as definitive for a given set of memory and board timing parameters. You will find timing under Report DDR in Timing Analyzer and on the Timing Analysis tab in the parameter editor.

The following flowchart illustrates the recommended process to follow during the design phase, to determine timing margin and make iterative improvements to your design.



For more detailed simulation guidance, refer to the wiki: https://community.intel.com/t5/FPGA-Wiki/Arria-10-EMIF-Simulation-Guidance/ta-p/735201

Intersymbol Interference/Crosstalk

For information on intersymbol interference and crosstalk, refer to the wiki: https://community.intel.com/t5/FPGA-Wiki/Measuring-Channel-Signal-Integrity/ta-p/735495

Board Skew

For information on calculating board skew parameters, refer to *Board Skew Equations*, in this chapter.

If you know the absolute delays for all the memory related traces, the interactive Board Skew Parameter Tool can help you calculate the necessary parameters.

Memory Timing Parameters

You can find the memory timing parameters to enter in the parameter editor, in your memory vendor's datasheet.

9.4.2. General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Intel recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

Table 316. General Layout Guidelines

Parameter	Guidelines
Impedance	• All unused via pads must be removed, because they cause unwanted capacitance. • Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.
Decoupling Parameter	• Use 0.1 uF in 0402 size to minimize inductance • Make VTT voltage decoupling close to termination resistors • Connect decoupling caps between VTT and ground • Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin • Verify the capacitive decoupling using the Intel Power Distribution Network Design Tool
Power	• Route GND and V_{CC} as planes • Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation • Route VTT as islands or 250-mil (6.35-mm) power traces • Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces
General Routing	All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Intel recommends that signals from the same net group always be routed on the same layer. • Use 45° angles (*not* 90° corners) • Avoid T-Junctions for critical nets or clocks • Avoid T-junctions greater than 250 mils (6.35 mm) • Disallow signals across split planes • Restrict routing other signals close to system reset signals • Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks

Related Information

Power Distribution Network

9.4.3. QDR-IV Layout Guidelines

Observe the following layout guidelines for your QDR-IV interface.

Parameter	Guidelines
General Routing	• If you must route signals of the same net group on different layers with the same impedance characteristic, simulate your worst case PCB trace tolerances to determine actual propagation delay differences. Typical layer-to-layer trace delay variations are on the order of 15 ps/inch. • Avoid T-junctions greater than 150 ps. • Match all signals within a given DQ group with a maximum skew of ±10 ps and route on the same layer.
Clock Routing	• Route clocks on inner layers with outer-layer run lengths held to less than 150 ps. • Clock signals should maintain a 10-mil (0.254 mm) spacing from other nets. • Clocks should maintain a length-matching between clock pairs of ±5 ps. • Differential clocks should maintain a length-matching between P and N signals of ±2 ps. • Space between different clock pairs should be at least three times the space between the traces of a differential pair.
Address and Command Routing	• - To minimize crosstalk, route address, bank address, and command signals on a different layer than the data signals. • Do not route the differential clock signals close to the address signals. • Keep the distance from the pin on the QDR-IV component to the stub termination resistor (VTT) to less than 50 ps for the address/command signal group. • - Route the mem_ck (CK/CK#) clocks and set as the target trace propagation delays for the address/command signal group. Match the CK/CK# clock to within ±50 ps of all the DK/DK# clocks for both ports. • - Route the address/control signal group ideally on the same layer as the mem_ck (CK/CK#) clocks, to within ±20 ps skew of the mem_ck (CK/CK#) traces.
Data Signals	• For port B only: Swap the polarity of the QKB and QKB# signals with respect to the polarity of the differential buffer inputs on the FPGA. Connect the positive leg of the differential input buffer on the FPGA to QDR-IV QKB# (negative) pin and vice-versa. Note that the port names at the top-level of the IP already reflect this swap (that is, mem_qkb is assigned to the negative buffer leg, and mem_qkb_n is assigned to the positive buffer leg). • For each port, route the DK/DK# write clock and QK/QK# read clock associated with a DQ group on the same PCB layer. Match these clock pairs to within ±5 ps. • For each port, set the DK/DK# or QK/QK# clock as the target trace propagation delay for the associated data signals (DQ). • For each port, route the data (DQ) signals for the DQ group ideally on the same layer as the associated QK/QK# and DK/DK# clocks to within ±10 ps skew of the target clock.
Maximum Trace Length	• Keep the maximum trace length of all signals from the FPGA to the QDR-IV components to 600 ps.
Spacing Guidelines	• Avoid routing two signal layers next to each other. Always make sure that the signals related to memory interface are routed between appropriate GND or power layers. • For Data and Data Strobe traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace. • For Address/Command/Control traces: Maintain at least 3H spacing between the edges (air-gap) of these traces, where H is the vertical distance to the closest return path for that particular trace. • For Clock (mem_CK) traces: Maintain at least 5H spacing between two clock pair or a clock pair and any other memory interface trace, where H is the vertical distance to the closest return path for that particular trace.
Trace Matching Guidance	The following layout approach is recommended, based on the preceding guidelines: 1. For port B only: Swap the polarity of the QKB and QKB# signals with respect to the polarity of the differential buffer inputs on the FPGA. Connect the positive leg of the differential input buffer on the FPGA to QDR-IV QKB# (negative) pin and vice-versa. Note

continued...

Parameter	Guidelines
	that the port names at the top-level of the IP already reflect this swap (that is, mem_qkb is assigned to the negative buffer leg, and mem_qkb_n is assigned to the positive buffer leg). 2. For each port, set the DK/DK# or QK/QK# clock as the target trace propagation delay for the associated data signals (DQ). 3. For each port, route the data (DQ) signals for the DQ group ideally on the same layer as the associated QK/QK# and DK/DK# clocks to within ±10 ps skew of the target clock. 4. Route the mem_ck (CK/CK#) clocks and set as the target trace propagation delays for the address/command signal group. Match the CK/CK# clock to within ±50 ps of all the DK/DK# clocks for both ports. 5. Route theaddress/control signal group ideally on the same layer as the mem_ck (CK/CK#) clocks, to within ±10 ps skew of the mem_ck (CK/CK#) traces.

9.4.4. Package Deskew

You should follow Intel's package deskew guidance.

Related Information

Package Deskew

9.4.5. Package Migration

Package delays can be different for the same pin in different packages. If you want to use multiple migratable packages in your system, you should compensate for package skew as described in this topic.The information in this topic applies to Intel Stratix 10 devices.

Scenario 1

Your PCB is designed for multiple migratable devices, but you have only one device with which to go to production.

Assume two migratable packages, device A and device B, and that you want to go to production with device A. Follow these steps:

1. Perform package deskew for device A.
2. Compile your design for device A, with the **Package Skew** option enabled.
3. Note the skews in the <core_name>.pin file for device A. Deskew these package skews with board trace lengths as described in the preceding examples.
4. Recompile your design for device A.
5. For device B, open the parameter editor and deselect the **Package Deskew** option.
6. Calculate board skew parameters, only taking into account the board traces for device B, and enter that value into the parameter editor for device B.
7. Regenerate the IP and recompile the design for device B.
8. Verify that timing requirements are met for both device A and device B.

Scenario 2

Your PCB is designed for multiple migratable devices, and you want to go to production with all of them.

Assume you have device A and device B, and plan to use both devices in production. Follow these steps:

1. Do not perform any package deskew compensation for either device.
2. Compile a Quartus Prime design for device A with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
3. Verify that the **Report DDR** timing report meets your timing requirements.
4. Compile a Quartus Prime design for device B with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
5. Verify that the **Report DDR** timing report meets your timing requirements.

9.4.6. Slew Rates

For optimum timing margins and best signal integrity for the address, command, and memory clock signals, you should generally use fast slew rates and external terminations.

In board simulation, fast slew rates may show a perceived signal integrity problem, such as reflections or a nonmonotonic waveform in the SSTL I/O switching region. Such indications may cause you to consider using slow slew rate options for either the address and command signals or the memory clock, or both.

If you set the **FPGA I/O tab parameter options ➤ Address/Command ➤ Slew Rate** and **Memory Clock ➤ Slew Rate** parameters to different values, a warning message appears: .

```
Warning: .emif_0: When the address/command signals and the memory clock signals
do not use the same slew rate setting, signals using the "Slow" setting are
delayed relative to signals using "Fast" setting. For accurate timing analysis,
you must perform I/O simulation and manually include the delay as board skew.
To avoid the issue, use the same slew rate setting for both address/command
signals and memory clock signals whenever possible.
```

Note: The warning message applies only to board-level simulation, and does not require any delay adjustments in the PCB design or Board tab parameter settings.

Due to limitations of the IBIS model correlation tolerance and the accuracy of the board simulation model, it is possible for signal integrity problems to appear when using fast slew rate during simulation but not occur during operation on hardware. If you observe a signal integrity problem during simulation with a fast slew rate, use an oscilloscope to view the signal at that point in hardware, to verify whether the problem exists on hardware, or only in simulation.

If the signal integrity problem exists on hardware as well as in simulation, using different slew rates for the address and command signals and the clock remains a valid approach, and the address and command calibration stage will help to improve the address and command to clock setup and hold time margins.

10. Intel Stratix 10 EMIF IP for RLDRAM 3

This chapter contains IP parameter descriptions, board skew equations, pin planning information, and board design guidance for Intel Stratix 10 external memory interfaces for RLDRAM 3.

10.1. Parameter Descriptions

The following topics describe the parameters available on each tab of the IP parameter editor, which you can use to configure your IP.

10.1.1. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: General

Table 317. Group: General / Interface

Display Name	Description
Configuration	Specifies the configuration of the memory interface. The available options depend on the protocol and the targeted FPGA product. (Identifier: PHY_RLD3_CONFIG_ENUM)

Table 318. Group: General / Clocks

Display Name	Description
Memory clock frequency	Specifies the **operating frequency** of the memory interface in MHz. If you change the memory frequency, you should update the memory latency parameters on the **Memory** tab and the memory timing parameters on the **Mem Timing** tab. (Identifier: PHY_RLD3_MEM_CLK_FREQ_MHZ)
Use recommended PLL reference clock frequency	Specifies that the PLL reference clock frequency is automatically calculated for best performance. *If you want to specify a different PLL reference clock frequency, uncheck the check box for this parameter.* (Identifier: PHY_RLD3_DEFAULT_REF_CLK_FREQ)
PLL reference clock frequency	This parameter tells the IP what PLL reference clock frequency the user will supply. Users must select a valid PLL reference clock frequency from the list. The values in the list can change when the memory interface frequency changes and/or the clock rate of user logic changes. It is recommended to use the fastest possible PLL reference clock frequency because it leads to better jitter performance. Selection is required only if the user does not check the "Use recommended PLL reference clock frequency" option. (Identifier: PHY_RLD3_USER_REF_CLK_FREQ_MHZ)
PLL reference clock jitter	Specifies the **peak-to-peak jitter** on the PLL reference clock source. The clock source of the PLL reference clock must meet or exceed the following jitter requirements: 10ps peak to peak, or 1.42ps RMS at 1e-12 BER, 1.22ps at 1e-16 BER. (Identifier: PHY_RLD3_REF_CLK_JITTER_PS)

continued...

Display Name	Description
Clock rate of user logic	Specifies the relationship between the user logic clock frequency and the memory clock frequency. For example, if the memory clock sent from the FPGA to the memory device is toggling at 800MHz, a quarter-rate interface means that the user logic in the FPGA runs at 200MHz. The list of available options is dependent on the memory protocol and device family. (Identifier: PHY_RLD3_RATE_ENUM)
Core clocks sharing	When a design contains multiple interfaces of the same protocol, rate, frequency, and PLL reference clock source, they can share a common set of core clock domains. By sharing core clock domains, they **reduce clock network usage and avoid clock synchronization logic between the interfaces**. To share core clocks, denote one of the interfaces as **"Master"**, and the remaining interfaces as **"Slave"**. In the RTL, connect the `clks_sharing_master_out` signal from the master interface to the `clks_sharing_slave_in` signal of all the slave interfaces. Both master and slave interfaces still expose their own output clock ports in the RTL (for example, `emif_usr_clk`, `afi_clk`), but the physical signals are equivalent, hence it does not matter whether a clock port from a master or a slave is used. *As the combined width of all interfaces sharing the same core clock increases, you may encounter timing closure difficulty for transfers between the FPGA core and the periphery.* (Identifier: PHY_RLD3_CORE_CLKS_SHARING_ENUM)
Export clks_sharing_slave_out to facilitate multi-slave connectivity	When more than one slave exist, you can either connect the clks_sharing_master_out interface from the master to the clks_sharing_slave_in interface of all the slaves (i.e. one-to-many topology), OR, you can connect the clks_sharing_master_out interface to one slave, and connect the clks_sharing_slave_out interface of that slave to the next slave (i.e. daisy-chain topology). Both approaches produce the same result. The daisy-chain approach may be easier to achieve in the Platform Designer tool, whereas the one-to-many approach may be more intuitive. (Identifier: PHY_RLD3_CORE_CLKS_SHARING_EXPOSE_SLAVE_OUT)
Specify additional core clocks based on existing PLL	Displays additional parameters allowing you to create additional output clocks based on the existing PLL. This parameter **provides an alternative clock-generation mechanism for when your design exhausts available PLL resources**. The additional output clocks that you create can be fed into the core. Clock signals created with this parameter are synchronous to each other, but asynchronous to the memory interface core clock domains (such as `emif_usr_clk` or `afi_clk`). *You must follow proper clock-domain-crossing techniques when transferring data between clock domains.* (Identifier: PLL_ADD_EXTRA_CLKS)

Table 319. Group: General / Clocks / Additional Core Clocks

Display Name	Description
Number of additional core clocks	Specifies the number of additional output clocks to create from the PLL. (Identifier: PLL_USER_NUM_OF_EXTRA_CLKS)

Table 320. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_0

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_5)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_5)

Table 321. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_1

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_6)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_6)

Table 322. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_2

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_7)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_7)

Table 323. Group: General / Clocks / Additional Core Clocks / pll_extra_clk_3

Display Name	Description
Frequency	Specifies the frequency of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_FREQ_MHZ_GUI_8)
Phase shift	Specifies the phase shift of the core clock signal. (Identifier: PLL_EXTRA_CLK_ACTUAL_PHASE_PS_GUI_8)

10.1.2. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: FPGA I/O

You should use Hyperlynx* or similar simulators to determine the best settings for your board. Refer to the EMIF Simulation Guidance wiki page for additional information.

Table 324. Group: FPGA I/O / FPGA I/O Settings

Display Name	Description
Voltage	The voltage level for the I/O pins driving the signals between the memory device and the FPGA memory interface. (Identifier: PHY_RLD3_IO_VOLTAGE)
Use default I/O settings	Specifies that a legal set of I/O settings are automatically selected. The default I/O settings are not necessarily optimized for a specific board. *To achieve optimal signal integrity, perform I/O simulations with IBIS models and enter the I/O settings manually, based on simulation results*. (Identifier: PHY_RLD3_DEFAULT_IO)

Table 325. Group: FPGA I/O / FPGA I/O Settings / Address/Command

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the address/command pins of the memory interface. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_RLD3_USER_AC_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_RLD3_USER_AC_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_RLD3_USER_AC_SLEW_RATE_ENUM)

Table 326. Group: FPGA I/O / FPGA I/O Settings / Memory Clock

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the memory clock pins. The selected I/O standard configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_RLD3_USER_CK_IO_STD_ENUM)
Output mode	This parameter allows you to change the current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_RLD3_USER_CK_MODE_ENUM)
Slew rate	Specifies the slew rate of the address/command output pins. The slew rate (or edge rate) describes how quickly the signal can transition, measured in voltage per unit time. *Perform board simulations to determine the slew rate that provides the best eye opening for the address and command signals.* (Identifier: PHY_RLD3_USER_CK_SLEW_RATE_ENUM)

Table 327. Group: FPGA I/O / FPGA I/O Settings / Data Bus

Display Name	Description
I/O standard	Specifies the I/O electrical standard for the data and data clock/strobe pins of the memory interface. The selected I/O standard option configures the circuit within the I/O buffer to match the industry standard. (Identifier: PHY_RLD3_USER_DATA_IO_STD_ENUM)
Output mode	This parameter allows you to change the output current drive strength or termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_RLD3_USER_DATA_OUT_MODE_ENUM)
Input mode	This parameter allows you to change the input termination settings for the selected I/O standard. *Perform board simulation with IBIS models to determine the best settings for your design.* (Identifier: PHY_RLD3_USER_DATA_IN_MODE_ENUM)

Table 328. Group: FPGA I/O / FPGA I/O Settings / PHY Inputs

Display Name	Description
PLL reference clock I/O standard	Specifies the I/O standard for the PLL reference clock of the memory interface. (Identifier: PHY_RLD3_USER_PLL_REF_CLK_IO_STD_ENUM)
RZQ I/O standard	Specifies the I/O standard for the RZQ pin used in the memory interface. (Identifier: PHY_RLD3_USER_RZQ_IO_STD_ENUM)

10.1.3. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Memory

Table 329. Group: Memory / Topology

Display Name	Description
DQ width per device	Specifies number of DQ pins per RLDRAM3 device. Available widths for DQ are x18 and x36. (Identifier: MEM_RLD3_DQ_PER_DEVICE)
Enable DM pins	Indicates whether the interface uses the DM pins. If enabled, one DM pin per write data group is added. (Identifier: MEM_RLD3_DM_EN)
Enable width expansion	Indicates whether to combine two memory devices to double the data bus width. With two devices, the interface supports a width expansion configuration up to 72-bits. For width expansion configuration, the address and control signals are routed to 2 devices. (Identifier: MEM_RLD3_WIDTH_EXPANDED)
Enable depth expansion using twin die package	Indicates whether to combine two RLDRAM3 devices to double the address space, resulting in more density. (Identifier: MEM_RLD3_DEPTH_EXPANDED)
Address width	Number of address pins. (Identifier: MEM_RLD3_ADDR_WIDTH)
Bank address width	Number of bank address pins (Identifier: MEM_RLD3_BANK_ADDR_WIDTH)

Table 330. Group: Memory / Mode Register Settings

Display Name	Description
tRC	Determines the mode register setting that controls the tRC(**activate to activate timing parameter**). *Refer to the tRC table in the memory vendor data sheet.* Set the tRC according to the memory speed grade and data latency. (Identifier: MEM_RLD3_T_RC_MODE_ENUM)
Data Latency	Determines the mode register setting that controls the data latency. **Sets both READ and WRITE latency (RL and WL).** (Identifier: MEM_RLD3_DATA_LATENCY_MODE_ENUM)
Output drive	Determines the mode register setting that controls the output drive setting. (Identifier: MEM_RLD3_OUTPUT_DRIVE_MODE_ENUM)
ODT	Determines the mode register setting that controls the ODT setting. (Identifier: MEM_RLD3_ODT_MODE_ENUM)
AREF protocol	**Determines the mode register setting that controls the AREF protocol setting.** The **AUTO REFRESH (AREF)** protocol is selected by setting mode register 1. There are two ways in which AREF commands can be issued to the RLDRAM, the memory controller can either issue bank address-controlled or multibank AREF commands. Multibank refresh protocol allows for the simultaneous refreshing of a row in up to four banks (Identifier: MEM_RLD3_AREF_PROTOCOL_ENUM)
Burst length	Determines the mode register setting that controls the burst length. (Identifier: MEM_RLD3_BL)
Write protocol	Determines the mode register setting that controls the write protocol setting. When multiple bank (dual bank or quad bank) is selected, identical data is written to multiple banks. (Identifier: MEM_RLD3_WRITE_PROTOCOL_ENUM)

10.1.4. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Mem Timing

These parameters should be read from the table in the datasheet associated with the speed bin of the memory device (not necessarily the frequency at which the interface is running).

Table 331. Group: Mem Timing

Display Name	Description
Speed bin	The speed grade of the memory device used. This parameter refers to the maximum rate at which the memory device is specified to run. (Identifier: MEM_RLD3_SPEEDBIN_ENUM)
tDS (base)	tDS(base) refers to the **setup time for the Data (DQ) bus** before the rising edge of the DQS strobe. (Identifier: MEM_RLD3_TDS_PS)
tDS (base) AC level	tDS (base) AC level refers to the **voltage level which the data bus must cross and remain above during the setup margin window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire setup period. (Identifier: MEM_RLD3_TDS_AC_MV)
tDH (base)	tDH (base) refers to the **hold time for the Data (DQ) bus** after the rising edge of CK. (Identifier: MEM_RLD3_TDH_PS)
tDH (base) DC level	tDH (base) DC level refers to the **voltage level which the data bus must not cross during the hold window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire hold period. (Identifier: MEM_RLD3_TDH_DC_MV)
tQKQ_max	tQKQ_max describes the **maximum skew** between the **read strobe (QK) clock edge** to the **data bus (DQ/DINV) edge**. (Identifier: MEM_RLD3_TQKQ_MAX_PS)
tQH	tQH specifies the **output hold time for the DQ/DINV** in relation to QK. (Identifier: MEM_RLD3_TQH_CYC)
tCKDK_max	tCKDK_max refers to the **maximum skew** from the **memory clock (CK)** to the **write strobe (DK)**. (Identifier: MEM_RLD3_TCKDK_MAX_CYC)
tCKDK_min	tCKDK_min refers to the **minimum skew** from the **memory clock (CK)** to the **write strobe (DK)**. (Identifier: MEM_RLD3_TCKDK_MIN_CYC)
tCKQK_max	tCKQK_max refers to the **maximum skew** from the **memory clock (CK)** to the **read strobe (QK)**. (Identifier: MEM_RLD3_TCKQK_MAX_PS)
tIS (base)	tIS (base) refers to the **setup time for the Address/Command/Control (A) bus** to the rising edge of CK. (Identifier: MEM_RLD3_TIS_PS)
tIS (base) AC level	tIS (base) AC level refers to the **voltage level which the address/ command signal must cross and remain above during the setup margin window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire setup period. (Identifier: MEM_RLD3_TIS_AC_MV)
tIH (base)	tIH (base) refers to the **hold time for the Address/Command (A) bus** after the rising edge of CK. Depending on what AC level the user has chosen for a design, the hold margin can vary (this variance will be automatically determined when the user chooses the "**tIH (base) AC level**"). (Identifier: MEM_RLD3_TIH_PS)
tIH (base) DC level	tIH (base) DC level refers to the **voltage level which the address/ command signal must not cross during the hold window**. The signal is considered stable only if it remains above this voltage level (for a logic 1) or below this voltage level (for a logic 0) for the entire hold period. (Identifier: MEM_RLD3_TIH_DC_MV)

10.1.5. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Board

Table 332. Group: Board / Intersymbol Interference/Crosstalk

Display Name	Description
Use default ISI/crosstalk values	You can enable this option to use default intersymbol interference and crosstalk values for your topology. Note that the default values are not optimized for your board. *For optimal signal integrity, it is recommended that you do not enable this parameter, but instead perform I/O simulation using IBIS models and Hyperlynx)*, and manually enter values based on your simulation results, instead of using the default values.* (Identifier: BOARD_RLD3_USE_DEFAULT_ISI_VALUES)
Address and command ISI/crosstalk	The address and command window reduction due to ISI and crosstalk effects. The number to be entered is the **total loss of margin on both the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_RLD3_USER_AC_ISI_NS)
QK/QK# ISI/crosstalk	QK/QK# ISI/crosstalk describes the reduction of the read data window due to intersymbol interference and crosstalk effects on the QK/QK# signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on both the setup and hold sides (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_RLD3_USER_RCLK_ISI_NS)
Read DQ ISI/crosstalk	The reduction of the read data window due to ISI and crosstalk effects on the DQ signal when driven by the memory device during a read. The number to be entered is the **total loss of margin on the setup and hold side (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_RLD3_USER_RDATA_ISI_NS)
DK/DK# ISI/crosstalk	DK/DK# ISI/crosstalk describes the reduction of the write data window due to intersymbol interference and crosstalk effects on the DK/DK# signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold side (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_RLD3_USER_WCLK_ISI_NS)
Write DQ ISI/crosstalk	The reduction of the write data window due to ISI and crosstalk effects on the DQ signal when driven by the FPGA during a write. The number to be entered is the **total loss of margin on the setup and hold side (measured loss on the setup side + measured loss on the hold side)**. *Refer to the EMIF Simulation Guidance wiki page for additional information.* (Identifier: BOARD_RLD3_USER_WDATA_ISI_NS)

Table 333. Group: Board / Board and Package Skews

Display Name	Description
Package deskewed with board layout (QK group)	If you are compensating for package skew on the QK bus in the board layout (hence checking the box here), please **include package skew in calculating the following board skew parameters.** (Identifier: BOARD_RLD3_IS_SKEW_WITHIN_QK_DESKEWED)
Maximum board skew within QK group	Maximum board skew within QK group refers to the largest skew between all DQ and DM pins in a QK group. This value can affect the read capture and write margins. (Identifier: BOARD_RLD3_BRD_SKEW_WITHIN_QK_NS)
Maximum system skew within QK group	The largest skew between all DQ and DM pins in a QK group. Enter combined board and package skew. This value affects the read capture and write margins. (Identifier: BOARD_RLD3_PKG_BRD_SKEW_WITHIN_QK_NS)

continued...

Display Name	Description
Package deskewed with board layout (address/command bus)	Enable this parameter if you are compensating for package skew on the address, command, control, and memory clock buses in the board layout. **Include package skew in calculating the following board skew parameters.** (Identifier: BOARD_RLD3_IS_SKEW_WITHIN_AC_DESKEWED)
Maximum board skew within address/ command bus	The largest skew between the address and command signals. Enter the board skew only; package skew is calculated automatically, based on the memory interface configuration, and added to this value. (Identifier: BOARD_RLD3_BRD_SKEW_WITHIN_AC_NS)
Maximum system skew within address/ command bus	Maximum system skew within address/command bus refers to the largest skew between the address and command signals. (Identifier: BOARD_RLD3_PKG_BRD_SKEW_WITHIN_AC_NS)
Average delay difference between DK and CK	This parameter describes the average delay difference between the DK signals and the CK signal, calculated by averaging the longest and smallest DK trace delay minus the CK trace delay. Positive values represent DK signals that are longer than CK signals and negative values represent DK signals that are shorter than CK signals. (Identifier: BOARD_RLD3_DK_TO_CK_SKEW_NS)
Maximum delay difference between devices	This parameter describes the largest propagation delay on the DQ signals between ranks. For example, in a two-rank configuration where devices are placed in series, there is an extra propagation delay for DQ signals going to and coming back from the furthest device compared to the nearest device. *This parameter is only applicable when there is more than one rank.* (Identifier: BOARD_RLD3_SKEW_BETWEEN_DIMMS_NS)
Maximum skew between DK groups	This parameter describes the largest skew between DK signals in different DK groups. (Identifier: BOARD_RLD3_SKEW_BETWEEN_DK_NS)
Average delay difference between address/command and CK	The average delay difference between the address/command signals and the CK signal, calculated by averaging the longest and smallest address/ command signal trace delay minus the maximum CK trace delay. Positive values represent address and command signals that are longer than CK signals and negative values represent address and command signals that are shorter than CK signals. (Identifier: BOARD_RLD3_AC_TO_CK_SKEW_NS)
Maximum CK delay to device	The maximum CK delay to device refers to the delay of the longest CK trace from the FPGA to any device. (Identifier: BOARD_RLD3_MAX_CK_DELAY_NS)
Maximum DK delay to device	The maximum DK delay to device refers to the delay of the longest DK trace from the FPGA to any device. (Identifier: BOARD_RLD3_MAX_DK_DELAY_NS)

10.1.6. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Diagnostics

Table 334. Group: Diagnostics / Simulation Options

Display Name	Description
Calibration mode	Specifies whether to **skip memory interface calibration** during simulation, or to **simulate the full calibration** process. Simulating the full calibration process can take hours (or even days), depending on the width and depth of the memory interface. You can achieve much faster simulation times by skipping the calibration process, but that is only expected to work when the memory model is ideal and the interconnect delays are zero.

continued...

Display Name	Description
	If you enable this parameter, the interface still performs some memory initialization before starting normal operations. Abstract PHY is supported with skip calibration. (Identifier: DIAG_RLD3_SIM_CAL_MODE_ENUM)
Abstract phy for fast simulation	Specifies that the system use Abstract PHY for simulation. **Abstract PHY replaces the PHY with a model for fast simulation and can reduce simulation time by 3-10 times.** Abstract PHY is available for certain protocols and device families, and only when you select **Skip Calibration**. (Identifier: DIAG_RLD3_ABSTRACT_PHY)
Show verbose simulation debug messages	This option allows adjusting the verbosity of the simulation output messages. (Identifier: DIAG_RLD3_SIM_VERBOSE)

Table 335. Group: Diagnostics / Calibration Debug Options

Display Name	Description
Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies the connectivity of an Avalon slave interface for use by the Quartus Prime EMIF Debug Toolkit or user core logic. If you set this parameter to "**Disabled**", no debug features are enabled. If you set this parameter to "**Export**", an Avalon slave interface named "`cal_debug`" is exported from the IP. To use this interface with the EMIF Debug Toolkit, you must instantiate and connect an EMIF debug interface IP core to it, or connect it to the `cal_debug_out` interface of another EMIF core. If you select "**Add EMIF Debug Interface**", an EMIF debug interface component containing a JTAG Avalon Master is connected to the debug port, allowing the core to be accessed by the EMIF Debug Toolkit. *Only one EMIF debug interface should be instantiated per I/O column.* You can chain additional EMIF or PHYLite cores to the first by enabling the "**Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option for all cores in the chain, and selecting "**Export**" for the "**Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**" option on all cores after the first. (Identifier: DIAG_RLD3_EXPORT_SEQ_AVALON_SLAVE)
Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port	Specifies that the IP export an Avalon-MM master interface (`cal_debug_out`) which can connect to the `cal_debug` interface of other EMIF cores residing in the same I/O column. **This parameter applies only if the EMIF Debug Toolkit or On-Chip Debug Port is enabled.** *Refer to the Debugging Multiple EMIFs wiki page for more information about debugging multiple EMIFs.* (Identifier: DIAG_RLD3_EXPORT_SEQ_AVALON_MASTER)
First EMIF Instance in the Avalon Chain	If selected, this EMIF instance will be the head of the Avalon interface chain connected to the master. For simulation purposes it is needed to identify the first EMIF instance in the avalon Chain. (Identifier: DIAG_RLD3_EXPORT_SEQ_AVALON_HEAD_OF_CHAIN)
Interface ID	Identifies interfaces within the I/O column, for use by the EMIF Debug Toolkit and the On-Chip Debug Port. Interface IDs should be unique among EMIF cores within the same I/O column. If the **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** parameter is set to **Disabled**, the interface ID is unused. (Identifier: DIAG_RLD3_INTERFACE_ID)
Use Soft NIOS Processor for On-Chip Debug	Enables a soft Nios processor as a peripheral component to access the **On-Chip Debug Port**. *Only one interface in a column can activate this option.* (Identifier: DIAG_SOFT_NIOS_MODE)

Table 336. Group: Diagnostics / Example Design

Display Name	Description
Number of core clocks sharing slaves to instantiate in the example design	Specifies the number of core clock sharing slaves to instantiate in the example design. This parameter applies only if you set the "**Core clocks sharing**" parameter in the "**General**" tab to "**Master**" or "**Slave**". (Identifier: DIAG_RLD3_EX_DESIGN_NUM_OF_SLAVES)
Enable In-System-Sources-and-Probes	Enables In-System-Sources-and-Probes in the example design for *common debug signals, such as calibration status or example traffic generator per-bit status*. This parameter must be enabled if you want to do driver margining using the EMIF Debug Toolkit. (Identifier: DIAG_RLD3_EX_DESIGN_ISSP_EN)

Table 339. Group: Diagnostics / Miscellaneous

Display Name	Description
Export PLL lock signal	Specifies whether to export the pll_locked signal at the IP top-level to indicate status of PLL. (Identifier: DIAG_EXPORT_PLL_LOCKED)

10.1.7. Intel Stratix 10 EMIF IP RLDRAM 3 Parameters: Example Designs

Table 340. Group: Example Designs / Available Example Designs

Display Name	Description
Select design	Specifies the *creation of a full Quartus Prime project, instantiating an external memory interface and an example traffic generator, according to your parameterization*. After the design is created, you can specify the target device and pin location assignments, run a full compilation, verify timing closure, and test the interface on your board using the programming file created by the Quartus Prime assembler. The '**Generate Example Design**' button lets you generate simulation or synthesis file sets. (Identifier: EX_DESIGN_GUI_RLD3_SEL_DESIGN)

Table 341. Group: Example Designs / Example Design Files

Display Name	Description
Simulation	Specifies that the '**Generate Example Design**' button create all necessary file sets for simulation. Expect a short additional delay as the file set is created. *If you do not enable this parameter, simulation file sets are not created.* Instead, the output directory will contain the `ed_sim.qsys` file which holds Qsys details of the simulation example design, and a `make_sim_design.tcl` file with other corresponding `tcl` files. You can run `make_sim_design.tcl` from a command line to generate the simulation example design. The generated example designs for various simulators are **stored in the /sim sub-directory**. (Identifier: EX_DESIGN_GUI_RLD3_GEN_SIM)
Synthesis	Specifies that the '**Generate Example Design**' button create all necessary file sets for synthesis. Expect a short additional delay as the file set is created. *If you do not enable this parameter, synthesis file sets are not created.* Instead, the output directory will contain the `ed_synth.qsys` file which holds Qsys details of the synthesis example design, and a `make_qii_design.tcl` script with other corresponding `tcl` files. You can run `make_qii_design.tcl` from a command line to generate the synthesis example design. The generated example design is **stored in the /qii sub-directory**. (Identifier: EX_DESIGN_GUI_RLD3_GEN_SYNTH)

Table 342. Group: Example Designs / Generated HDL Format

Display Name	Description
Simulation HDL format	This option lets you choose the format of HDL in which generated simulation files are created. (Identifier: EX_DESIGN_GUI_RLD3_HDL_FORMAT)

Table 343. Group: Example Designs / Target Development Kit

Display Name	Description
Select board	Specifies that *when you select a development kit with a memory module, the generated example design contains all settings and fixed pin assignments to run on the selected board. You must select a development kit preset to generate a working example design for the specified development kit*. Any IP settings not applied directly from a development kit preset will not have guaranteed results when testing the development kit. To exclude hardware support of the example design, select '**none**' from the '**Select board**' pull down menu. When you apply a development kit preset, all IP parameters are automatically set appropriately to match the selected preset. If you want to save your current settings, you should do so before you apply the preset. You can save your settings under a different name using **File->Save as**. (Identifier: EX_DESIGN_GUI_RLD3_TARGET_DEV_KIT)
PARAM_EX_DESIGN_PREV_PRESET_NAME	PARAM_EX_DESIGN_PREV_PRESET_DESC (Identifier: EX_DESIGN_GUI_RLD3_PREV_PRESET)

10.2. Board Skew Equations

The following table presents the underlying equations for the board skew parameters.

10.2.1. Equations for RLDRAM 3 Board Skew Parameters

Table 344. Board Skew Parameter Equations

Parameter	Description/Equation
Maximum CK delay to device	The delay of the longest CK trace from the FPGA to any device. $\max_n(CK_n PathDelay)$ where *n* is the number of memory clocks. For example, the maximum CK delay for two pairs of memory clocks is expressed by the following equation: $\max_2(CK_1 PathDelay,\ CK_2 PathDelay)$
Maximum DK delay to device	The delay of the longest DK trace from the FPGA to any device. $\max_n\left(DK_n PathDelay\right)$ where n is the number of DK. For example, the maximum DK delay for two DK is expressed by the following equation: $\max_2\left(DK_1 PathDelay,\ DK_2 PathDelay\right)$
Average delay difference between DK and CK	The average delay difference between the DK signals and the CK signal, calculated by averaging the longest and smallest DK delay minus the CK delay. Positive values represent DK signals that are longer than CK signals and negative values represent DK signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the DK signals to have appropriate setup and hold margins.

continued...

Parameter	Description/Equation
	$\frac{\max_{n,\,m}\left(CK_nPathDelay - DK_mPathDelay\right) + \min_{n,\,m}\left(CK_nPathDelay - DK_mPathDelay\right)}{2}$ where n is the number of memory clocks and m is the number of DK.
Maximum system skew within address/command bus	$(MaxAC - MinAC)$ The largest skew between the address and command signals. Enter combined board and package skew.
Average delay difference between address/command and CK	The average delay difference between the address and command signals and the CK signal, calculated by averaging the longest and smallest Address/Command signal delay minus the CK delay. Positive values represent address and command signals that are longer than CK signals and negative values represent address and command signals that are shorter than CK signals. The Quartus Prime software uses this skew to optimize the delay of the address and command signals to have appropriate setup and hold margins. $\frac{\sum_{n=1}^{n=n}\left[\left(\frac{\text{LongestACPathDelay} + \text{ShortestACPathDelay}}{2}\right) - \text{CK}_n\text{PathDelay}\right]}{n}$
Maximum board skew within QK group	The largest skew between all DQ and DM pins in a QK group. Enter your board skew only. Package skew is calculated automatically, based on the memory interface configuration, and added to this value. This value affects the read capture and write margins. $\max_n(\max DQ_n - \min DQ_n)$ where n is the number of DQ.
Maximum skew between DK groups	The largest skew between DK signals in different DK groups. $\max_n(\max DK_n - \min DK_n)$ where n is the number of DQ.

10.3. Pin and Resource Planning

The following topics provide guidelines on pin placement for external memory interfaces.

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

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

Once all the requirements are known for your external memory interface, you can begin planning your system.

10.3.1. Interface Pins

DQS (data strobe or data clock) and DQ (data) pins are listed for EMIF supported banks in the device pin tables and are fixed at specific locations in the device. You must adhere to these pin locations to optimize routing, minimize skew, and maximize margins. Always check the device pin table for the actual locations of the DQS and DQ pins, and the EMIF pin table for location of address and control pins.

Pin tables are available here: https://www.intel.com/content/www/us/en/programmable/support/literature/lit-dp.html?1.

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

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

- Maximum possible interface width in any particular device is limited by the number of DQS groups available.
- Sufficient clock networks are available to the interface PLL as required by the IP.
- Sufficient spare pins exist within the chosen bank or side of the device to include all other address and command, and clock pin placement requirements.

Note: The greater the number of banks, the greater the skew, hence Intel recommends that you always generate a test project of your desired configuration and confirm that it meets timing.

10.3.1.1. Estimating Pin Requirements

You should use the Intel Quartus Prime software for final pin fitting. However, you can estimate whether you have enough pins for your memory interface using the EMIF Device Selector on www.altera.com, or perform the following steps:

1. Determine how many read/write data pins are associated per data strobe or clock pair.
2. Calculate the number of other memory interface pins needed, including any other clocks (write clock or memory system clock), address, command, and RZQ. Refer to the External Memory Interface Pin Table to determine necessary Address/Command/Clock pins based on your desired configuration.
3. Calculate the total number of I/O banks required to implement the memory interface, given that an I/O bank supports up to 48 GPIO pins.

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

Related Information

Intel FPGA IP for External Memory Interfaces - Support Center

10.3.1.2. Maximum Number of Interfaces

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

Unless otherwise noted, the calculation for the maximum number of interfaces is based on independent interfaces where the address or command pins are not shared.

Note: You may need to share PLL clock outputs depending on your clock network usage.

For interface information for Intel Stratix 10, consult the EMIF Device Selector on www.altera.com.

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

Related Information

- Intel FPGA IP for External Memory Interfaces - Support Center
- Intel Stratix 10 EMIF Architecture: PLL Reference Clock Networks on page 21
- External Memory Interface Device Selector
- Intel Quartus Prime Pro Edition Handbook

10.3.1.3. FPGA Resources

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

10.3.1.4. OCT

You require one OCT calibration block if you are using an FPGA OCT calibrated series, parallel, or dynamic termination for any I/O in your design. You can select any available OCT calibration block—it need not be within the same bank or side of the device as the memory interface pins. The only requirement is that the I/O bank where you place the OCT calibration block must use the same V_{CCIO} voltage as the memory interface.

The OCT calibration block uses a single R_{ZQ} pin. The R_{ZQ} pin in Intel Stratix 10 devices can be used as a general purpose I/O pin when it is not used to support OCT, provided the signal conforms to the bank voltage requirements.

10.3.1.5. PLL

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

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

10.3.1.6. Pin Guidelines for Intel Stratix 10 EMIF IP

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

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

- The numeric component of the `Bank Number` value identifies the I/O column, while the letter represents the I/O bank.
- The `Index within I/O Bank` value falls within one of the following ranges: 0 to 11, 12 to 23, 24 to 35, or 36 to 47, and represents I/O lanes 1, 2, 3, and 4, respectively.
- To determine if I/O banks are adjacent, you can refer to the I/O Pin Counts tables located in the *Intel Stratix 10 General Purpose I/O User Guide*. You can always assume I/O banks are adjacent within an I/O column except in the following conditions:
  - When an I/O bank is not bonded out on the package (contains the '-' symbol in the I/O table).
  - An I/O bank does not contain 48 pins, indicating it is only partially bonded out.
- The pairing pin for an I/O pin is located in the same I/O bank. You can identify the pairing pin by adding one to its `Index within I/O Bank` number (if it is an even number), or by subtracting one from its `Index within I/O Bank` number (if it is an odd number).

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

10.3.1.6.1. General Guidelines

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

If you are using the hard memory controller, you should employ the relative pin locations defined in the *<variation_name>*/altera_emif_arch_nd_*version number*/*<synth|sim>*/*<variation_name>*_altera_emif_arch_nd_*version number*_*<unique ID>*_readme.txt file, which is generated with your IP.

Note:

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

Observe the following general guidelines when placing pins for your Intel Stratix 10 external memory interface:

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

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

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

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

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

   b. QDR-IV requires that the polarity of all QKB/QKB# pins be swapped with respect to the polarity of the differential buffer inputs on the FPGA to ensure correct data capture on port B. All QKB pins on the memory device must be connected to the negative pins of the input buffers on the FPGA side, and all QKB# pins on the memory device must be connected to the positive pins of the input buffers on the FPGA side. Notice that the port names at the top-level of the IP already reflect this swap (that is, `mem_qkb` is assigned to the negative buffer leg, and `mem_qkb_n` is assigned to the positive buffer leg).
10. You can implement two x4 DQS groups with a single I/O lane. The pin table specifies which pins within an I/O lane can be used for the two pairs of DQS and DQS# signals. In addition, for x4 DQS groups you must observe the following rules:
    - There must be an even number of x4 groups in an external memory interface.
    - DQS group 0 and DQS group 1 must be placed in the same I/O lane. Similarly, DQS group 2 and group 3 must be in the same I/O lane. Generally, DQS group *X* and DQS group *X*+1 must be in the same I/O lane, where *X* is an even number.
11. You should place the write data groups according to the DQS grouping in the pin table and Pin Planner. Output-only data clocks for QDR II, QDR II+, and QDR II+ Extreme, and RLDRAM 3 protocols need not be placed on DQS/DQSn pins, but must be placed on a differential pin pair. They must be placed in the same I/O bank as the corresponding DQS group.

    Note: For RLDRAM 3, x36 device, `DQ[8:0]` and `DQ[26:18]` are referenced to `DK0/DK0#`, and `DQ[17:9]` and `DQ[35:27]` are referenced to `DK1/DK1#`.
12. For protocols and topologies with bidirectional data pins where a write data group consists of multiple read data groups, you should place the data groups and their respective write and read clock in the same bank to improve I/O timing.

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

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

Note:

1. x4 mode does not support DM/DBI, or Intel Stratix 10 EMIF IP for HPS.
2. If you are using an Intel Stratix 10 EMIF IP-based RLDRAM 3 external memory interface, you should ensure that all the pins in a DQS group (that is, DQ, DM, DK, and QK) are placed in the same I/O bank. This requirement facilitates timing closure and is necessary for successful compilation of your design.

I/O Banks Selection

- For each memory interface, select consecutive I/O banks. (That is, select banks that contain the same column number and letter before or after the respective I/O bank letter.)
- A memory interface can only span across I/O banks in the same I/O column.
- The number of I/O banks that you require depends on the memory interface width.
- In some device packages, the number of I/O pins in some LVDS I/O banks is less that 48 pins.

Address/Command Pins Location

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

CK Pins Assignment

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

- If the number of I/O banks is odd, assign one CK pin to the middle I/O bank.
- If the number of I/O banks is even, assign the CK pin to either of the middle two I/O banks.

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

PLL Reference Clock Pin Placement

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

- If you are sharing the PLL reference clock pin between several interfaces, the I/O banks must be adjacent. (That is, the banks must contain the same column number and letter before or after the respective I/O bank letter.)

The Intel Stratix 10 external memory interface IP does not support PLL cascading.

RZQ Pin Placement

You may place the R_{ZQ} pin in any I/O bank in an I/O column with the correct V_{CCIO} and V_{CCPT} for the memory interface I/O standard in use. However, the recommended location is in the address/command I/O bank, for greater flexibility during debug if a narrower interface project is required for testing.

DQ and DQS Pins Assignment

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

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

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

Sharing an I/O Bank Across Multiple Interfaces

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

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

10.3.1.6.2. RLDRAM 3 Commands and Addresses

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

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

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

Note: The RLDRAM 3 external memory interface IP does not support multiplexed addressing.

10.3.1.6.3. RLDRAM 3 Clock Signals

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

Instead of a strobe, RLDRAM 3 devices use two sets of free-running differential clocks to accompany the data. The DK and DK# clocks are the differential input data clocks used during writes while the QK or QK# clocks are the output data clocks used during reads. Even though QK and QK# signals are not differential signals according to the RLDRAM 3 data sheet, Micron treats these signals as such for their testing and characterization. Each pair of DK and DK#, or QK and QK# clocks are associated with either 9 or 18 data bits.

The exact clock-data relationships are as follows:

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

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

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

10.3.1.6.4. RLDRAM 3 Data, DM and QVLD Signals

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

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

Figure 84. Edge-aligned DQ and QK Relationship During RLDRAM 3 Read

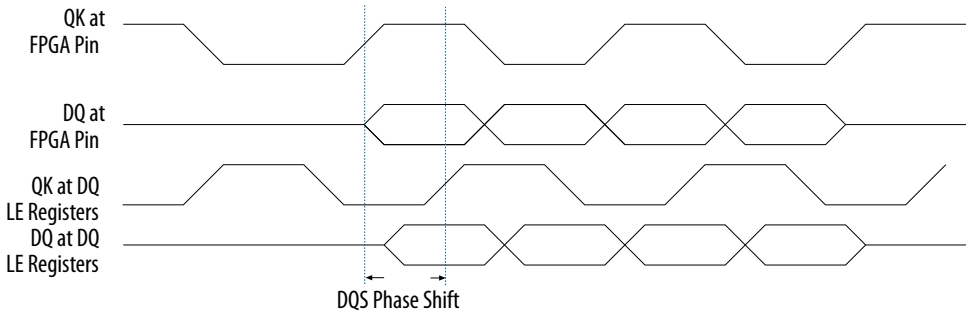
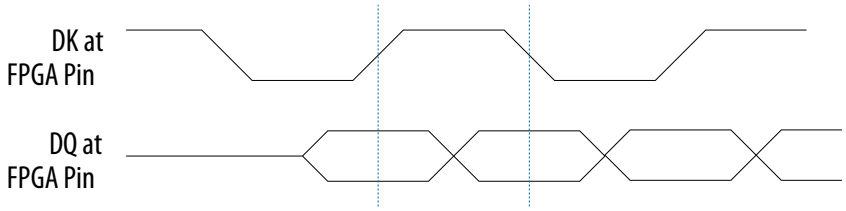


Figure 85. Center-aligned DQ and DK Relationship During RLDRAM 3 Write



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

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

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

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

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

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

Note: The Intel FPGA external memory interface IP does not use the QVLD signal.

10.3.1.7. Resource Sharing Guidelines (Multiple Interfaces)

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

PLL Reference Clock Pin

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

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

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

Note: You can place the `pll_ref_clk` pin in the address and command I/O bank or in a data I/O bank, there is no impact on timing. However, for greatest flexibility during debug (such as when creating designs with narrower interfaces), the recommended placement is in the address and command I/O bank.

Core Clock Network

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

Observe the following guidelines for sharing the core clock network:

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

I/O Bank

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

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

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

Hard Nios Processor

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

10.4. RLDRAM 3 Board Design Guidelines

The following topics provide layout guidelines for you to improve your system's signal integrity and to successfully implement an RLDRAM 3 interface.

The following topics focus on the following key factors that affect signal integrity:

- I/O standards
- RLDRAM 3 configurations
- Signal terminations
- Printed circuit board (PCB) layout guidelines

I/O Standards

RLDRAM 3 interface signals use the following JEDEC I/O signaling standards: HSTL 1.2 V and SSTL-12.

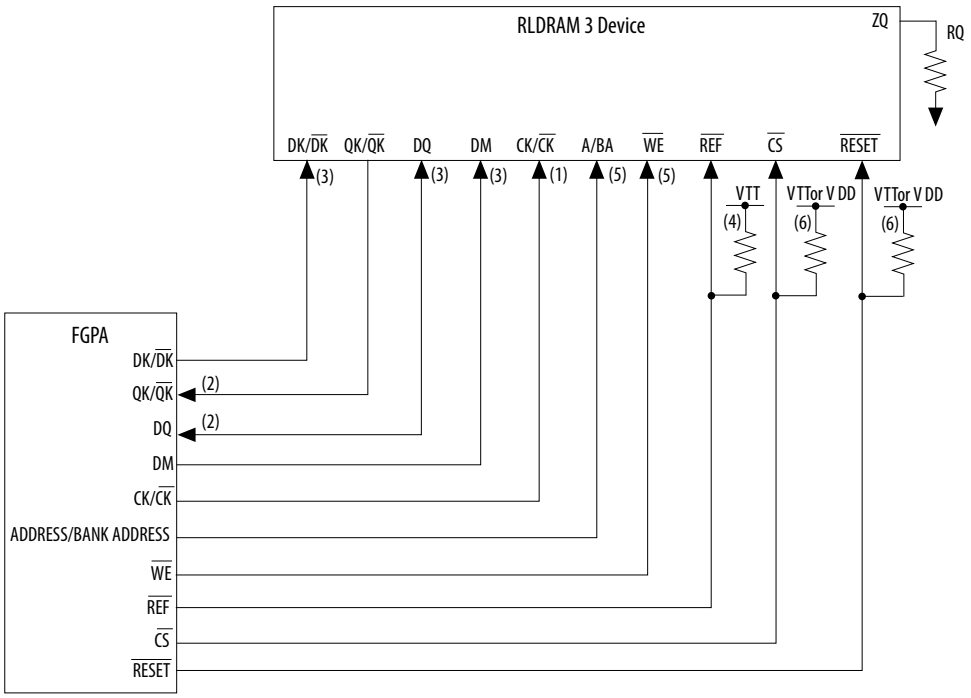
The RLDRAM 3 IP defaults to HSTL 1.2 V Class I outputs and HSTL 1.2 V inputs.

10.4.1. RLDRAM 3 Configurations

The Intel Stratix 10 EMIF IP for RLDRAM 3 supports interfaces for CIO RLDRAM 3 with one or two devices. With two devices, the interface supports a width expansion configuration up to 72-bits. The termination and layout principles for SIO RLDRAM 3 interfaces are similar to CIO RLDRAM 3, except that SIO RLDRAM 3 interfaces have unidirectional data buses.

The following figure shows the main signal connections between the FPGA and a single CIO RLDRAM 3 component.

Figure 86. Configuration with a Single CIO RLDRAM 3 Component

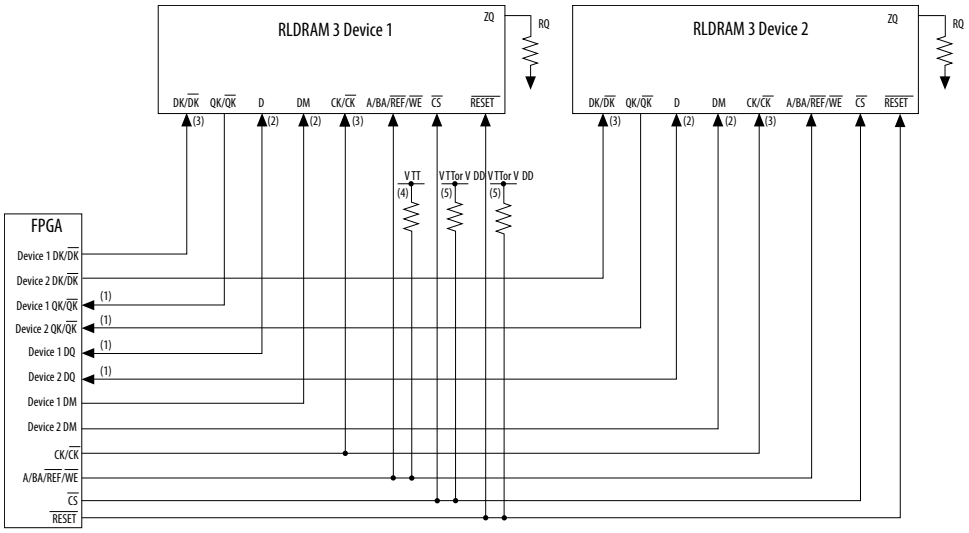


Notes to Figure:

1. Use external differential termination on CK/CK#.
2. Use FPGA parallel on-chip termination (OCT) for terminating QK/QK# and DQ on reads.
3. Use RLDRAM 3 component on-die termination (ODT) for terminating DQ, DM, and DK, DK# on writes.
4. Use external discrete termination with fly-by placement to avoid stubs.
5. Use external discrete termination for this signal, as shown for REF.
6. Use external discrete termination, as shown for REF, but you may require a pull-up resistor to VDD as an alternative option. Refer to the RLDRAM 3 device data sheet for more information about RLDRAM 3 power-up sequencing.

The following figure shows the main signal connections between the FPGA and two CIO RLDRAM 3 components in a width expansion configuration.

Figure 87. Configuration with Two CIO RLDRAM 3 Components in a Width Expansion Configuration



Notes to Figure:

1. Use FPGA parallel OCT for terminating QK/QK# and DQ on reads.
2. Use RLDRAM 3 component ODT for terminating DQ, DM, and DK on writes.
3. Use external dual 200 Ω differential termination.
4. Use external discrete termination at the trace split of the balanced T or Y topology.
5. Use external discrete termination at the trace split of the balanced T or Y topology, but you may require a pull-up resistor to VDD as an alternative option. Refer to the RLDRAM 3 device data sheet for more information about RLDRAM 3 power-up sequencing.

10.4.2. General Layout Guidelines

The following table lists general board design layout guidelines. These guidelines are Intel recommendations, and should not be considered as hard requirements. You should perform signal integrity simulation on all the traces to verify the signal integrity of the interface. You should extract the propagation delay information, enter it into the IP and compile the design to ensure that timing requirements are met.

Table 345. General Layout Guidelines

Parameter	Guidelines
Impedance	• All unused via pads must be removed, because they cause unwanted capacitance. • Trace impedance plays an important role in the signal integrity. You must perform board level simulation to determine the best characteristic impedance for your PCB. For example, it is possible that for multi rank systems 40 ohms could yield better results than a traditional 50 ohm characteristic impedance.
Decoupling Parameter	• Use 0.1 uF in 0402 size to minimize inductance • Make VTT voltage decoupling close to termination resistors • Connect decoupling caps between VTT and ground • Use a 0.1 uF cap for every other VTT pin and 0.01 uF cap for every VDD and VDDQ pin • Verify the capacitive decoupling using the Intel Power Distribution Network Design Tool
Power	• Route GND and V_{CC} as planes • Route VCCIO for memories in a single split plane with at least a 20-mil (0.020 inches, or 0.508 mm) gap of separation • Route VTT as islands or 250-mil (6.35-mm) power traces • Route oscillators and PLL power as islands or 100-mil (2.54-mm) power traces
General Routing	All specified delay matching requirements include PCB trace delays, different layer propagation velocity variance, and crosstalk. To minimize PCB layer propagation variance, Intel recommends that signals from the same net group always be routed on the same layer. • Use 45° angles (*not* 90° corners) • Avoid T-Junctions for critical nets or clocks • Avoid T-junctions greater than 250 mils (6.35 mm) • Disallow signals across split planes • Restrict routing other signals close to system reset signals • Avoid routing memory signals closer than 0.025 inch (0.635 mm) to PCI or system clocks

Related Information

Power Distribution Network

10.4.3. RLDRAM 3 Layout Guidelines

The following table lists the RLDRAM 3 general routing layout guidelines. These guidelines apply to Intel Stratix 10 devices.

Table 346. RLDRAM 3 Layout Guidelines

Parameter	Guidelines
General Routing	• If you must route signals of the same net group on different layers with the same impedance characteristic, simulate your worst case PCB trace tolerances to ascertain actual propagation delay differences. Typical layer to layer trace delay variations are of 15 ps/inch order. • Avoid T-junctions greater than 150 ps. • Match all signals within a given DQ group with a maximum skew of ±10 ps and route on the same layer.
Clock Routing	• Route clocks on inner layers with outer-layer run lengths held to under 150 ps. • These signals should maintain a 10-mil (0.254 mm) spacing from other nets. • Clocks should maintain a length-matching between clock pairs of ±5 ps. • Differential clocks should maintain a length-matching between P and N signals of ±2 ps. • Space between different clock pairs should be at least three times the space between the traces of a differential pair.
Address and Command Routing	• To minimize crosstalk, route address, bank address, and command signals on a different layer than the data and data mask signals. • Do not route the differential clock signals close to the address signals. • Keep the distance from the pin on the RLDRAM 3 component to the stub termination resistor (V_{TT}) to less than 50 ps for the address/command signal group. • Keep the distance from the pin on the RLDRAM 3 component to the fly-by termination resistor (V_{TT}) to less than 100 ps for the address/command signal group.
External Memory Routing Rules	• Apply the following parallelism rules for the RLDRAM 3 data/address/command groups: — 4 mils for parallel runs < 0.1 inch (approximately 1× spacing relative to plane distance). — 5 mils for parallel runs < 0.5 inch (approximately 1× spacing relative to plane distance). — 10 mils for parallel runs between 0.5 and 1.0 inches (approximately 2× spacing relative to plane distance). — 15 mils for parallel runs between 1.0 and 3.3 inch (approximately 3× spacing relative to plane distance).
Maximum Trace Length	• Keep the maximum trace length of all signals from the FPGA to the RLDRAM 3 components to 600 ps.
Trace Matching Guidance	The following layout approach is recommended, based on the preceding guidelines: 1. If the RLDRAM 3 interface has multiple DQ groups (×18 or ×36 RLDRAM 3 component or width expansion configuration), match all the DK/DK# and QK , QK # clocks as tightly as possible to optimize the timing margins in your design. 2. Route the DK/DK# write clock and QK/QK# read clock associated with a DQ group on the same PCB layer. Match these clock pairs to within ±5 ps. 3. Set the DK/DK# or QK/QK# clock as the target trace propagation delay for the associated data and data mask signals. 4. Route the data and data mask signals for the DQ group ideally on the same layer as the associated QK/QK# and DK/DK# clocks to within ±10 ps skew of the target clock. 5. Route the CK/CK# clocks and set as the target trace propagation delays for the address/command signal group. Match the CK/CK# clock to within ±50 ps of all the DK/DK# clocks. 6. Route the address/control signal group (address, bank address, CS, WE, and REF) ideally on the same layer as the CK/CK# clocks, to within ±20 ps skew of the CK/CK# traces.

continued...

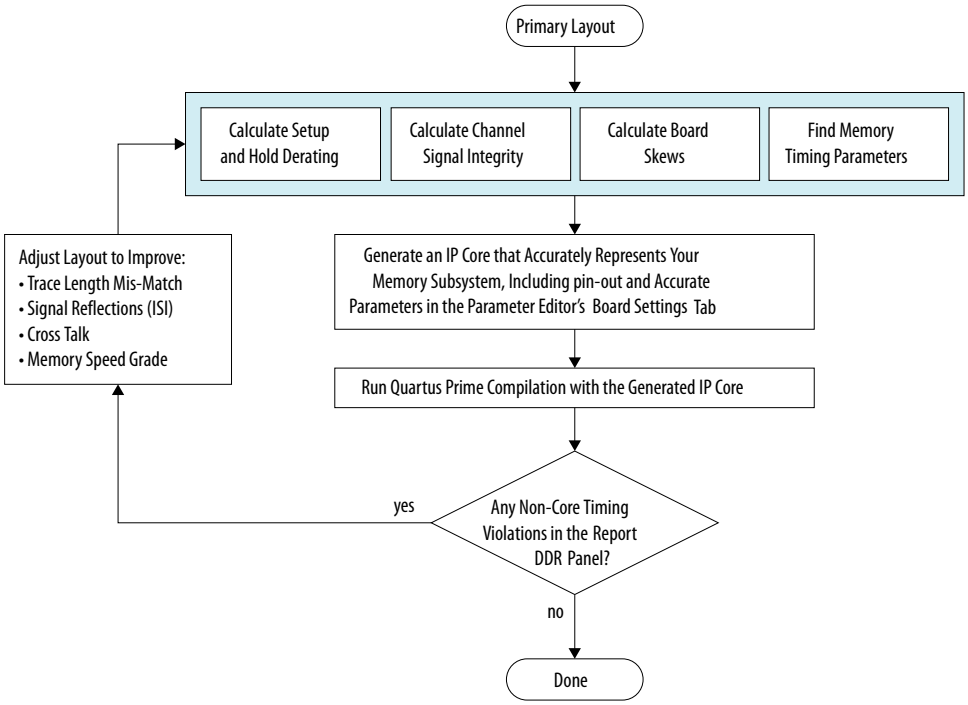
Parameter	Guidelines
	Note: It is important to match the delays of CK vs. DK, and CK vs. Addr-Cmd as much as possible. This layout approach provides a good starting point for a design requirement of the highest clock frequency supported for the RLDRAM 3 interface.

10.4.4. Layout Approach

For all practical purposes, you can regard the Timing Analyzer report on your memory interface as definitive for a given set of memory and board timing parameters.

You can find timing information under **Report DDR** in the Timing Analyzer and on the **Timing Analysis** tab in the parameter editor.

The following flowchart illustrates the recommended process to follow during the board design phase, to determine timing margin and make iterative improvements to your design.



Board Skew

For information on calculating board skew parameters, refer to *Board Skew Equations*, in this chapter.

The Board Skew Parameter Tool is an interactive tool that can help you calculate board skew parameters if you know the absolute delay values for all the memory related traces.

Memory Timing Parameters

For information on the memory timing parameters to be entered into the parameter editor, refer to the datasheet for your external memory device.

Related Information

Board Skew Parameter Tool

10.4.5. Package Deskew

You should follow Intel's package deskew guidance.

Related Information

Package Deskew

10.4.6. Package Migration

Package delays can be different for the same pin in different packages. If you want to use multiple migratable packages in your system, you should compensate for package skew as described in this topic.The information in this topic applies to Intel Stratix 10 devices.

Scenario 1

Your PCB is designed for multiple migratable devices, but you have only one device with which to go to production.

Assume two migratable packages, device A and device B, and that you want to go to production with device A. Follow these steps:

1. Perform package deskew for device A.
2. Compile your design for device A, with the **Package Skew** option enabled.
3. Note the skews in the `<core_name>.pin` file for device A. Deskew these package skews with board trace lengths as described in the preceding examples.
4. Recompile your design for device A.
5. For device B, open the parameter editor and deselect the **Package Deskew** option.
6. Calculate board skew parameters, only taking into account the board traces for device B, and enter that value into the parameter editor for device B.
7. Regenerate the IP and recompile the design for device B.
8. Verify that timing requirements are met for both device A and device B.

Scenario 2

Your PCB is designed for multiple migratable devices, and you want to go to production with all of them.

Assume you have device A and device B, and plan to use both devices in production. Follow these steps:

1. Do not perform any package deskew compensation for either device.
2. Compile a Quartus Prime design for device A with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
3. Verify that the **Report DDR** timing report meets your timing requirements.
4. Compile a Quartus Prime design for device B with the **Package Deskew** option disabled, and ensure that all board skews are entered accurately.
5. Verify that the **Report DDR** timing report meets your timing requirements.

10.4.7. Slew Rates

For optimum timing margins and best signal integrity for the address, command, and memory clock signals, you should generally use fast slew rates and external terminations.

In board simulation, fast slew rates may show a perceived signal integrity problem, such as reflections or a nonmonotonic waveform in the SSTL I/O switching region. Such indications may cause you to consider using slow slew rate options for either the address and command signals or the memory clock, or both.

If you set the **FPGA I/O tab parameter options ➤ Address/Command ➤ Slew Rate** and **Memory Clock ➤ Slew Rate** parameters to different values, a warning message appears: .

```
Warning: .emif_0: When the address/command signals and the memory clock signals
do not use the same slew rate setting, signals using the "Slow" setting are
delayed relative to signals using "Fast" setting. For accurate timing analysis,
you must perform I/O simulation and manually include the delay as board skew.
To avoid the issue, use the same slew rate setting for both address/command
signals and memory clock signals whenever possible.
```

Note: The warning message applies only to board-level simulation, and does not require any delay adjustments in the PCB design or Board tab parameter settings.

Due to limitations of the IBIS model correlation tolerance and the accuracy of the board simulation model, it is possible for signal integrity problems to appear when using fast slew rate during simulation but not occur during operation on hardware. If you observe a signal integrity problem during simulation with a fast slew rate, use an oscilloscope to view the signal at that point in hardware, to verify whether the problem exists on hardware, or only in simulation.

If the signal integrity problem exists on hardware as well as in simulation, using different slew rates for the address and command signals and the clock remains a valid approach, and the address and command calibration stage will help to improve the address and command to clock setup and hold time margins.

11. Intel Stratix 10 EMIF IP Timing Closure

This chapter describes timing analysis and optimization techniques that you can use to achieve timing closure.

11.1. Timing Closure

The following sections describe the timing analysis using the respective FPGA data sheet specifications and the user-specified memory data sheet parameters.

- Core to core (C2C) transfers have timing constraint created and are analyzed by the Timing Analyzer. Core timing does not include user logic timing within core or to and from EMIF block. The EMIF IP provides the constrained clock to the customer logic.
- Core to periphery (C2P) transfers have timing constraint created and are timing analyzed by the Timing Analyzer. Because of the increased number of C2P/P2C signals in 20nm families compared to previous families, more work is expected to ensure that these special timing arcs are properly modeled, both during timing analysis and compilation.
- Periphery to core (P2C) transfers have timing constraint created and are timing analyzed by the Timing Analyzer. Because of the increased number of C2P/P2C signals in 20nm families compared to previous families, more work is expected to ensure that these special timing arcs are properly modeled, both during timing analysis and compilation.
- Periphery to periphery (P2P) transfers are modeled entirely by a minimum pulse width violation on the hard block, and have no internal timing arc. P2P transfers are modeled only by a minimum pulse width violation on hardened block.

To account for the effects of calibration, the EMIF IP includes additional scripts that are part of the `<phy_variation_name>_report_timing.tcl` and `<phy_variation_name>_ report_timing_core.tcl` files that determine the timing margin after calibration. These scripts use the setup and hold slacks of individual pins to emulate what is occurring during calibration to obtain timing margins that are representative of calibrated PHYs. The effects considered as part of the calibrated timing analysis include improvements in margin because of calibration, and quantization error and calibration uncertainty because of voltage and temperature changes after calibration.

Related Information

Timing Analysis on page 349

11.1.1. Timing Analysis

Timing analysis of Intel Stratix 10 EMIF IP is somewhat simpler than that of earlier device families, because Intel Stratix 10 devices have more hardened blocks and there are fewer soft logic registers to be analyzed, because most are user logic registers.

Your Intel Stratix 10 EMIF IP includes a Synopsys Design Constraints File (`.sdc`) which contains timing constraints specific to your IP. The `.sdc` file also contains Tool Command Language (`.tcl`) scripts which perform various timing analyses specific to memory interfaces.

Two timing analysis flows are available for Intel Stratix 10 EMIF IP:

- Early I/O Timing Analysis, which is a precompilation flow.
- Full Timing Analysis, which is a post-compilation flow.

Related Information

Timing Closure on page 348

11.1.1.1. PHY or Core

Timing analysis of the PHY or core path includes the path from the last set of registers in the core to the first set of registers in the periphery (C2P), path from the last set of registers in the periphery to the first set of registers in the core (P2C) and ECC related path if it is enabled.

Core timing analysis excludes user logic timing to or from EMIF blocks. The EMIF IP provides a constrained clock (for example: ddr3_usr_clk) with which to clock customer logic; pll_afi_clk serves this purpose.

The PHY or core analyzes this path by calling the `report_timing` command in `<variation_name>_report_timing.tcl` and `<variation_name>_report_timing_core.tcl`.

Note: In version 14.1 and later, the *Spatial Pessimism Removal* slack values in the **Core to Periphery** and **Periphery to Core** tables are always equal to zero. This occurs because pessimism removal is integrated into the base timing analysis.

11.1.1.2. I/O Timing

I/O timing analysis includes analysis of read capture, write, address and command, DQS gating, and write leveling.

The Timing Analyzer provides a breakdown of the timing budgets which details margin loss due to transmitter, receiver, and channel. The Timing Analyzer displays the total margin in the last row of the timing report.

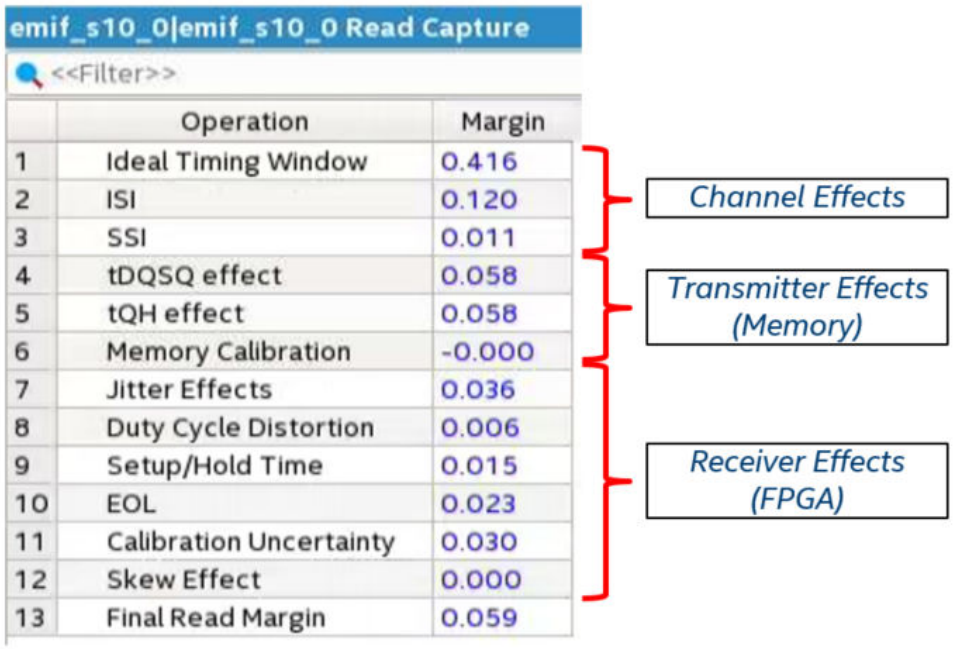
The I/O timing analysis described in the following topics is based on a 2 speed-grade device, interfacing with a DDR3 SDRAM UDIMM at 1066 MHz. A 1066 MHz DDR3 SDRAM UDIMM is used for the analysis.

11.1.1.2.1. Read Capture

Read capture timing analysis indicates the amount of slack on the DQ signals that are latched by the FPGA using the DQS strobe output of the memory device.

The Timing Analyzer analyzes read capture timing paths through conventional static timing analysis and further processing steps that account for memory calibration (which may include pessimism removal) and calibration uncertainties as shown in the following figure.

Figure 88. Read Capture Timing Analysis

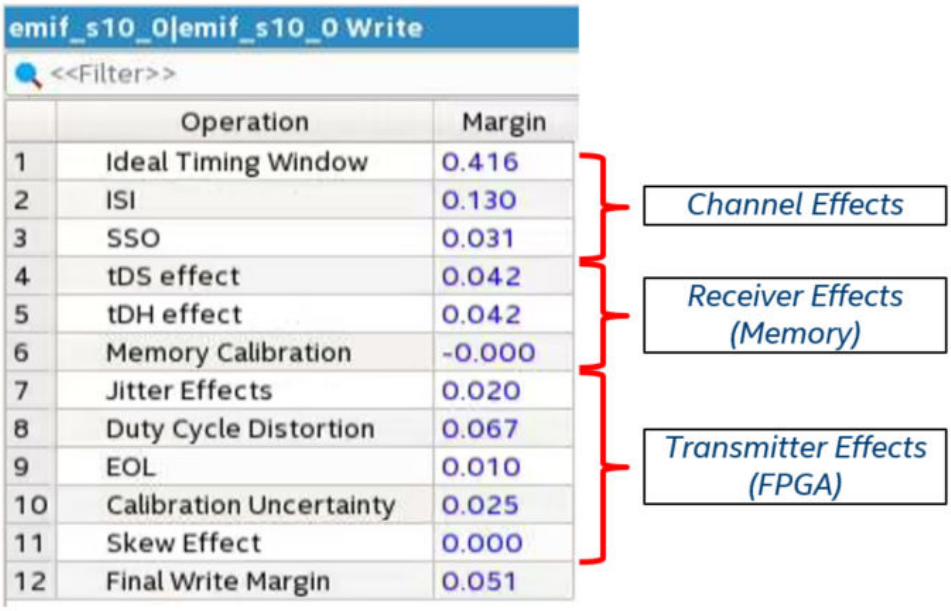


11.1.1.2.2. Write

Write timing analysis indicates the amount of slack on the DQ signals that are latched by the memory device using the DQS strobe output from the FPGA device.

As with read capture, the Timing Analyzer analyzes write timing paths through conventional static timing analysis and further processing steps that account for memory calibration (which may include pessimism removal) and calibration uncertainties as shown in the following figure.

Figure 89. Write Timing Analysis

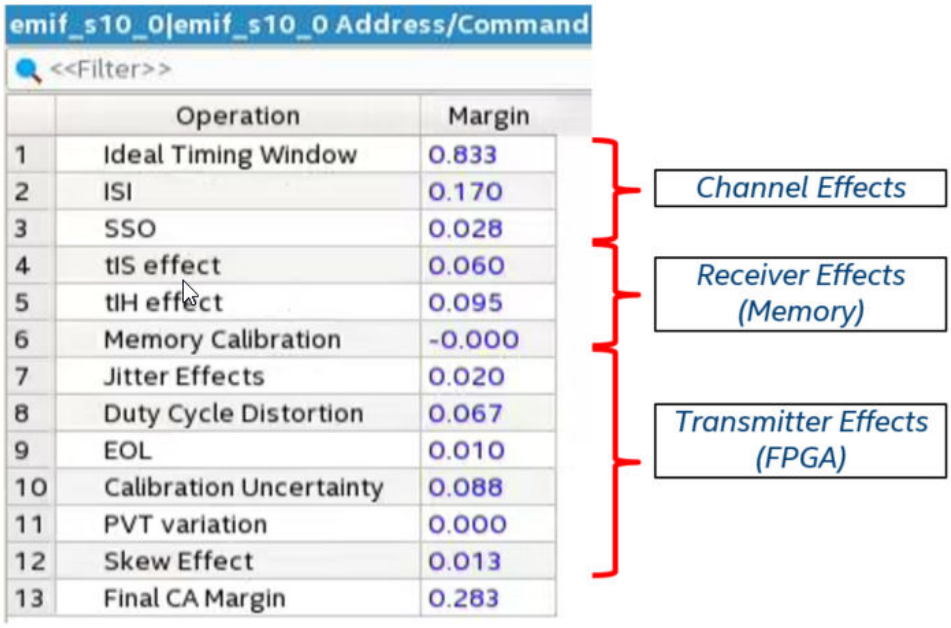


11.1.1.2.3. Address and Command

Address and command signals are single data rate signals latched by the memory device using the FPGA output clock; some are half-rate data signals, while others, such as the chip select, are full-rate signals.

The Timing Analyzer analyzes the address and command timing paths through conventional static timing analysis and further processing steps that account for memory pessimism removal (as shown in the following figure). Depending on the memory protocol in use, if address command calibration is performed, calibration uncertainty is subtracted from the timing window while PVT variation and skew effects are not subtracted, and vice versa.

Figure 90. Address and Command Timing Analysis

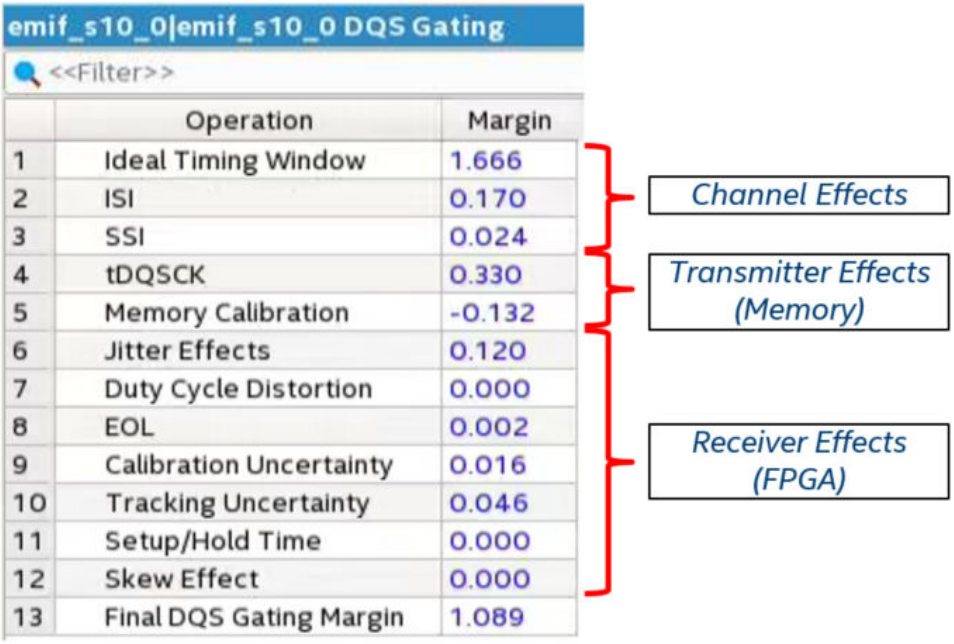


11.1.1.2.4. DQS Gating / Postamble

Postamble timing is a setup period during which the DQS signal goes low after all the DQ data has been received from the memory device during a read operation. After postamble time, the DQS signal returns from a low-impedance to a high-impedance state to disable DQS and disallow any glitches from writing false data over valid data.

The Timing Analyzer analyzes the postamble timing path in DDRx memory protocols only through an equation which considers memory calibration, calibration uncertainty, and tracking uncertainties as shown in the following figure.

Figure 91. DQS Gating Timing Analysis



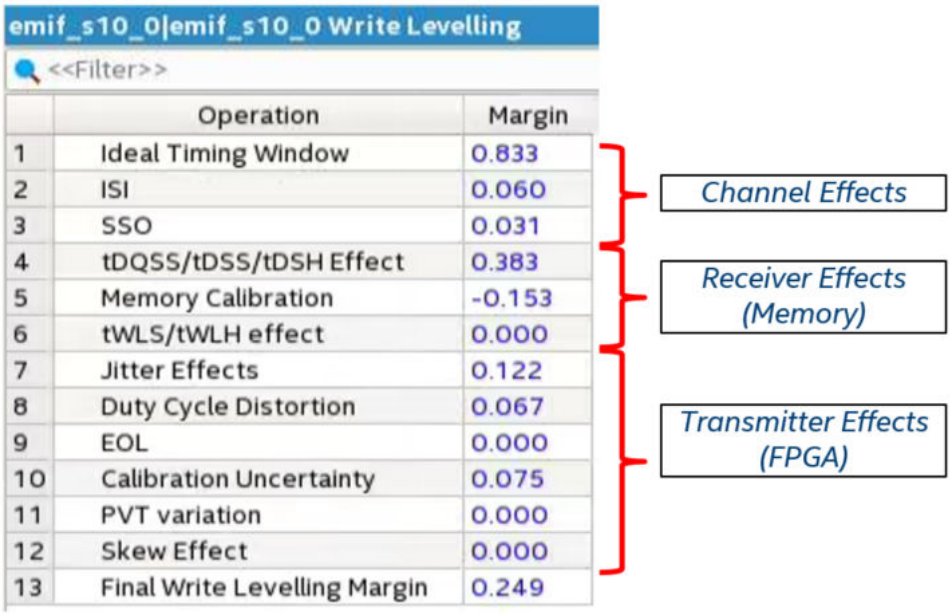
	Operation	Margin
1	Ideal Timing Window	1.666
2	ISI	0.170
3	SSI	0.024
4	tDQSCK	0.330
5	Memory Calibration	-0.132
6	Jitter Effects	0.120
7	Duty Cycle Distortion	0.000
8	EOL	0.002
9	Calibration Uncertainty	0.016
10	Tracking Uncertainty	0.046
11	Setup/Hold Time	0.000
12	Skew Effect	0.000
13	Final DQS Gating Margin	1.089

11.1.1.2.5. Write Leveling

In DDR3 SDRAM and DDR4 SDRAM interfaces, write leveling details the margin for the DQS strobe with respect to CK/CK# at the memory side.

The Timing Analyzer analyzes the write leveling timing path through an equation which considers memory calibration, calibration uncertainty and PVT variation as shown in the following figure.

Figure 92. Write Leveling Timing Analysis



	Operation	Margin
1	Ideal Timing Window	0.833
2	ISI	0.060
3	SSO	0.031
4	tDQSS/tDSS/tDSH Effect	0.383
5	Memory Calibration	-0.153
6	tWLS/tWLH effect	0.000
7	Jitter Effects	0.122
8	Duty Cycle Distortion	0.067
9	EOL	0.000
10	Calibration Uncertainty	0.075
11	PVT variation	0.000
12	Skew Effect	0.000
13	Final Write Levelling Margin	0.249

11.2. Timing Report DDR

The **Report DDR** task in the Timing Analyzer generates custom timing margin reports for all EMIF IP instances in your design. The Timing Analyzer generates this custom report by sourcing the wizard-generated `<variation_name>_report_timing.tcl` script.

This `<variation_name>_report_timing.tcl` script reports the following timing slacks on specific paths of the DDR SDRAM:

- Read capture
- Read resynchronization
- Mimic, address and command
- Core
- Core reset and removal
- Half-rate address and command
- DQS versus CK
- Write
- Write leveling (t_{DQSS})
- Write leveling (t_{DSS}/t_{DSH})
- DQS Gating (Postamble)

The `<variation_name>_report_timing.tcl` script checks basic design rules and assumptions; if violations are found, you receive critical warnings when the Timing Analyzer runs during compilation or when you run the **Report DDR** task.

To generate a timing margin report, follow these steps:

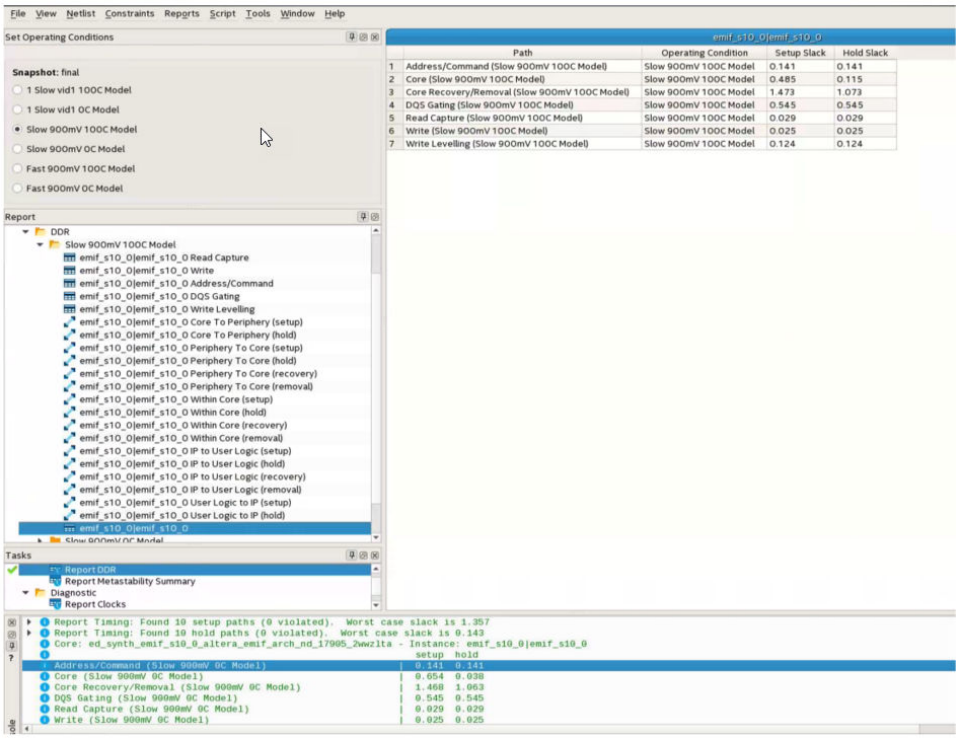
1. Compile your design in the Intel Quartus Prime software.
2. Launch the Timing Analyzer.
3. Double-click **Report DDR** from the **Tasks** pane. This action automatically executes the **Create Timing Netlist**, **Read SDC File**, and **Update Timing Netlist** tasks for your project.

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

The **Report DDR** feature creates a new DDR folder in the Timing Analyzer **Report** pane.

Expanding the DDR folder reveals the detailed timing information for each PHY timing path, in addition to an overall timing margin summary for the instance, as shown in the following figure.

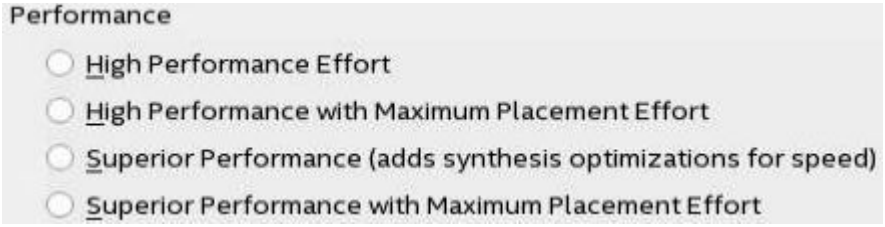
Figure 93. Timing Margin Summary Window Generated by Report DDR Task



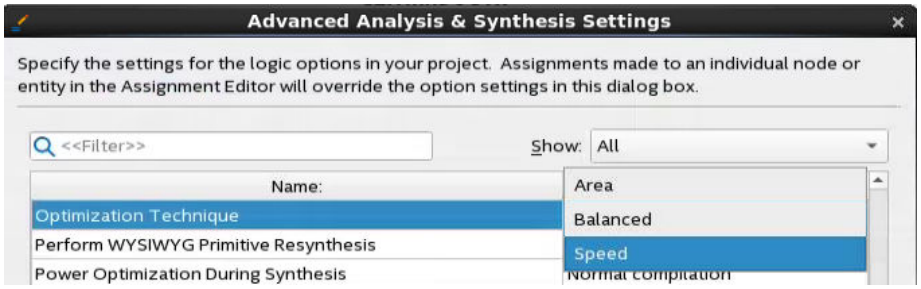
11.3. Optimizing Timing

The Intel Quartus Prime software offers several advanced features that you can use to assist in meeting core timing requirements.

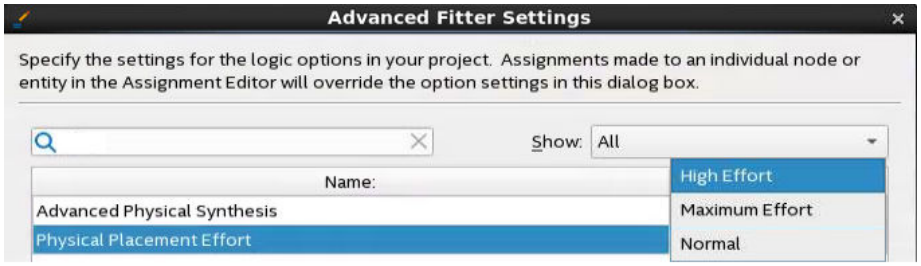
1. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings**. Under **Optimization mode**, select one of the **Performance** options.



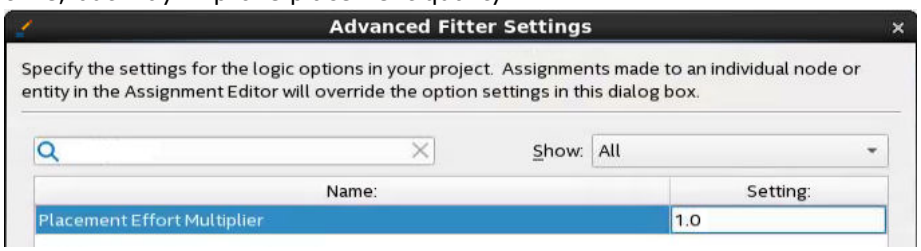
2. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings ➤ Advanced Settings (Synthesis).** For **Optimization Technique**, select **Speed**.



3. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings ➤ Advanced Settings (Fitter).** For **Physical Placement Effort**, select **High Effort** or **Maximum Effort**. The High and Maximum effort settings take additional compilation time to further optimize placement.



4. On the **Assignments** menu, click **Settings**. In the **Category** list, click **Compiler Settings ➤ Advanced Settings (Fitter).** For **Placement Effort Multiplier**, select a number higher than the preset value of 1.0. A higher value increases CPU time, but may improve placement quality.



Related Information

Netlist Optimizations and Physical Synthesis

11.4. Early I/O Timing Estimation

Early I/O timing analysis allows you to run I/O timing analysis without first compiling your design. You can use early I/O timing analysis to quickly evaluate whether adequate timing margin exists on the I/O interface between the FPGA and external memory device.

Early I/O timing analysis performs the following analyses:

- Read analysis
- Write analysis
- Address and command analysis
- DQS gating analysis
- Write leveling analysis

Early I/O timing analysis takes into consideration the following factors:

- The timing parameters of the memory device
- The speed and topology of the memory interface
- The board timing and ISI characteristics
- The timing of the selected FPGA device

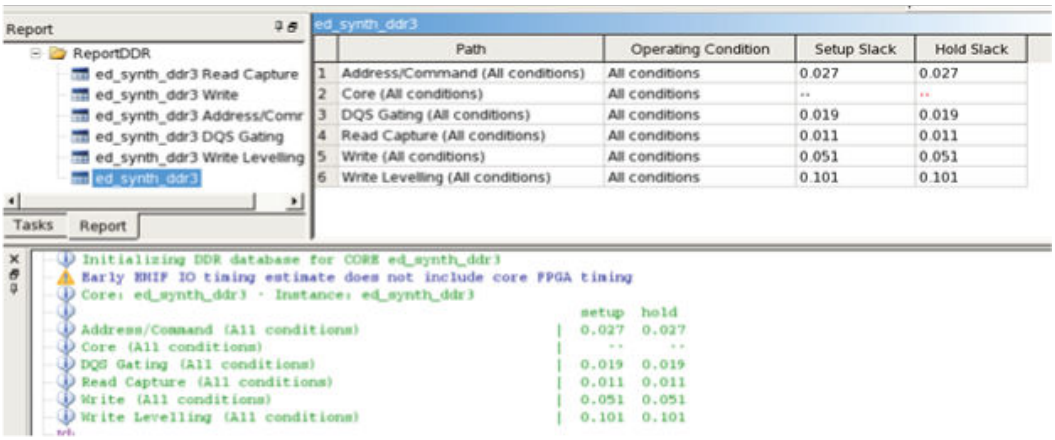
11.4.1. Performing Early I/O Timing Analysis

To perform early I/O timing analysis, follow these steps:

1. Instantiate an EMIF IP core.
   a. On the **Memory Timing** tab, enter accurate memory parameters.
   b. On the **Board Timing** tab, enter accurate values for Intersymbol Interference, and Board and Package Skews.
2. After generating your IP core, create a Intel Quartus Prime project and select your device from the **Available devices** list.
3. To launch the Timing Analyzer, select **Timing Analyzer** from the **Tools** menu.
4. To run early I/O timing analysis:
   a. Select **Run Tcl Script** from the **Script** menu.
   b. Run `\ip\ed_synth \ed_synth_emif_s10_0\altera_emif_arch_nd_<variation_name> \synth\<variation_name>_report_io_timing.tcl`.

The following figure shows an early I/O timing analysis from the Timing Analyzer using a DDR3 example design.

Figure 97. Report DDR Timing Results



Report DDR details the read capture, write, address and command, DQS gating, and write leveling timing analyses, which are identical to those obtained after a full design compilation. Core FPGA timing paths are not included in early I/O timing analysis.

12. Optimizing Controller Performance

When designing an external memory interface, you should understand the ways available to increase the efficiency and bandwidth of the memory controller.

The following topics discuss factors that affect controller efficiency and ways to increase the efficiency of the controller.

Controller Efficiency

Controller efficiency varies depending on data transaction. The best way to determine the efficiency of the controller is to simulate the memory controller for your specific design.

Controller efficiency is expressed as:

Efficiency = number of active cycles of data transfer/total number of cycles

The total number of cycles includes the number of cycles required to issue commands or other requests.

Note: You calculate the number of active cycles of data transfer in terms of local clock cycles. For example, if the number of active cycles of data transfer is 2 memory clock cycles, you convert that to the local clock cycle which is 1.

The following cases are based on a high-performance controller design targeting an FPGA device with a CAS latency of 3, and burst length of 4 on the memory side (2 cycles of data transfer), with accessed bank and row in the memory device already open. The FPGA has a command latency of 9 cycles in half-rate mode. The `local_ready` signal is high.

- Case 1: The controller performs individual reads.

  Efficiency = 1/(1 + CAS + command latency) = 1/(1+1.5+9) = 1/11.5 = 8.6%
- Case 2: The controller performs 4 back to back reads.

  In this case, the number of data transfer active cycles is 8. The CAS latency is only counted once because the data coming back after the first read is continuous. Only the CAS latency for the first read has an impact on efficiency. The command latency is also counted once because the back to back read commands use the same bank and row.

  Efficiency = 4/(4 + CAS + command latency) = 4/(4+1.5+9) = 1/14.5 = 27.5%

12.1. Interface Standard

Complying with certain interface standard specifications affects controller efficiency.

When interfacing the memory device to the memory controller, you must observe timing specifications and perform the following bank management operations:

- **Activate**

  Before you issue any read (RD) or write (WR) commands to a bank within an SDRAM device, you must open a row in that bank using the activate (ACT) command. After you open a row, you can issue a read or write command to that row based on the t_{RCD} specification. Reading or writing to a closed row has negative impact on the efficiency as the controller has to first activate that row and then wait until t_{RCD} time to perform a read or write.

- **Precharge**

  To open a different row in the same bank, you must issue a precharge command. The precharge command deactivates the open row in a particular bank or the open row in all banks. Switching a row has a negative impact on the efficiency as you must first precharge the open row, then activate the next row and wait t_{RCD} time to perform any read or write operation to the row.

- **Device CAS latency**

  The higher the CAS latency, the less efficient an individual access. The memory device has its own read latency, which is about 12 ns to 20 ns regardless of the actual frequency of the operation. The higher the operating frequency, the longer the CAS latency is in number of cycles.

- **Refresh**

  A refresh, in terms of cycles, consists of the precharge command and the waiting period for the auto refresh. Based on the memory data sheet, these components require the following values:

  — t_{RP} = 15 ns, 18 clock cycles for a 1200-MHz operation (0.833 ns period for 1200 MHz)

  — t_{RFC} = 260 ns, 313 clock cycles for a 1200-MHz operation.

  Based on this calculation, a refresh pauses read or write operations for 18 clock cycles. So, at 1200 MHz, you lose 3.53% (331 x 0.833 ns/7.8 us) of the total efficiency.

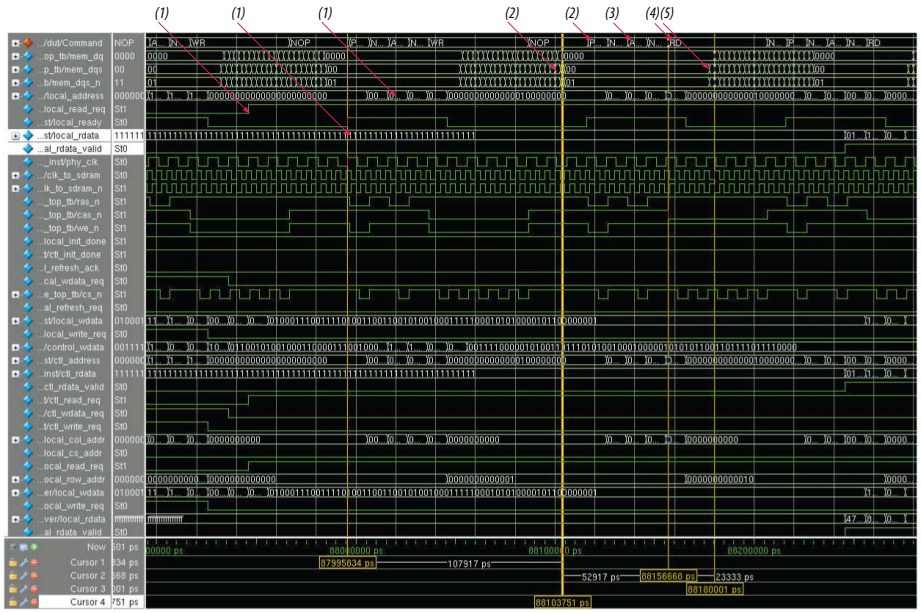
12.2. Bank Management Efficiency

The following figures show examples of how the bank management operations affect controller efficiency.

The first figure shows a read operation in which you have to change a row in a bank. This figure shows how CAS latency and precharge and activate commands affect efficiency.

The following figure illustrates a read-after-write operation. The controller changes the row address after the write-to-read from a different row.

Figure 98. Read Operation—Changing A Row in A Bank



The following sequence of events describes the above figure:

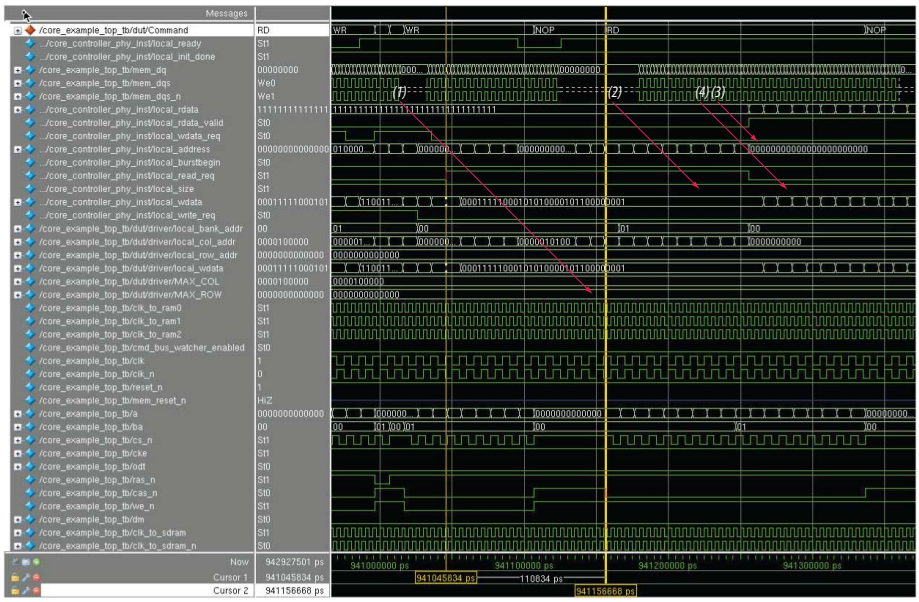
1. The `local_read_req` signal goes high, and when the `local_ready` signal goes high, the controller accepts the read request along with the address.
2. After the memory receives the last write data, the row changes for read. Now you require a precharge command to close the row opened for write. The controller waits for t_{WR} time (3 memory clock cycles) to give the precharge command after the memory receives the last write data.
3. After the controller issues the precharge command, it must wait for t_{RP} time to issue an activate command to open a row.
4. After the controller gives the activate command to activate the row, it needs to wait t_{RCD} time to issue a read command.
5. After the memory receives the read command, it takes the memory some time to provide the data on the pin. This time is known as CAS latency, which is 3 memory clock cycles in this case.

Note: The t_{WR}, t_{RP}, t_{RCD}, and CAS values depend on memory timing parameters.

For this particular case, you need approximately 17 local clock cycles to issue a read command to the memory. Because the row in the bank changes, the read operation takes a longer time, as the controller has to issue the precharge and activate commands first. You do not have to take into account t_{WTR} for this case because the precharge and activate operations already exceeded t_{WTR} time.

The following figure shows the case where you use the same the row and bank address when the controller switches from write to read. In this case, the read command latency is reduced.

Figure 99. Changing From Write to Read—Same Row and Bank Address



The following sequence of events describes the above figure:

1. The `local_read_req` signal goes high and the `local_ready` signal is high already. The controller accepts the read request along with the address.
2. When switching from write to read, the controller has to wait t_{WTR} time before it gives a read command to the memory.
3. The SDRAM device receives the read command.
4. After the SDRAM device receives the read command, it takes some time to give the data on the pin. This time is called CAS latency, which is 3 memory clock cycles in this case.

Note: The t_{WTR} and CAS values depend on memory timing parameters.

For the case illustrated in the second figure above, you need approximately 11 local clock cycles to issue a read command to the memory. Because the row in the bank remains the same, the controller does not have to issue the precharge and activate commands, which speeds up the read operation and in turn results in a better efficiency compared to the case in the first figure above.

Similarly, if you do not switch between read and write often, the efficiency of your controller improves significantly.

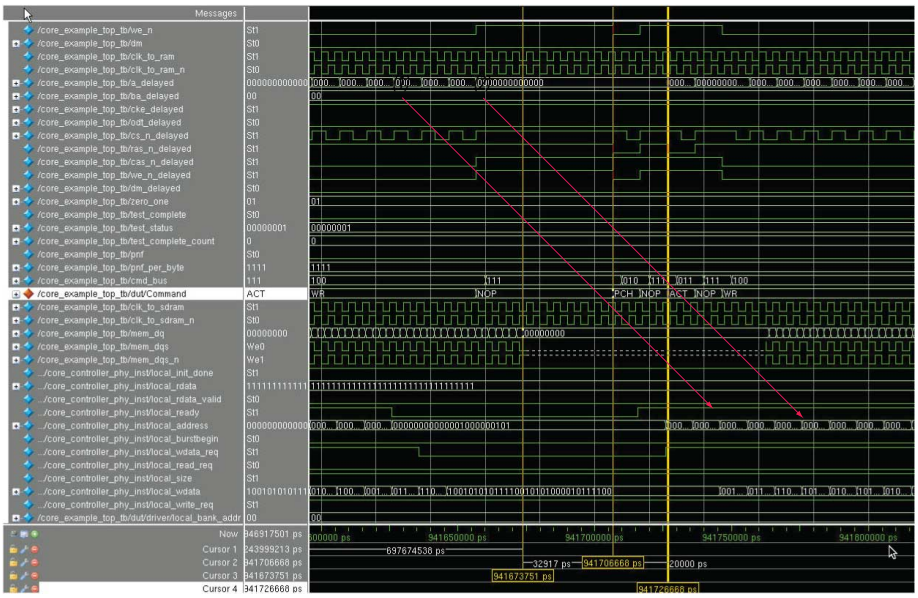
12.3. Data Transfer

The following methods of data transfer reduce the efficiency of your controller:

- Performing individual read or write accesses is less efficient.
- Switching between read and write operation has a negative impact on the efficiency of the controller.
- Performing read or write operations from different rows within a bank or in a different bank—if the bank and a row you are accessing is not already open—also affects the efficiency of your controller.

The following figure shows an example of changing the row in the same bank.

Figure 100. Changing Row in the Same Bank



The following sequence of events describes the above figure:

1. You have to wait t_{WR} time before giving the precharge command
2. You then wait t_{RP} time to give the activate command.

Note: The t_{WR} and t_{RP} values depend on memory timing parameters.

12.4. Improving Controller Efficiency

You can use the following tools and methods to improve the efficiency of your controller.

- Auto-Precharge Commands
- Additive Latency
- Calibration
- Bank Interleaving
- Additive Latency and Bank Interleaving
- User-Controlled Refresh
- Frequency of Operation

- Series of Reads or Writes
- Data Reordering
- Starvation Control
- Command Reordering
- Bandwidth
- Enable Command Priority Control

The following sections discuss these methods in detail.

12.4.1. Auto-Precharge Commands

The auto-precharge read and write commands allow you to indicate to the memory device that a given read or write command is the last access to the currently opened row.

The memory device automatically closes or auto-precharges the page that is currently being accessed, so that the next access to the same bank is faster. The Auto-Precharge command is useful when you want to perform fast random memory accesses.

The Timer Bank Pool (TBP) block supports the dynamic page policy, where depending on user input on local autoprecharge input would keep a page open or close. In a closed-page policy, a page is always closed after it is accessed with auto-precharge command. When the data pattern consists of repeated reads or writes to addresses not within the same page, the optimal system achieves the maximum efficiency allowed by continuous page miss limited access. Efficiency losses are limited to those associated with activating and refreshing. An efficiency of 10-20% should be expected for this closed-page policy.

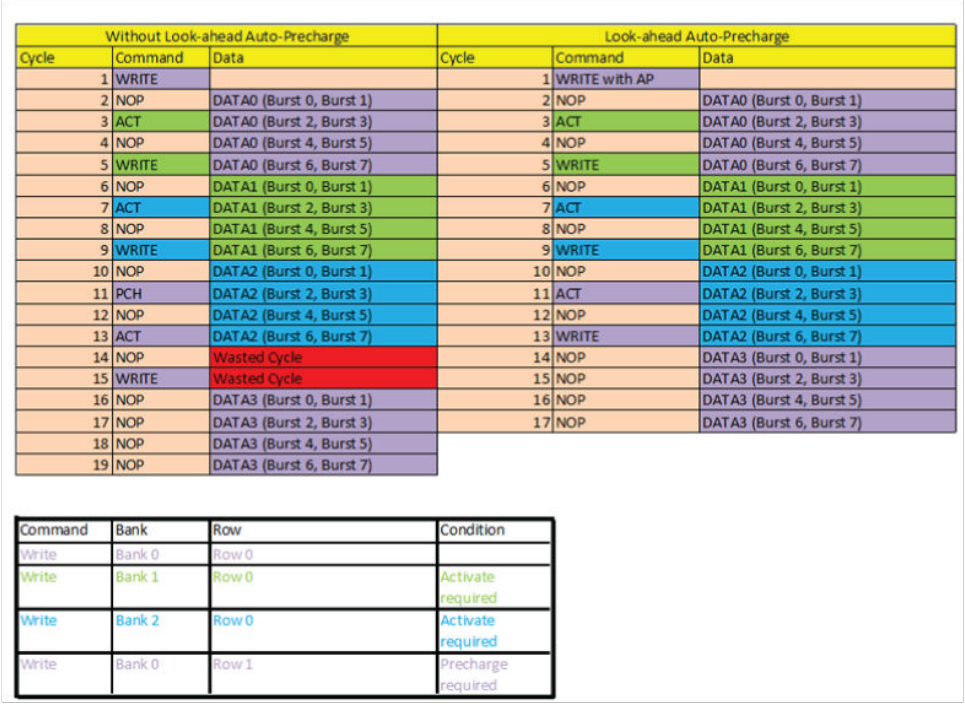
In an open-page policy, the page remains open after it is accessed for incoming commands. When the data pattern consists of repeated reads or writes to sequential addresses within the same page, the optimal system can achieve 100% efficiency for page-open transactions (ignoring the effects of periodic refreshes, which typically consume around 2-3% of total efficiency), with minimum latency for highest priority single transactions.

If you turn on **Enable Auto-Precharge Control**, you can instruct the controller to issue an autoprecharge read or write command. The next time you access that bank, the access is faster because the controller does not have to precharge the bank before activating the row that you want to access.

The controller-derived autoprecharge logic evaluates the pending commands in the command buffer and determines the most efficient autoprecharge operation to perform. The autoprecharge logic can reorder commands if necessary. When all TBP are occupied due to tracking an open page, TBP uses a scheme called on-demand flush, where it stops tracking a page to create space for an incoming command.

The following figure compares auto-precharge with and without look-ahead support.

Figure 101. Comparison With and Without Look-ahead Auto-Precharge



Without Look-ahead Auto-Precharge			Look-ahead Auto-Precharge		
Cycle	Command	Data	Cycle	Command	Data
1	WRITE		1	WRITE with AP	
2	NOP	DATA0 (Burst 0, Burst 1)	2	NOP	DATA0 (Burst 0, Burst 1)
3	ACT	DATA0 (Burst 2, Burst 3)	3	ACT	DATA0 (Burst 2, Burst 3)
4	NOP	DATA0 (Burst 4, Burst 5)	4	NOP	DATA0 (Burst 4, Burst 5)
5	WRITE	DATA0 (Burst 6, Burst 7)	5	WRITE	DATA0 (Burst 6, Burst 7)
6	NOP	DATA1 (Burst 0, Burst 1)	6	NOP	DATA1 (Burst 0, Burst 1)
7	ACT	DATA1 (Burst 2, Burst 3)	7	ACT	DATA1 (Burst 2, Burst 3)
8	NOP	DATA1 (Burst 4, Burst 5)	8	NOP	DATA1 (Burst 4, Burst 5)
9	WRITE	DATA1 (Burst 6, Burst 7)	9	WRITE	DATA1 (Burst 6, Burst 7)
10	NOP	DATA2 (Burst 0, Burst 1)	10	NOP	DATA2 (Burst 0, Burst 1)
11	PCH	DATA2 (Burst 2, Burst 3)	11	ACT	DATA2 (Burst 2, Burst 3)
12	NOP	DATA2 (Burst 4, Burst 5)	12	NOP	DATA2 (Burst 4, Burst 5)
13	ACT	DATA2 (Burst 6, Burst 7)	13	WRITE	DATA2 (Burst 6, Burst 7)
14	NOP	Wasted Cycle	14	NOP	DATA3 (Burst 0, Burst 1)
15	WRITE	Wasted Cycle	15	NOP	DATA3 (Burst 2, Burst 3)
16	NOP	DATA3 (Burst 0, Burst 1)	16	NOP	DATA3 (Burst 4, Burst 5)
17	NOP	DATA3 (Burst 2, Burst 3)	17	NOP	DATA3 (Burst 6, Burst 7)
18	NOP	DATA3 (Burst 4, Burst 5)			
19	NOP	DATA3 (Burst 6, Burst 7)			

Command	Bank	Row	Condition
Write	Bank 0	Row 0	
Write	Bank 1	Row 0	Activate required
Write	Bank 2	Row 0	Activate required
Write	Bank 0	Row 1	Precharge required

Without using the look-ahead auto-precharge feature, the controller must precharge to close and then open the row before the write or read burst for every row change. When using the look-ahead precharge feature, the controller decides whether to do auto-precharge read/write by evaluating the incoming command; subsequent reads or writes to same bank/different row require only an activate command.

As shown in the preceding figure, the controller performs an auto-precharge for the write command to bank 0 at cycle 1. The controller detects that the next write at cycle 13 is to a different row in bank 0, and hence saves 2 data cycles.

The following efficiency results apply to the above figure:

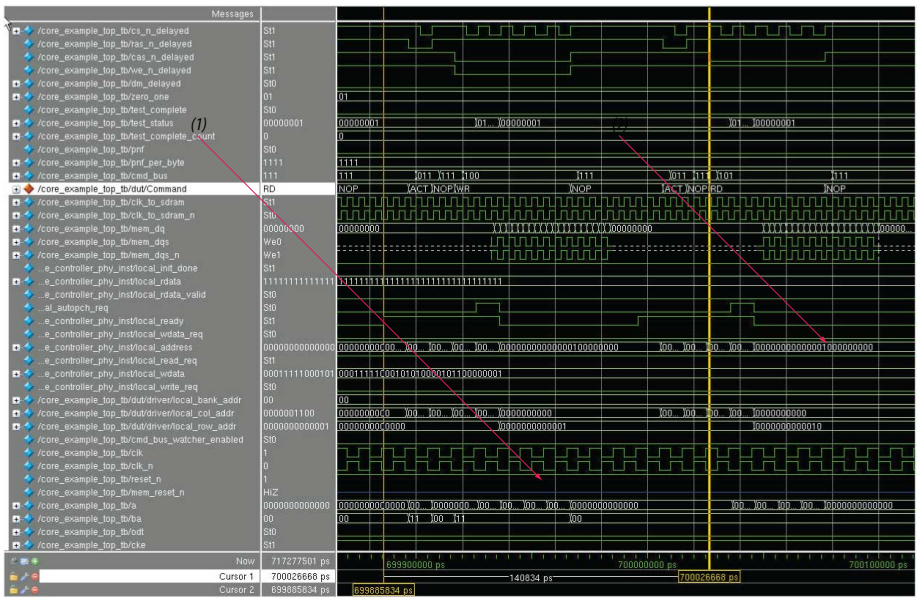
Table 347. Comparative Efficiencies With and Without Look-Ahead Auto-Precharge Feature

	Without Look-ahead Auto-precharge	With Look-ahead Auto-precharge
Active cycles of data transfer	16	16
Total number of cycles	19	17
Approximate efficiency	84%	94%

The look-ahead auto-precharge used increases efficiency by approximately 10%.

The following figure shows how you can improve controller efficiency using the auto-precharge command.

Figure 102. Improving Efficiency Using Auto-Precharge Command



The following sequence of events describes the above figure:

1. The controller accepts a read request from the local side as soon as the `local_ready` signal goes high.
2. The controller gives the activate command and then gives the read command. The read command latency is approximately 14 clock cycles for this case as compared to the similar case with no auto precharge which had approximately 17 clock cycles of latency (described in the "data Transfer" topic).

When using the auto-precharge option, note the following guidelines:

- Use the auto-precharge command if you know the controller is issuing the next read or write to a particular bank and a different row.
- Auto-precharge does not improve efficiency if you auto-precharge a row and immediately reopen it.

12.4.2. Latency

The following latency data applies to all memory protocols supported by the Intel Stratix 10 EMIF IP.

Table 348. Latency in Full-Rate Memory Clock Cycles

Rate [1]	Controller Address & Command	PHY Address & Command	Memory Read Latency [2]	PHY Read Data Return	Controller Read Data Return	Round Trip	Round Trip Without Memory
Half:Write	12	2	3-23	—	—	—	—
Half:Read	8	2	3-23	6	8	27-47	24
Quarter:Write	14	2	3-23	—	—	—	—

continued...

Rate [1]	Controller Address & Command	PHY Address & Command	Memory Read Latency [2]	PHY Read Data Return	Controller Read Data Return	Round Trip	Round Trip Without Memory
Quarter:Read	10	2	3-23	6	14	35-55	32
Half:Write (ECC)	14	2	3-23	—	—	—	—
Half:Read (ECC)	12	2	3-23	6	8	31-51	28
Quarter:Write (ECC)	14	2	3-23	—	—	—	—
Quarter:Read (ECC)	12	2	3-23	6	14	37-57	34

1. User interface rate; the controller always operates in half rate.
2. Minimum and maximum read latency range for DDR3 and DDR4.

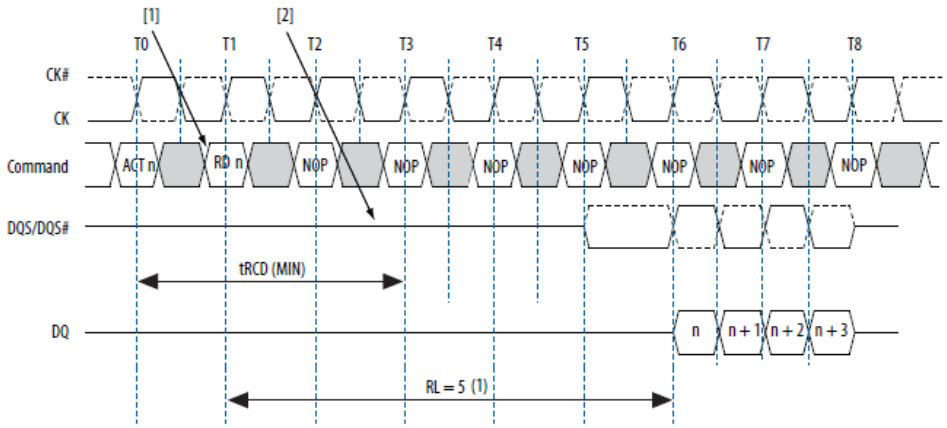
12.4.2.1. Additive Latency

Additive latency increases the efficiency of the command and data bus for sustainable bandwidths.

You may issue the commands externally but the device holds the commands internally for the duration of additive latency before executing, to improve the system scheduling. The delay helps to avoid collision on the command bus and gaps in data input or output bursts. Additive latency allows the controller to issue the row and column address commands—activate, and read or write—in consecutive clock cycles, so that the controller need not hold the column address for several (t_{RCD}) cycles. This gap between the activate and the read or write command can cause bubbles in the data stream.

The following figure shows an example of additive latency.

Figure 103. Additive Latency—Read



The following sequence of events describes the above figure:

1. The controller issues a read or write command before the t_{RCD} (MIN) requirement — additive latency less than or equal to t_{RCD} (MIN).
2. The controller holds the read or write command for the time defined by additive latency before issuing it internally to the SDRAM device.

Read latency = additive latency + CAS latency

Write latency = additive latency + CAS latency – t_{CK}

12.4.3. Calibration

The time needed for calibration varies, depending on many factors including the interface width, the number of ranks, frequency, board layout, and difficulty of calibration.

The following table lists approximate typical calibration times for various protocols and configurations.

Table 349. Intel Stratix 10 EMIF IP Approximate Calibration Times

Protocol	Rank and Frequency	Typical Calibration Time
DDR3, x64 UDIMM, DQS x8, DM on	1 rank, 933 MHz	102 ms
	1 rank, 800 MHz	106 ms
	2 rank, 933 MHz	198 ms
	2 rank, 800 MHz	206 ms
DDR4, x64 UDIMM, DQS x8, DBI on	1 rank, 1067 MHz	314 ms
	1 rank, 800 MHz	353 ms
	2 rank 1067 MHz	625 ms
	2 rank 800 MHz	727 ms
RLDRAM 3, x36	1200 MHz	2808 ms
	1067 MHz	2825 ms
	1200 MHz, with DM	2818 ms
	1067 MHz, with DM	2833 ms
QDR II, x36, BWS on	333 MHz	616 ms
	633 MHz	833 ms
QDR-IV, x36, BWS on	1067 MHz	1563 ms
	1067 MHz, with DBI	1556 ms

12.4.4. Bank Interleaving

You can use bank interleaving to sustain bus efficiency when the controller misses a page, and that page is in a different bank.

Note: Page size refers to the minimum number of column locations on any row that you access with a single activate command. For example: For a 512Mb x8 DDR3 SDRAM with 1024 column locations (column address A[9:0]), page size = 1024 columns x 8 = 8192 bits = 8192/8 bytes = 1024 bytes (1 KB)

Without interleaving, the controller sends the address to the SDRAM device, receives the data requested, and then waits for the SDRAM device to precharge and reactivate before initiating the next data transaction, thus wasting several clock cycles.

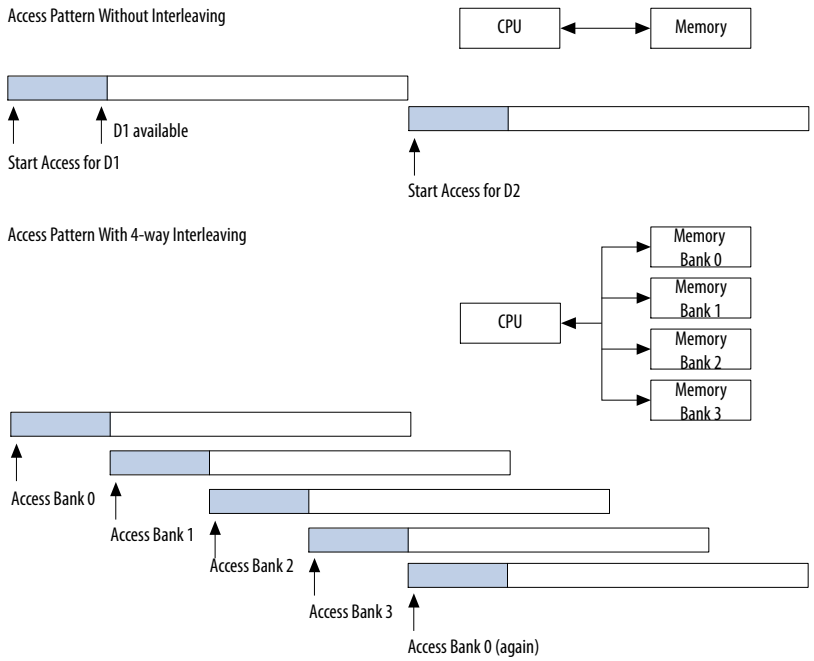
Interleaving allows banks of the SDRAM device to alternate their background operations and access cycles. One bank undergoes its precharge/activate cycle while another is being accessed. By alternating banks, the controller improves its performance by masking the precharge/activate time of each bank. If there are four banks in the system, the controller can ideally send one data request to each of the banks in consecutive clock cycles.

For example, in the first clock cycle, the CPU sends an address to Bank 0, and then sends the next address to Bank 1 in the second clock cycle, before sending the third and fourth addresses to Banks 2 and 3 in the third and fourth clock cycles respectively. The sequence is as follows:

1. Controller sends address 0 to Bank 0.
2. Controller sends address 1 to Bank 1 and receives data 0 from Bank 0.
3. Controller sends address 2 to Bank 2 and receives data 1 from Bank 1.
4. Controller sends address 3 to Bank 3 and receives data 2 from Bank 2.
5. Controller receives data 3 from Bank 3.

The following figure shows how you can use interleaving to increase bandwidth.

Figure 104. Using Interleaving to Increase Bandwidth



The controller supports three interleaving options:

Chip-Bank-Row-Col (or CS-BG-Bank-CID-Row-Col) – This is a noninterleaved option. Select this option to improve efficiency with random traffic

Chip-Row-Bank-Col (or CS-CID-Row-Col-Bank-BG) – This option uses bank interleaving without chip select interleaving. Select this option to improve efficiency with sequential traffic, by spreading smaller data structures across all banks in a chip.

Row-Chip-Bank-Col (or CID-Row-CS-Bank-Col-BG) – This option uses bank interleaving with chip select interleaving. Select this option to improve efficiency with sequential traffic and multiple chip selects. This option allows smaller data structures to spread across multiple banks and chips.

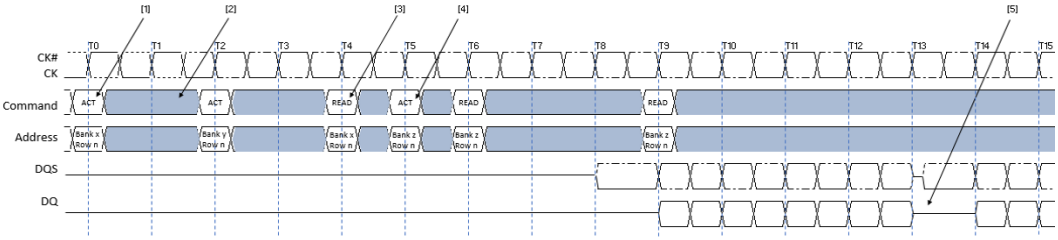
Bank interleaving is a fixed pattern of data transactions, enabling best-case bandwidth and latency, and allowing for sufficient interleaved transactions between opening banks to completely hide t_{RC}. Optimum efficiency is achieved for bank interleave transactions with 8 banks.

12.4.5. Additive Latency and Bank Interleaving

Using additive latency together with bank interleaving increases the bandwidth of the controller.

The following figure shows an example of bank interleaving in a read operation without additive latency. The example uses bank interleave reads with CAS latency of 5, and burst length of 8.

Figure 105. Bank Interleaving—Without Additive Latency



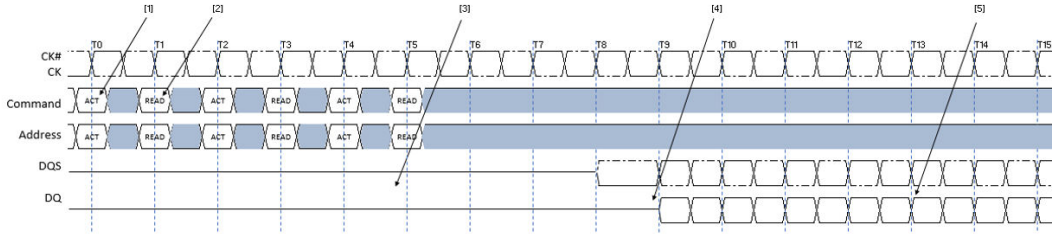
The following sequence of events describes the above figure:

1. The controller issues an activate command to open the bank, which activates bank *x* and the row in it.
2. After t_{RCD} time, the controller issues a read with auto-precharge command to the specified bank.
3. Bank *y* receives an activate command after t_{RRD} time.
4. The controller cannot issue an activate command to bank *z* at its optimal location because it must wait for bank *x* to receive the read with auto-precharge command, thus delaying the activate command for one clock cycle.
5. The delay in activate command causes a gap in the output data from the memory device.

Note: If you use additive latency of 1, the latency affects only read commands and not the timing for write commands.

The following figure shows an example of bank interleaving in a read operation with additive latency. The example uses bank interleave reads with additive latency of 3, CAS latency of 5, and burst length of 8. In this configuration, the controller issues back-to-back activate and read with auto-precharge commands.

Figure 106. Bank Interleaving—With Additive Latency



The following sequence of events describes the above figure:

1. The controller issues an activate command to bank *x*.
2. The controller issues a read with auto precharge command to bank *x* right after the activate command, before waiting for the t_{RCD} time.
3. The controller executes the read with auto-precharge command t_{RCD} time later on the rising edge T4.
4. 5 cycles of CAS latency later, the SDRAM device issues the data on the data bus.
5. For burst length of 8, you need 2 cycles for data transfer. With 2 clocks of giving activate and read with auto-precharge commands, you get a continuous flow of output data.

Compare the efficiency results in the two preceding figures:

- bank interleave reads with no additive latency, CAS latency of 5, and burst length of 8 (first figure),

  Number of active cycles of data transfer = 8.

  Total number of cycles = 18

  Efficiency = 44%
- bank interleave reads with additive latency of 3, CAS latency of 4, and burst length of 4 (second figure),

  Number of active cycles of data transfer = 8.

  Total number of cycles = 17

  Efficiency = approximately 47%

The interleaving reads used with additive latency increases efficiency by approximately 3%.

Note: Additive latency improves the efficiency of back-to-back interleaved reads or writes, but not individual random reads or writes.

12.4.6. User-Controlled Refresh

The requirement to periodically refresh memory contents is normally handled by the memory controller; however, the **User Controlled Refresh** option allows you to determine when memory refresh occurs.

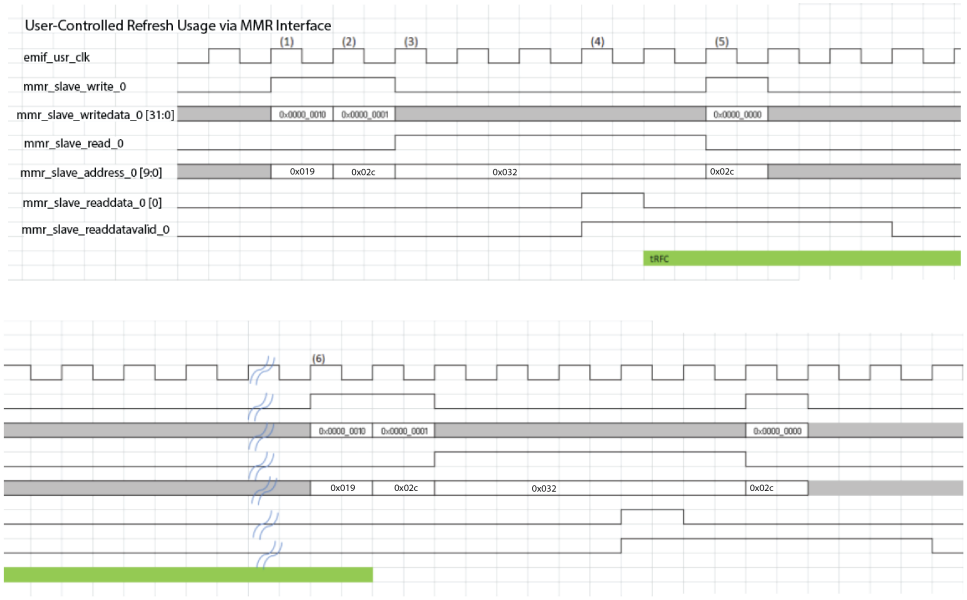
With specific knowledge of traffic patterns, you can time the refresh operations so that they do not interrupt read or write operations, thus improving efficiency.

Note: If you enable the auto-precharge control, you must ensure that the average periodic refresh requirement is met, because the controller does not issue any refreshes until you instruct it to.

12.4.6.1. Back-to-Back User-Controlled Refresh Usage

The following diagram illustrates the user-controlled refresh for the hard memory controller (HMC), using the MMR interface.

Figure 107. User-Controlled Refresh via MMR Interface



To perform a user-controlled refresh in the hard memory controller using the MMR interface, follow these steps:

1. Write to the **cfg_user_rfsh_en** register (address=0x019) with the data `0x0000_0010` to enable user refresh.
2. Write to the **mmr_refresh_req** register (address=0x02c) with the data `0x0000_0001` to send a refresh request to rank 0.

   Note:
   - Each bit corresponds to one specific rank; for example, data `0x0000_0002` corresponds to rank 1.
   - You may program refreshes to more than one rank at a time.
3. Read from the **mmr_refresh_ack** register (address=0x032) until the `readdatavalid` signal is asserted and the read data is `1'b1`, indicating that a refresh operation is in progress.

4. You can issue the next refresh request only after you see the acknowledge signal asserted (at time 4).
5. Write to the **mmr_refresh_req** register (address=0x02c) with data `0x0000_0000` to disable the refresh request.
6. You can implement a timer to track tRFC before sending the next user-controlled refresh.

12.4.7. Frequency of Operation

Certain frequencies of operation give you the best possible latency based on the memory parameters. The memory parameters you specify through the parameter editor are converted to clock cycles and rounded up.

In most cases, the frequency and parameter combination is not optimal. If you are using a memory device that has t_{RCD} = 15 ns and running the interface at 1200 MHz, you get the following results:

- For quarter-rate implementation (t_{Ck} = 3.33 ns):

  t_{RCD} convert to clock cycle = 15/3.33 = 4.5, rounded up to 5 clock cycles or 16.65 ns.

12.4.8. Series of Reads or Writes

Performing a series of reads or writes from the same bank and row increases controller efficiency.

The case shown in the second figure in the "Bank Management Efficiency" topic demonstrates that a read performed from the same row takes only 14.5 clock cycles to transfer data, making the controller 27% efficient.

Do not perform random reads or random writes. When you perform reads and writes to random locations, the operations require row and bank changes. To change banks, the controller must precharge the previous bank and activate the row in the new bank. Even if you change the row in the same bank, the controller has to close the bank (precharge) and reopen it again just to open a new row (activate). Because of the precharge and activate commands, efficiency can decrease by as much as 3–15%, as the controller needs more time to issue a read or write.

If you must perform a random read or write, use additive latency and bank interleaving to increase efficiency.

Controller efficiency depends on the method of data transfer between the memory device and the FPGA, the memory standards specified by the memory device vendor, and the type of memory controller.

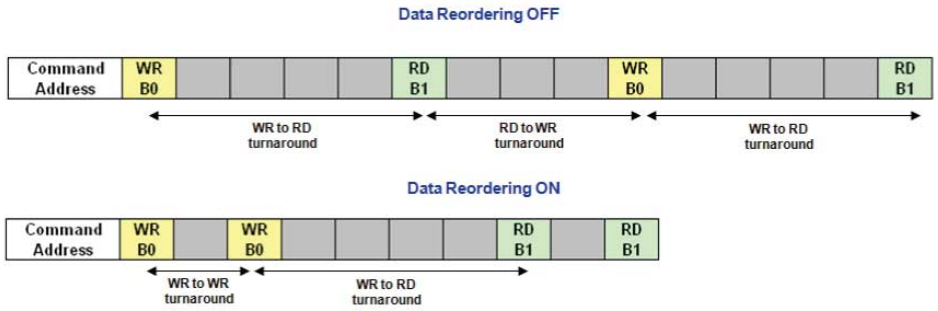
12.4.9. Data Reordering

Data reordering and command reordering can both contribute towards achieving controller efficiency.

The Data Reordering feature allows the single-port memory controller to change the order of read and write commands to achieve highest efficiency. You can enable data reordering by turning on **Enable Reordering** on the **Controller Settings** tab of the parameter editor.

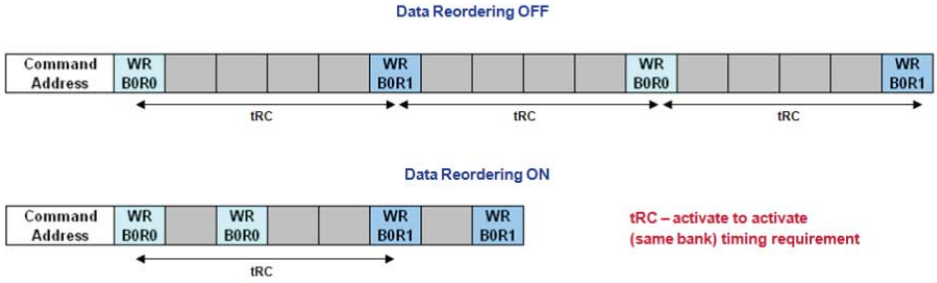
In the soft memory controller, inter-bank data reordering serves to minimize bus turnaround time by optimizing the ordering of read and write commands going to different banks; commands going to the same bank address are not reordered.

Figure 108. Data Reordering for Minimum Bus Turnaround



In the hard memory controller, inter-row data reordering serves to minimize t_{RC} by reordering commands going to different bank and row addresses; command going to the same bank and row address are not reordered. Inter-row data reordering inherits the minimum bus turnaround time benefit from inter-bank data reordering.

Figure 109. Data Reordering for Minimum t_{RC}



12.4.10. Starvation Control

The controller implements a starvation counter to ensure that lower-priority requests are not forgotten as higher-priority requests are reordered for efficiency.

In starvation control, a counter is incremented for every command served. You can set a starvation limit, to ensure that a waiting command is served immediately upon the starvation counter reaching the specified limit.

For example, if you set a starvation limit of 10, a lower-priority command is treated as high priority and served immediately, after ten other commands are served before it.

12.4.11. Command Reordering

Data reordering and command reordering can both contribute towards achieving controller efficiency. You can enable command reordering by turning on **Enable Reordering** on the **Controller Settings** tab of the parameter editor.

DDR protocols are naturally inefficient, because commands are fetched and processed sequentially. The DDRx command and DQ bus are not fully utilized as few potential cycles are wasted and degrading the efficiency

The command reordering feature, or look-ahead bank management feature, allows the controller to issue bank management commands early based on incoming patterns, so that when the command reaches the memory interface, the desired page in memory is already open.

The command cycles during the t_{RCD} period are idle and the bank-management commands are issued to next access banks. When the controller is serving the next command, the bank is already precharged. The command queue look-ahead depth is configurable from 1-16, to specify how many read or write requests the look-ahead bank management logic examines. With the look-ahead command queue, if consecutive write or read requests are to a sequential address with same row, same bank, and column incremental by 1, the controller merges the write or read requests at the memory transaction into a single burst.

Figure 110. Comparison With and Without Look-Ahead Bank Management Feature

Without Lookahead			With Lookahead		
Cycle	Command	Data	Cycle	Command	Data
1	ACT	tRCD	1	ACT	
2	NOP		2		
3	NOP		3		
4	READ		4	READ	
5		DATA0 (Burst 0, Burst 1)	5	ACT	DATA0 (Burst 0, Burst 1)
6	NOP	DATA0 (Burst 2, Burst 3)	6	NOP	DATA0 (Burst 2, Burst 3)
7	ACT	DATA0 (Burst 4, Burst 5)	7	ACT	DATA0 (Burst 4, Burst 5)
8	NOP	DATA0 (Burst 6, Burst 7)	8	READ	DATA0 (Burst 6, Burst 7)
9	NOP	Wasted Cycle	9	NOP	DATA1 (Burst 0, Burst 1)
10	READ	Wasted Cycle	10	NOP	DATA1 (Burst 2, Burst 3)
11		DATA1 (Burst 0, Burst 1)	11	NOP	DATA1 (Burst 4, Burst 5)
12	NOP	DATA1 (Burst 2, Burst 3)	12	READ	DATA1 (Burst 6, Burst 7)
13	ACT	DATA1 (Burst 4, Burst 5)	13	NOP	DATA2 (Burst 0, Burst 1)
14	NOP	DATA1 (Burst 6, Burst 7)	14	NOP	DATA2 (Burst 2, Burst 3)
15	NOP	Wasted Cycle	15	NOP	DATA2 (Burst 4, Burst 5)
16	READ	Wasted Cycle	16	NOP	DATA2 (Burst 6, Burst 7)
17	NOP	DATA2 (Burst 0, Burst 1)			
18	NOP	DATA2 (Burst 2, Burst 3)			
19	NOP	DATA2 (Burst 4, Burst 5)			
20	NOP	DATA2 (Burst 6, Burst 7)			

Command	Address	Condition
Read	Bank 0	Activate required
Read	Bank 1	Precharge required
Read	Bank 2	Precharge required

Compare the following efficiency results for the above figure:

Table 350. Efficiency Results for Above Figure

	Without Look-ahead Bank Management	With Look-ahead Bank Management
Active cycles of data transfer	12	12
Total number of cycles	20	16
Approximate efficiency	60%	75%

In the above table, the use of look-ahead bank management increases efficiency by 15%. The bank look-ahead pattern verifies that the system is able to completely hide the bank precharge and activation for specific sequences in which the minimum number of page-open transactions are placed between transactions to closed pages to allow bank look-ahead to occur just in time for the closed pages. An optimal system would completely hide bank activation and precharge performance penalties for the bank look-ahead traffic pattern and achieve 100% efficiency, ignoring refresh.

12.4.12. Bandwidth

Bandwidth depends on the efficiency of the memory controller controlling the data transfer to and from the memory device.

You can express bandwidth as follows:

Bandwidth = data width (bits) × data transfer rate (1/s) × efficiency.

Data rate transfer (1/s) = 2 × frequency of operation (4 × for QDR SRAM interfaces).

The following example shows the bandwidth calculation for a 16-bit interface that has 70% efficiency and runs at 200 MHz frequency:

Bandwidth = 16 bits × 2 clock edges × 1200 MHz × 70% = 26.88 Gbps.

DRAM typically has an efficiency of around 70%, but when you use the memory controller, efficiency can vary from 10 to 92%.

In QDR II+ or QDR II SRAM the IP implements two separate unidirectional write and read data buses, so the data transfer rate is four times the clock rate. The data transfer rate for a 400-MHz interface is 1, 600 Mbps. The efficiency is the percentage of time the data bus is transferring data. It is dependent on the type of memory. For example, in a QDR II+ or QDR II SRAM interface with separate write and read ports, the efficiency is 100% when there is an equal number of read and write operations on these memory interfaces.

12.4.13. Enable Command Priority Control

The **Enable Command Priority Control** option allows you to assign priority to read or write commands.

With knowledge of traffic patterns, you can identify certain read or write requests that the controller should treat as high priority. The controller issues high priority commands sooner, to reduce latency.

To enable user-requested command priority control on the controller top level, select **Enable Command Priority Control** on the **Controller Settings** tab.

13. Intel Stratix 10 EMIF IP Debugging

This chapter discusses issues and strategies for debugging your external memory interface IP.

For support resources for external memory interface debugging, visit the External Memory Interfaces Support Center on `www.intel.com`.

Related Information

- Intel FPGA IP for External Memory Interfaces - Support Center
- Timing Closure

13.1. Interface Configuration Performance Issues

There are many interface combinations and configurations possible in an Intel design, therefore it is impractical for Intel to explicitly state the achievable f_{MAX} for every combination.

Intel seeks to provide guidance on typical performance, but this data is subject to memory component timing characteristics, interface widths, depths directly affecting timing deration requirements, and the achieved skew and timing numbers for a specific PCB.

FPGA timing issues should generally not be affected by interface loading or layout characteristics. In general, the Intel performance figures for any given device family and speed-grade combination should usually be achievable.

To resolve FPGA (PHY and PHY reset) timing issues, refer to the *Analyzing Timing of Memory IP* chapter.

Achievable interface timing (address and command, half-rate address and command, read and write capture) is directly affected by any layout issues (skew), loading issues (deration), signal integrity issues (crosstalk timing deration), and component speed grades (memory timing size and tolerance). Intel performance figures are typically stated for the default (single rank, unbuffered DIMM) case. Intel provides additional expected performance data where possible, but the f_{MAX} is not achievable in all configurations. Intel recommends that you optimize the following items whenever interface timing issues occur:

- Improve PCB layout tolerances
- Use a faster speed grade of memory component
- Ensure that the interface is fully and correctly terminated
- Reduce the loading (reduce the deration factor)

13.1.1. Interface Configuration Bottleneck and Efficiency Issues

Depending on the transaction types, efficiency issues can exist where the achieved data rate is lower than expected. Ideally, these issues should be assessed and resolved during the simulation stage because they are sometimes impossible to solve later without rearchitecting the product.

Any interface has a maximum theoretical data rate derived from the clock frequency, however, in practice this theoretical data rate can never be achieved continuously due to protocol overhead and bus turnaround times.

Simulate your desired configuration to ensure that you have specified a suitable external memory family and that your chosen controller configuration can achieve your required bandwidth.

Efficiency can be assessed in several different ways, and the primary requirement is an achievable continuous data rate. The local interface signals combined with the memory interface signals and a command decode trace should provide adequate visibility of the operation of the IP to understand whether your required data rate is sufficient and the cause of the efficiency issue.

To show if under ideal conditions the required data rate is possible in the chosen technology, follow these steps:

1. Use the memory vendors own testbench and your own transaction engine.
2. Use either your own driver, or modify the provided example driver, to replicate the transaction types typical of your system.
3. Simulate this performance using your chosen memory controller and decide if the achieved performance is still acceptable.

Observe the following points that may cause efficiency or bottleneck issues at this stage:

- Identify the memory controller rate (full, half, or quarter) and commands, which may take two or four times longer than necessary
- Determine whether the memory controller is starved for data by observing the appropriate request signals.
- Determine whether the memory controller processor transactions at a rate sufficient to meet throughput requirements by observing appropriate signals, including the local ready signal.

Intel has several versions and types of memory controller, and where possible you can evaluate different configurations based on the results of the first tests.

Consider using either a faster interface, or a different memory type to better align your data rate requirements to the IP available directly from Intel.

Intel also provides stand-alone PHY configurations so that you may develop custom controllers or use third-party controllers designed specifically for your requirements.

13.2. Functional Issue Evaluation

Functional issues occur at all frequencies (using the same conditions) and are not altered by speed grade, temperature, or PCB changes. You should use functional simulation to evaluate functional issues.

The Intel FPGA IP includes the option to autogenerate a testbench specific to your IP configuration, which provides an easy route to functional verification.

The following issues should be considered when trying to debug functional issues in a simulation environment.

13.2.1. Intel IP Memory Model

Intel memory IP autogenerates a generic simplified memory model that works in all cases. This simple read and write model is not designed or intended to verify all entered IP parameters or transaction requirements.

The Intel-generated memory model may be suitable to evaluate some limited functional issues, but it does not provide comprehensive functional simulation.

13.2.2. Vendor Memory Model

Contact the memory vendor directly, because many additional models are available from the vendor's support system.

When using memory vendor models, ensure that the model is correctly defined for the following characteristics:

- Speed grade
- Organization
- Memory allocation
- Maximum memory usage
- Number of ranks on a DIMM
- Buffering on the DIMM
- ECC

Note: Refer to the **readme.txt** file supplied with the memory vendor model, for more information about how to define this information for your configuration. Also refer to Transcript Window Messages, for more information.

Note: Intel does not provide support for vendor-specific memory models.

During simulation vendor models output a wealth of information regarding any device violations that may occur because of incorrectly parameterized IP.

13.2.3. Transcript Window Messages

When you are debugging a functional issue in simulation, vendor models typically provide much more detailed checks and feedback regarding the interface and their operational requirements than the Intel generic model.

In general, you should use a vendor-supplied model whenever one is available. Consider using second-source vendor models in preference to the Intel generic model.

Many issues can be traced to incorrectly configured IP for the specified memory components. Component data sheets usually contain settings information for several different speed grades of memory. Be aware data sheet specify parameters in fixed units of time, frequencies, or clock cycles.

The Intel generic memory model always matches the parameters specified in the IP, as it is generated using the same engine. Because vendor models are independent of the IP generation process, they offer a more robust IP parameterization check.

During simulation, review the transcript window messages and do not rely on the Simulation Passed message at the end of simulation. This message only indicates that the example driver successfully wrote and then read the correct data for a single test cycle.

Even if the interface functionally passes in simulation, the vendor model may report operational violations in the transcript window. These reported violations often specifically explain why an interface appears to pass in simulation, but fails in hardware.

Vendor models typically perform checks to ensure that the following types of parameters are correct:

- Burst length
- Burst order
- tMRD
- tMOD
- tRFC
- tREFPDEN
- tRP
- tRAS
- tRC
- tACTPDEN
- tWR
- tWRPDEN
- tRTP
- tRDPDEN
- tINIT
- tXPDLL
- tCKE
- tRRD
- tCCD
- tWTR
- tXPR
- PRECHARGE
- CAS length
- Drive strength
- AL
- tDQS
- CAS_WL

- Refresh
- Initialization
- tIH
- tIS
- tDH
- tDS

If a vendor model can verify all these parameters are compatible with your chosen component values and transactions, it provides a specific insight into hardware interface failures.

13.2.4. Modifying the Example Driver to Replicate the Failure

Often during debugging, you may discover that the example driver design works successfully, but that your custom logic encounters data errors.

When the example design works but your custom design doesn't, the underlying problem may be either of the following:

- Related to the way that the local interface transactions are occurring. You should probe and compare using the Signal Tap II analyzer.
- Related to the types or format of transactions on the external memory interface. You should try modifying the example design to replicate the problem.

Typical issues on the local interface side include:

- Incorrect local-address-to-memory-address translation causing the word order to be different than expected. Refer to *Burst Definition* in your memory vendor data sheet.
- Incorrect timing on the local interface. When your design requests a transaction, the local side must be ready to service that transaction as soon as it is accepted without any pause.
- For more information, refer to the *Avalon® Interface Specification*.

The default example driver performs only a limited set of transaction types, consequently potential bus contention or preamble and postamble issues can often be masked in its default operation. For successful debugging, isolate the custom logic transaction types that are causing the read and write failures and modify the example driver to demonstrate the same issue. Then, you can try to replicate the failure in RTL simulation with the modified driver.

A problem that you can replicate in RTL simulation indicates a potential bug in the IP. You should recheck the IP parameters. A problem that you can not replicate in RTL simulation indicates a timing issue on the PCB. You can try to replicate the issue on an Intel development platform to rule out a board issue.

Note: Ensure that all PCB timing, loading, skew, and deration information is correctly defined in the Intel Quartus Prime software. The timing report is inaccurate if this initial data is not correct.

Functional simulation allows you to identify any issues with the configuration of either the memory controller or the PHY. You can then check the operation against both the memory vendor data sheet and the respective JEDEC specification. After you resolve functional issues, you can start testing hardware.

For more information about simulation, refer to the Simulation chapter.

Related Information

- Avalon Interface Specifications
- Intel Stratix 10 EMIF – Simulating Memory IP on page 133

13.3. Timing Issue Characteristics

The PHY and controller combinations autogenerate timing constraint files to ensure that the PHY and external interface are fully constrained and that timing is analyzed during compilation. However, timing issues can still occur. This topic discusses how to identify and resolve any timing issues that you may encounter.

Timing issues typically fall into two distinct categories:

- FPGA core timing reported issues
- External memory interface timing issues in a specific mode of operation or on a specific PCB

Timing Analyzer reports timing issues in two categories: core to core and core to IOE transfers. These timing issues include the PHY and PHY reset sections in the Timing Analyzer Report DDR subsection of timing analysis. External memory interface timing issues are specifically reported in the Timing Analyzer Report DDR subsection, excluding the PHY and PHY reset. The Report DDR PHY and PHY reset sections only include the PHY, and specifically exclude the controller, core, PHY-to-controller and local interface. Intel Quartus Prime timing issues should always be evaluated and corrected before proceeding to any hardware testing.

PCB timing issues are usually Intel Quartus Prime timing issues, which are not reported in the Intel Quartus Prime software, if incorrect or insufficient PCB topology and layout information is not supplied. PCB timing issues are typically characterized by calibration failure, or failures during user mode when the hardware is heated or cooled. Further PCB timing issues are typically hidden if the interface frequency is lowered.

13.3.1. Evaluating FPGA Timing Issues

Usually, you should not encounter timing issues with Intel-provided IP unless your design exceeds the published performance range or you are using a device for which the Intel Quartus Prime software offers only preliminary timing model support. Nevertheless, timing issues can occur in the following circumstances:

- The **.sdc** files are incorrectly added to the Intel Quartus Prime project
- Intel Quartus Prime analysis and synthesis settings are not correct
- Intel Quartus Prime Fitter settings are not correct

For all of these issues, refer to the correct user guide for more information about recommended settings and follow these steps:

1. Ensure that the IP generated **.sdc** files are listed in the Intel Quartus Prime Timing Analyzer files to include in the project window.
2. Ensure that **Analysis and Synthesis Settings** are set to **Optimization Technique Speed**.
3. Ensure that **Fitter Settings** are set to **Fitter Effort Standard Fit**.
4. Use **Timing Analyzer Report Ignored Constraints**, to ensure that **.sdc** files are successfully applied.
5. Use **Timing Analyzer Report Unconstrained Paths**, to ensure that all critical paths are correctly constrained.

More complex timing problems can occur if any of the following conditions are true:

- The design includes multiple PHY or core projects
- Devices where the resources are heavily used
- The design includes wide, distributed, maximum performance interfaces in large die sizes

Any of the above conditions can lead to suboptimal placement results when the PHY or controller are distributed around the FPGA. To evaluate such issues, simplify the design to just the autogenerated example top-level file and determine if the core meets timing and you see a working interface. Failure implies that a more fundamental timing issue exists. If the standalone design passes core timing, evaluate how this placement and fit is different than your complete design.

Use Logic Lock (Standard)regions, or design partitions to better define the placement of your memory controllers. When you have your interface standalone placement, repeat for additional interfaces, combine, and finally add the rest of your design.

Additionally, use fitter seeds and increase the placement and router effort multiplier.

13.3.2. Evaluating External Memory Interface Timing Issues

External memory interface timing issues usually relate to the FPGA input and output characteristics, PCB timing, and the memory component characteristics.

The FPGA input and output characteristics are usually fixed values, because the IOE structure of the devices is fixed. Optimal PLL characteristics and clock routing characteristics do have an effect. Assuming the IP is correctly constrained with autogenerated assignments, and you follow implementation rules, the design should reach the stated performance figures.

Memory component characteristics are fixed for any given component or DIMM. Consider using faster components or DIMMs in marginal cases when PCB skew may be suboptimal, or your design includes multiple ranks when deration may cause read capture or write timing challenges. Using faster memory components often reduces the memory data output skew and uncertainty easing read capture, and lowering the memory's input setup and hold requirement, which eases write timing.

Increased PCB skew reduces margins on address, command, read capture and write timing. If you are narrowly failing timing on these paths, consider reducing the board skew (if possible), or using faster memory. Address and command timing typically requires you to manually balance the reported setup and hold values with the dedicated address and command phase in the IP.

Refer to the respective IP user guide for more information.

Multiple-slot multiple-rank UDIMM interfaces can place considerable loading on the FPGA driver. Typically a quad rank interface can have thirty-six loads. In multiple-rank configurations, Intel's stated maximum data rates are not likely to be achievable because of loading deration. Consider using different topologies, for example registered DIMMs, so that the loading is reduced.

Deration because of increased loading, or suboptimal layout may result in a lower than desired operating frequency meeting timing. You should close timing in the Timing Analyzer software using your expected loading and layout rules before committing to PCB fabrication.

Ensure that any design with an Intel PHY is correctly constrained and meets timing in the Timing Analyzer software. You must address any constraint or timing failures before testing hardware.

For more information about timing constraints, refer to the Timing Analysis chapter.

Related Information

Timing Closure

13.4. Verifying Memory IP Using the Signal Tap II Logic Analyzer

The Signal Tap II logic analyzer shows read and write activity in the system.

For more information about using the Signal Tap II logic analyzer, refer to the *Design Debugging Using the Signal Tap II Embedded Logic Analyzer* chapter in volume 3 of the *Intel Quartus Prime Handbook*

To add the Signal Tap II logic analyzer, follow these steps:

1. On the Tools menu click **Signal Tap II Logic Analyzer** .
2. In the **Signal Configuration** window next to the **Clock** box, click **...** (Browse Node Finder).
3. Type the memory interface system clock (typically * `phy_clk`) in the **Named** box, for **Filter** select **Signal Tap II: presynthesis** and click **List**.
4. Select the memory interface clock that is exposed to the user logic.
5. Click **OK**.
6. Under Signal Configuration, specify the following settings:
   - For **Sample depth**, select **512**
   - For **RAM type**, select **Auto**
   - For **Trigger flow control**, select **Sequential**
   - For **Trigger position**, select **Center trigger position**
   - For **Trigger conditions** , select **1**

7. On the Edit menu, click **Add Nodes**.
8. Search for specific nodes that you want to monitor, and click **Add**.

   Note: Signal Tap can probe only nodes that are exposed to FPGA core logic. Refer to pin descriptions for help in deciding which signals to monitor.
9. Decide which signal and event you want to trigger on, and set the corresponding trigger condition.
10. On the File menu, click **Save**, to save the Signal Tap II **. stp** file to your project.

    Note: If you see the message **Do you want to enable Signal Tap II file "stp1.stp" for the current project**, click **Yes**.
11. After you add signals to the Signal Tap II logic analyzer, recompile your design by clicking **Start Compilation** on the **Processing** menu.
12. Following compilation, verify that Timing Analyzer timing analysis passes successfully.
13. Connect the development board to your computer.
14. On the Tools menu, click **Signal Tap II Logic Analyzer**.
15. Add the correct `<project_name>.sof` file to the SOF Manager:
    a. Click **...** to open the **Select Program Files** dialog box.
    b. Select ***<your_project_name>*.sof**.
    c. Click **Open**.
    d. To download the file, click the **Program Device** button.
16. When the example design including Signal Tap II successfully downloads to your development board, click **Run Analysis** to run once, or click **Autorun Analysis** to run continuously.

Related Information

Design Debugging with the Signal Tap Logic Analyzer

13.4.1. Signals to Monitor with the Signal Tap II Logic Analyzer

This topic lists the memory controller signals you should consider analyzing for different memory interfaces. This list is not exhaustive, but is a starting point.

Monitor the following signals:

- `amm_addr`
- `amm_rdata`
- `amm_rdata_valid`
- `amm_read_req`
- `amm_ready`
- `amm_wdata`
- `amm_write_req`
- `fail`
- `pass`

- afi_cal_fail
- afi_cal_success
- test_complete
- be_reg (QDRII only)
- pnf_per_bit
- rdata_reg
- rdata_valid_reg
- data_out
- data_in
- written_data_fifo|data_out
- usequencer|state *
- usequencer|phy_seq_rdata_valid
- usequencer|phy_seq_read_fifo_q
- usequencer|phy_read_increment_vfifo *
- usequencer|phy_read_latency_counter
- uread_datapath|afi_rdata_en
- uread_datapath|afi_rdata_valid
- uread_datapath|ddio_phy_dq
- qvld_wr_address *
- qvld_rd_address *

13.5. Hardware Debugging Guidelines

Before debugging your design, confirm that it follows the recommended design flow.

Always keep a record of tests, to avoid repeating the same tests later. To start debugging the design, perform the following initial steps.

Related Information

Recommended Design Flow

13.5.1. Create a Simplified Design that Demonstrates the Same Issue

To help debugging, create a simple design that replicates the problem.

A simple design should compile quickly and be easy to understand. The EMIF IP generates an example top-level file that is ideal for debugging. The example top-level file uses all the same parameters, pin-outs, and so on.

Related Information

External Memory Interface Debug Toolkit

13.5.2. Measure Power Distribution Network

Measure voltages of the various power supplies on their hardware development platform over a suitable time base and with a suitable trigger.

Ensure that you use an appropriate probe and grounding scheme. In addition, take the measurements directly on the pins or vias of the devices in question, and with the hardware operational.

13.5.3. Measure Signal Integrity and Setup and Hold Margin

Measure the signals on the PCB. When measuring any signal, consider the edge rate of the signal, not just its frequency. Modern FPGA devices have very fast edge rates, therefore you must use a suitable oscilloscope, probe, and grounding scheme when you measure the signals.

You can take measurements to capture the setup and hold time of key signal classes with respect to their clock or strobe. Ensure that the measured setup and hold margin is at least better than that reported in the Intel Quartus Prime software. A worse margin indicates a timing discrepancy somewhere in the project; however, this issue may not be the cause of your problem.

13.5.4. Vary Voltage

Vary the voltage of your system, if you suspect a marginality issue.

Increasing the voltage usually causes devices to operate faster and also usually provides increased noise margin.

13.5.5. Operate at a Lower Speed

Test the interface at a lower speed. If the interface works at a lower speed, the interface is correctly pinned out and functional.

If the interface fails at a lower speed, determine if the test is valid. Many high-speed memory components have a minimal operating frequency, or require subtly different configurations when operating at a lower speeds.

For example, DDR3 SDRAM typically requires modification to the following parameters if you want to operate the interface at lower speeds:

- t_{MRD}
- t_{WTR}
- CAS latency and CAS write latency

13.5.6. Determine Whether the Issue Exists in Previous Versions of Software

Hardware that works before an update to either the Intel Quartus Prime software or the memory IP indicates that the development platform is not the issue.

However, the previous generation IP may be less susceptible to a PCB issue, masking the issue.

13.5.7. Determine Whether the Issue Exists in the Current Version of Software

Designs are often tested using previous generations of Intel software or IP.

Projects may not be upgraded for various reasons:

- Multiple engineers are on the same project. To ensure compatibility, a common release of Intel software is used by all engineers for the duration of the product development. The design may be several releases behind the current Intel Quartus Prime software version.
- Many companies delay before adopting a new release of software so that they can first monitor Internet forums to get a feel for how successful other users say the software is.
- Many companies never use the latest version of any software, preferring to wait until the first service pack is released that fixes the primary issues.
- Some users may only have a license for the older version of the software and can only use that version until their company makes the financial decision to upgrade.
- The local interface specification from Intel FPGA IP to the customer's logic sometimes changes from software release to software release. If you have already spent resources designing interface logic, you may be reluctant to repeat this exercise. If a block of code is already signed off, you may be reluctant to modify it to upgrade to newer IP from Intel.

In all of the above scenarios, you must determine if the issue still exists in the latest version of the Intel software. Bug fixes and enhancements are added to the Intel FPGA IP every release. Depending on the nature of the bug or enhancement, it may not always be clearly documented in the release notes.

Finally, if the latest version of the software resolves the issue, it may be easier to debug the version of software that you are using.

13.5.8. Try A Different PCB

If you are using the same Intel FPGA IP on several different hardware platforms, determine whether the problem occurs on all platforms or just on one.

Multiple instances of the same PCB, or multiple instances of the same interface, on physically different hardware platforms may exhibit different behavior. You can determine if the configuration is fundamentally not working, or if some form of marginality is involved in the issue.

Issues are often reported on the alpha build of a development platform. These are produced in very limited numbers and often have received limited bare-board testing, or functional testing. These early boards are often more unreliable than production quality PCBs.

Additionally, if the IP is from a previous project to help save development resources, determine whether the specific IP configuration works on a previous platform.

13.5.9. Try Other Configurations

Designs are often quite large, using multiple blocks of IP in many different combinations. Determine whether any other configurations work correctly on the development platform.

The full project may have multiple external memory controllers in the same device, or may have configurations where only half the memory width or frequency is required. Find out what does and does not work to help the debugging of the issue.

13.5.10. Debugging Checklist

The following checklist is a good starting point when debugging an external memory interface.

Table 351. Checklist

Check	Item
☐	Try a different fit.
☐	Check IP parameters at the operating frequency (t_{MRD}, t_{WTR} for example).
☐	Ensure you have constrained your design with proper timing deration and have closed timing.
☐	Simulate the design. If it fails in simulation, it shall fail in hardware.
☐	Analyze timing.
☐	Place and assign R_{ZQ} (OCT).
☐	Measure the power distribution network (PDN).
☐	Measure signal integrity.
☐	Measure setup and hold timing.
☐	Measure FPGA voltages.
☐	Vary voltages.
☐	Heat and cool the PCB.
☐	Operate at a lower or higher frequency.
☐	Check board timing and trace Information.
☐	Check LVDS and clock sources, I/O voltages and termination.
☐	Check PLL clock source, specification, and jitter.
☐	Retarget to a smaller interface width or a single bank.

13.6. Categorizing Hardware Issues

The following topics divide issues into categories. By determining which category (or categories) an issue belongs in, you may be able to better focus on the cause of the issue.

Hardware issues fall into three categories:

- Signal integrity issues
- Hardware and calibration issues
- Intermittent issues

13.6.1. Signal Integrity Issues

Many design issues, including some at the protocol layer, can be traced back to signal integrity problems. You should check circuit board construction, power systems, command, and data signaling to determine if they meet specifications.

If infrequent, random errors exist in the memory subsystem, product reliability suffers. Check the bare circuit board or PCB design file. Circuit board errors can cause poor signal integrity, signal loss, signal timing skew, and trace impedance mismatches. Differential traces with unbalanced lengths or signals that are routed too closely together can cause crosstalk.

13.6.1.1. Characteristics of Signal Integrity Issues

Signal integrity problems often appear when the performance of the hardware design is marginal.

The design may not always initialize and calibrate correctly, or may exhibit occasional bit errors in user mode. Severe signal integrity issues can result in total failure of an interface at certain data rates, and sporadic component failure because of electrical stress. PCB component variance and signal integrity issues often show up as failures on one PCB, but not on another identical board. Timing issues can have a similar characteristic. Multiple calibration windows or significant differences in the calibration results from one calibration to another can also indicate signal integrity issues.

13.6.1.2. Evaluating SignaI Integrity Issues

Signal integrity problems can only really be evaluated in two ways:

- direct measurement using suitable test equipment like an oscilloscope and probe
- simulation using a tool like HyperLynx or Allegro PCB SI

Compare signals to the respective electrical specification. You should look for overshoot and undershoot, non-monotonicity, eye height and width, and crosstalk.

13.6.1.2.1. Skew

Ensure that all clocked signals, commands, addresses, and control signals arrive at the memory inputs at the same time.

Trace length variations cause data valid window variations between the signals, reducing margin. For example, DDR3-800 at 400 MHz has a data valid window that is smaller than 1,250 ps. Trace length skew or crosstalk can reduce this data valid window further, making it difficult to design a reliably operating memory interface.

Ensure that the skew figure previously entered into the Intel FPGA IP matches that actually achieved on the PCB, otherwise Intel Quartus Prime timing analysis of the interface is accurate.

13.6.1.2.2. Crosstalk

Crosstalk is best evaluated early in the memory design phase.

Check the clock-to-data strobes, because they are bidirectional. Measure the crosstalk at both ends of the line. Check the data strobes to clock, because the clocks are unidirectional, these only need checking at the memory end of the line.

13.6.1.2.3. Power System

Some memory interfaces draw current in spikes from their power delivery system as SDRAMs are based on capacitive memory cells.

Rows are read and refreshed one at a time, which causes dynamic currents that can stress any power distribution network (PDN). The various power rails should be checked either at or as close as possible to the SDRAM power pins. Ideally, you should use a real-time oscilloscope set to fast glitch triggering to check the power rails.

13.6.1.2.4. Clock Signals

The clock signal quality is important for any external memory system.

Measurements include frequency, digital core design (DCD), high width, low width, amplitude, jitter, rise, and fall times.

13.6.1.2.5. Read Data Valid Window and Eye Diagram

The memory generates the read signals. Take measurements at the FPGA end of the line.

To ease read diagram capture, modify the example driver to mask writes or modify the PHY to include a signal that you can trigger on when performing reads.

13.6.1.2.6. Write Data Valid Window and Eye Diagram

The FPGA generates the write signals. Take measurements at the memory device end of the line.

To ease write diagram capture, modify the example driver to mask reads or modify the PHY export a signal that is asserted when performing writes.

13.6.1.2.7. OCT and ODT Usage

Modern external memory interface designs typically use OCT for the FPGA end of the line, and ODT for the memory component end of the line. If either the OCT or ODT are incorrectly configured or enabled, signal integrity problems occur.

If the design uses OCT, the R_{ZQ} pin must be placed correctly for the OCT to work. If you do not place the R_{ZQ} pin, the Intel Quartus Prime software allocates them automatically with the following warning:

```
Critical Warning(12677): No exact pin location assignment(s)
for 1 pins of 122 total pins. For the list of pins please refer to
the I/O Assignment Warnings table in the fitter report.
```

If you see these warnings, the R_{ZQ} pin may have been allocated to a pin that does not have the required external resistor present on the board. This allocation renders the OCT circuit faulty, resulting in unreliable calibration and or interface behavior. The pins with the required external resistor must be specified in the Intel Quartus Prime software.

For the FPGA, ensure that you perform the following:

- Connect the R_{ZQ} pin to the correct resistors and pull-down to ground in the schematic or PCB.
- Contain the R_{ZQ} pins within a bank of the device that is operating at the same VCCIO voltage as the interface that is terminated.
- Review the Fitter Pin-Out file for R_{ZQ} pins to ensure that they are on the correct pins, and that only the correct number of calibration blocks exists in your design.
- Check in the fitter report that the input, output, and bidirectional signals with calibrated OCT all have the termination control block applicable to the associated R_{ZQ} pins.

For the memory components, ensure that you perform the following:

- Connect the required resistor to the correct pin on each and every component, and ensure that it is pulled to the correct voltage.
- Place the required resistor close to the memory component.
- Correctly configure the IP to enable the desired termination at initialization time.
- Check that the speed grade of memory component supports the selected ODT setting.
- Check that the second source part that may have been fitted to the PCB, supports the same ODT settings as the original

13.6.2. Hardware and Calibration Issues

Hardware and calibration issues have the following definitions:

- Calibration issues result in calibration failure, which usually causes the `ctl_cal_fail` signal to be asserted.
- Hardware issues result in read and write failures, which usually causes the pass not fail (`pnf`) signal to be asserted.

Note: Ensure that functional, timing, and signal integrity issues are not the direct cause of your hardware issue, as functional, timing or signal integrity issues are usually the cause of any hardware issue.

13.6.2.1. Postamble Timing Issues and Margin

The postamble timing is set by the PHY during calibration.

You can diagnose postamble issues by viewing the `pnf_per_byte` signal from the example driver. Postamble timing issues mean only read data is corrupted during the last beat of any read request.

13.6.2.2. Intermittent Issue Evaluation

Intermittent issues are typically the hardest type of issue to debug—they appear randomly and are hard to replicate.

Errors that occur during run-time indicate a data-related issue, which you can identify by the following actions:

- Add the Signal Tap II logic analyzer and trigger on the post-trigger `pnf`
- Use a stress pattern of data or transactions, to increase the probability of the issue
- Heat up or cool down the system
- Run the system at a slightly faster frequency

If adding the Signal Tap II logic analyzer or modifying the project causes the issue to go away, the issue is likely to be placement or timing related.

Errors that occur at start-up indicate that the issue is related to calibration, which you can identify by the following actions:

- Modify the design to continually calibrate and reset in a loop until the error is observed
- Where possible, evaluate the calibration margin either from the debug toolkit or system console.
- Identify the calibration error stage from the debug toolkit, and use this information with whatever specifically occurs at that stage of calibration to assist with your debugging of the issue.

Related Information

External Memory Interface Debug Toolkit

13.7. Debugging Intel Stratix 10 EMIF IP

You can debug hardware failures by connecting to the Legacy EMIF Debug Toolkit or the EMIF Unified Calibration Debug Toolkit, or by exporting an Avalon-MM slave port, from which you can access information gathered during calibration. You can also connect to this port to mask ranks and to request recalibration.

About the Legacy EMIF Debug Toolkit and the EMIF Unified Calibration Debug Toolkit

Commencing with the Intel Stratix 10 EMIF IP version 19.2.1 (Intel Quartus Prime software version 20.2), two debug toolkits are available. In general, the two toolkits offer many similar features, however the Unified Toolkit does not require an installation of the Intel Quartus Prime software. In addition, it can read ISSPs and easily rerun the traffic generator.

The following table summarizes the features of the two debug toolkits:

Feature	Supported by the Legacy EMIF Debug Toolkit?	Supported by the EMIF Unified Calibration Debug Toolkit?
Protocol support	All protocols.	DDR4 only.
Can run in System Console — no full Intel Quartus Prime software required	Not supported	Yes
Reading Memory Configuration	Yes	Yes
Reading Data Pin Calibration Margins/delay settings	Yes	Yes
Reading A/C Margins/delay settings	Yes	Not supported
Reading Vref margins/settings	Yes	Yes
Rerunning Calibration	Yes	Yes
Reading Calibration Status Report	Yes	Yes
Rerunning traffic generator (through ISSPs)	Not supported	Yes
Driver Margining	Yes	Yes
ODT Calibration	Yes	Yes
Vref Margining	Yes	Yes
Manually Adjusting Pin Delays	Yes	Yes
Graphic representations of margins	Yes	Yes
Reading and Writing to all ISSPs	Not supported	Yes

Accessing the Exported Avalon-MM Port

You can access the exported Avalon-MM port in two ways:

- Via the External Memory Interface Debug Toolkit
- Via On-Chip Debug (core logic on the FPGA)

13.7.1. Debugging With the Legacy External Memory Interface Debug Toolkit

The Legacy External Memory Interface Debug Toolkit provides access to data collected by the Nios II sequencer during memory calibration, and allows you to perform certain tasks.

The Legacy External Memory Interface Debug Toolkit provides access to data including the following:

- General interface information, such as protocol and interface width
- Calibration results per group, including pass/fail status, failure stage, and delay settings

You can also perform the following tasks:

- Mask ranks from calibration (you might do this to skip specific ranks)
- Request recalibration of the interface

13.7.1.1. User Interface

The EMIF toolkit provides a graphical user interface for communication with connections.

All functions provided in the toolkit are also available directly from the `quartus_sh` TCL shell, through the `external_memif_toolkit` TCL package. The availability of TCL support allows you to create scripts to run automatically from TCL. You can find information about specific TCL commands by running `help -pkg external_memif_toolkit` from the `quartus_sh` TCL shell.

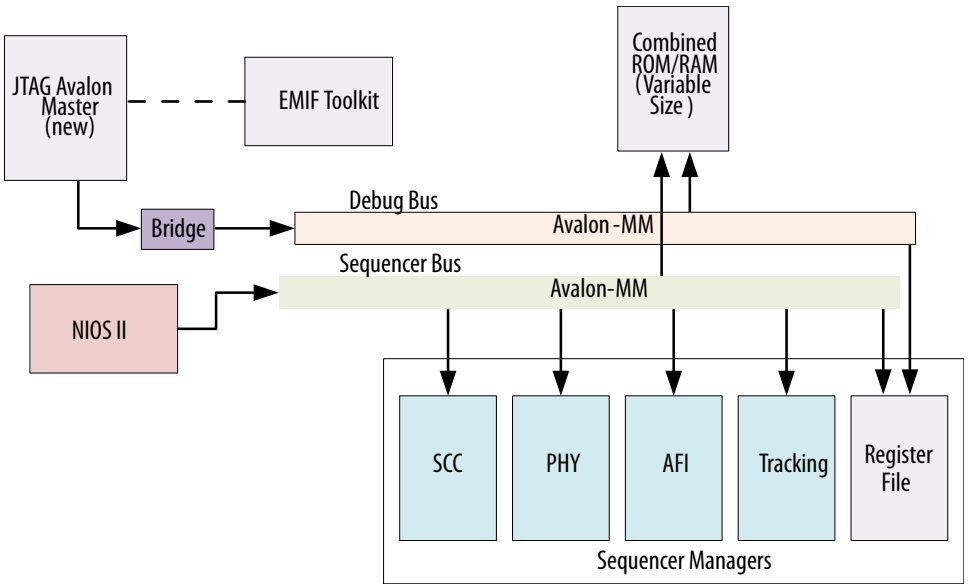
If you want, you can begin interacting with the toolkit through the GUI, and later automate your workflow by creating TCL scripts. The toolkit GUI records a history of the commands that you run. You can see the command history on the History tab in the toolkit GUI.

13.7.1.2. Communication

Communication between the EMIF Toolkit and external memory interface connections is achieved using a JTAG Avalon-MM master attached to the sequencer bus.

The following figure shows the structure of EMIF IP with JTAG Avalon-MM master attached to sequencer bus masters.

Figure 111. EMIF IP with JTAG Avalon-MM Master



13.7.1.3. Setup and Use

Before using the EMIF Toolkit, you should compile your design and program the target device with the resulting SRAM Object File (**. sof**). For designs compiled in the Intel Quartus Prime software, all debugging information resides in the **.sof** file.

You can run the toolkit using all your project files, or using only the Intel Quartus Prime Project File (**.qpf**), Intel Quartus Prime Settings File (**.qsf**), and **.sof** file.

After you have programmed the target device, you can run the EMIF Toolkit and open your project. You can then use the toolkit to create connections to the external memory interface.

13.7.1.4. Configuring Your EMIF IP for Use with the Legacy Debug Toolkit

The Intel Stratix 10 EMIF Debug Interface IP core contains the access point through which the Legacy EMIF Debug Toolkit reads calibration data collected by the Nios II sequencer.

Connecting an EMIF IP Core to an Intel Stratix 10 EMIF Debug Interface

For the EMIF Debug Toolkit to access the calibration data for a Intel Stratix 10 EMIF IP core, you must connect one of the EMIF cores in each I/O column to a Intel Stratix 10 EMIF Debug Interface IP core. Subsequent EMIF IP cores in the same column must connect in a daisy chain to the first.

There are two ways that you can add the Intel Stratix 10 EMIF Debug Interface IP core to your design:

- When you generate your EMIF IP core, on the **Diagnostics** tab, select **Add EMIF Debug Interface** for the **EMIF Debug Toolkit/On-Chip Debug Port**; you do not have to separately instantiate a Intel Stratix 10 EMIF Debug Interface core. This method does not export an Avalon-MM slave port. You can use this method if you require only EMIF Debug Toolkit access to this I/O column; that is, if you do not require On-Chip Debug Port access, or PHYLite reconfiguration access.
- When you generate your EMIF IP core, on the **Diagnostics** tab, select **Export** for the **EMIF Debug Toolkit/On-Chip Debug Port**. Then, separately instantiate an Intel Stratix 10 EMIF Debug Interface core and connect its `to_ioaux` interface to the `cal_debug` interface on the EMIF IP core. This method is appropriate if you want to also have On-Chip Debug Port access to this I/O column, or PHYLite reconfiguration access.

For each of the above methods, you must assign a unique interface ID for each external memory interface in the I/O column, to identify that interface in the Legacy Debug Toolkit. You can assign an interface ID using the dropdown list that appears when you enable the **Debug Toolkit/On-Chip Debug Port** option.

Connecting an EMIF IP Core and PHYLite Core

If you place any PHYLite cores with dynamic reconfiguration enabled into the same I/O column as an EMIF IP core, you should instantiate and connect the PHYLite cores in a similar way. See the *Intel FPGA PhyLite for Parallel Interfaces IP Core User Guide* for more information.

13.7.1.4.1. Daisy-Chaining Additional EMIF IP Cores for Debugging

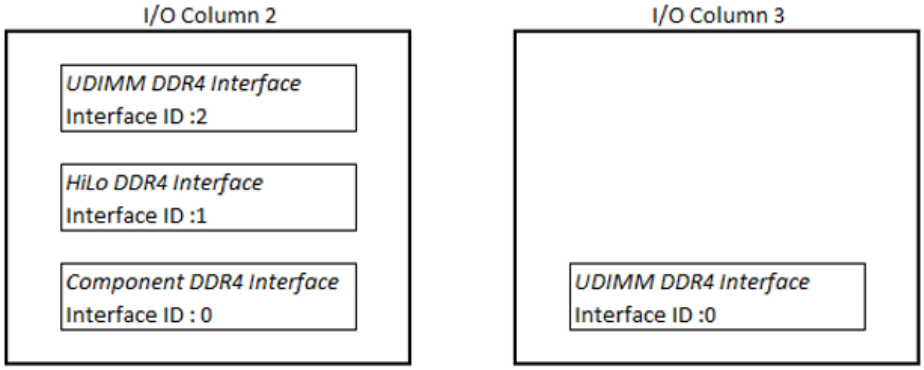
After you have connected a Intel Stratix 10 EMIF Debug Interface to one of the EMIF IP cores in an I/O column, you must then connect subsequent EMIF IP cores in that column in a daisy-chain manner.

If you don't require debug capabilities for a particular EMIF IP core, you do not have to connect that core to the daisy chain.

Example of Daisy-Chaining Multiple EMIF Cores

This example assumes a total of four EMIF IP cores, with three residing in column 2 and one residing in column 3. In this example, column 2 has a DDR4 component, HiLo, and UDIMM EMIF interfaces, and column 3 has a DDR4 UDIMM interface.

Figure 112. EMIF IP Cores in the Example



To create a daisy chain of EMIF IP cores, follow these steps:

1. On the first EMIF IP core, select **Add EMIF Debug Interface for EMIF Debug Toolkit/On-Chip Debug Port**.
2. Select **Enable Daisy-Chaining for EMIF Debug Toolkit/On-Chip Debug Port** to create an Avalon-MM interface called `cal_debug_out`.
3. Select **First EMIF Instance in the Avalon Chain**.
4. Set **Interface ID** to 0. You can start **Interface ID** at any number, so long as you select **First EMIF Instance in the Avalon** for the first EMIF IP core in a column.

Figure 113. Calibration Debug Options for First EMIF IP Core (Component Interface)

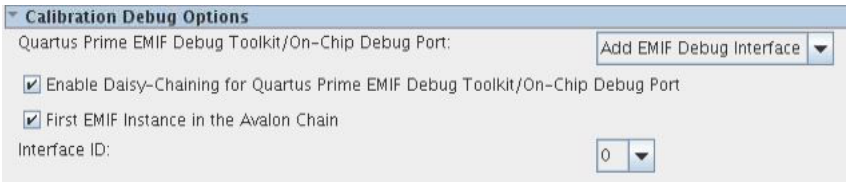
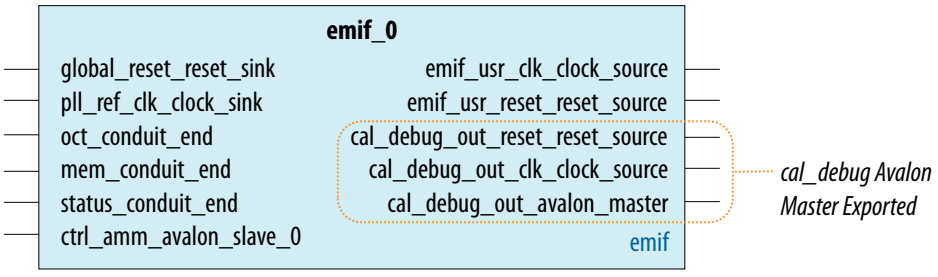


Figure 114. EMIF with EMIF Debug Interface and Daisy-Chaining Enabled



Subsequent EMIF IP cores in the same column require an incremental **Interface ID** value. For ease of use, you can start with an **Interface ID** value of 0 for the first EMIF IP core in a column. For two EMIF IP cores in two different columns, each IP core can have an **Interface ID** value beginning at 0, with the value incremented for each subsequent EMIF IP core in the same column.

5. On the second EMIF IP core in the same column, select **Export** as the **EMIF Debug Toolkit/On-Chip Debug Port** mode, to export an Avalon-MM interface called `cal_debug`. Also select the **Enable Daisy-Chaining for EMIF Debug Toolkit/On-Chip Debug Port** option.

Figure 115. Calibration Debug Options for the Second EMIF IP Core

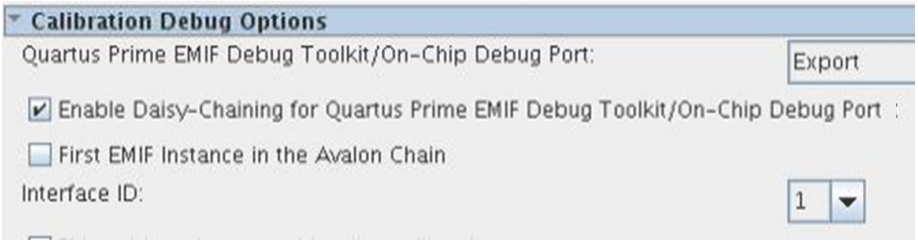
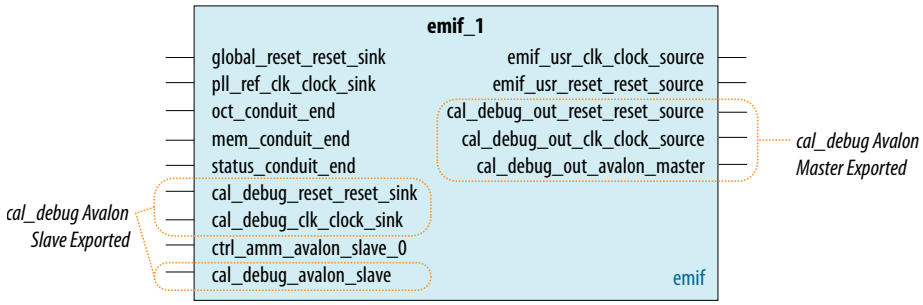


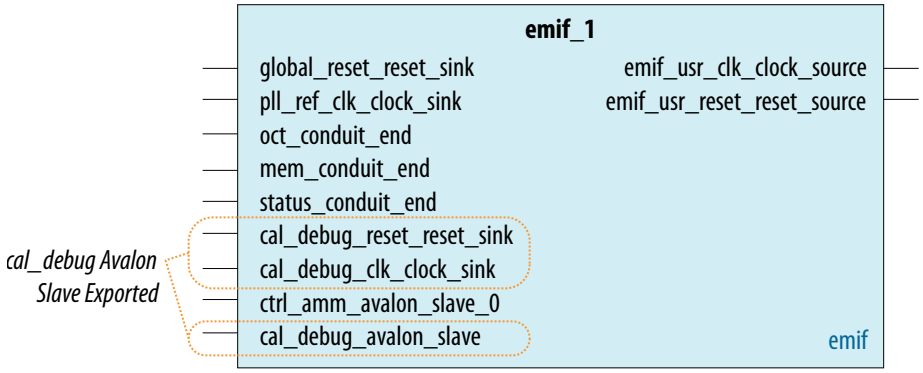
Figure 116.



6. On the last EMIF IP core in the same column, select **Export** as the **EMIF Debug Toolkit/On-Chip Debug Port** mode. For the last EMIF IP in the debug daisy chain, do not select the **Enable Daisy-Chaining for EMIF Debug Toolkit/On-Chip Debug Port** option.

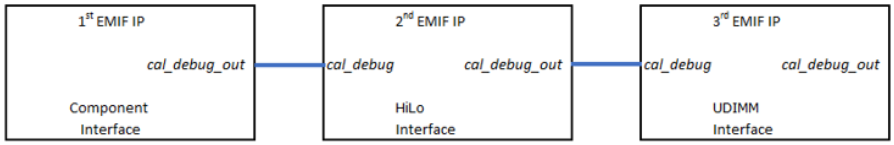
Figure 117. Calibration Debug Options for the Last EMIF IP Core (UDIMM interface)

Figure 118. EMIF with EMIF Debug Interface Exported and Daisy-Chain Disabled



7. Connect the IP cores on I/O column 2 as shown below.

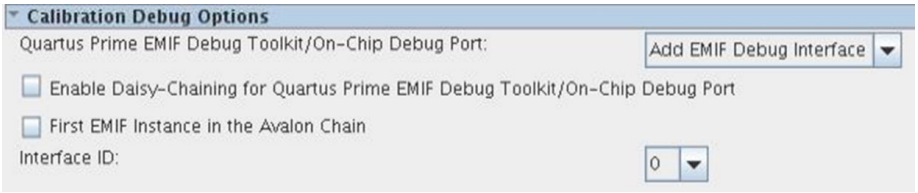
Figure 119. Daisy-Chain of Multiple EMIF IP Cores in I/O column 2



- Connect the `cal_debug_out` interface of the first EMIF IP core (the component interface in the above example) to the `cal_debug` interface of the second EMIF IP core (the HiLo interface in the above example).
- Connect the `cal_debug_out` interface of the second EMIF IP core (the HiLo interface in the above example) to the `cal_debug` interface of the third EMIF IP core (the UDIMM interface in the above example).

The following figure shows the setting on a single EMIF IP core in columns 3. Daisy-chaining is not required on this interface as there is only one EMIF IP core in column 3.

Figure 120. UDIMM Calibration Debug Options



Connecting an EMIF IP Core and a PHYLite Core.

If you place any PHYLite cores with dynamic reconfiguration enabled into the same I/O column as an EMIF IP core, you should instantiate and connect the PHYLite cores in a similar way. See the *Intel FPGA PHYLite for Parallel Interfaces IP Core User Guide* for more information.

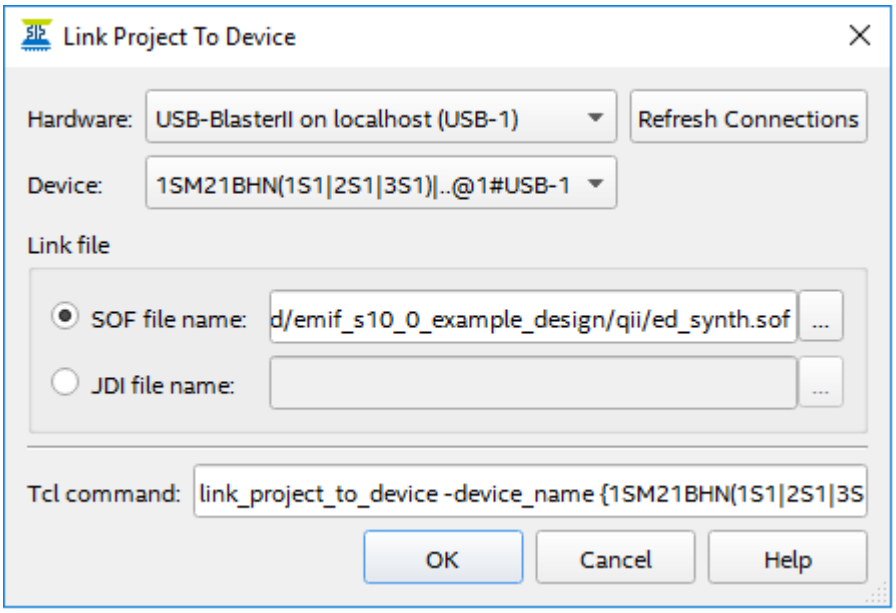
13.7.1.4.2. General Workflow

To use the EMIF Toolkit, you must link your compiled project to a device, and create a communication channel to the connection that you want to examine.

13.7.1.4.3. Linking the Project to a Device

1. To launch the toolkit, select External Memory Interface Toolkit from the Tools menu in the Intel Quartus Prime software.
2. After you have launched the toolkit, open your project and click the **Initialize connections** task in the **Tasks** window, to initialize a list of all known connections.
3. To link your project to a specific device on specific hardware, perform the following steps:
   a. Click the **Link Project to Device** task in the **Tasks** window.
   b. Select the desired hardware from the **Hardware** dropdown menu in the **Link Project to Device** dialog box.
   c. Select the desired device on the hardware from the **Device** dropdown menu in the **Link Project to Device** dialog box.
   d. Select **SOF** as the **Link file type**, verify that the **.sof** file is correct for your programmed device, and click **Ok.**

Figure 121. Link Project to Device Dialog Box



For designs compiled in the Intel Quartus Prime software, the SOF file contains a design hash to ensure the SOF file used to program the device matches the SOF file specified for linking to a project. If the hash does not match, an error message appears.

If the toolkit successfully verifies all connections, it then attempts to determine the connection type for each connection. Connections of a known type are listed in the Linked Connections report, and are available for the toolkit to use.

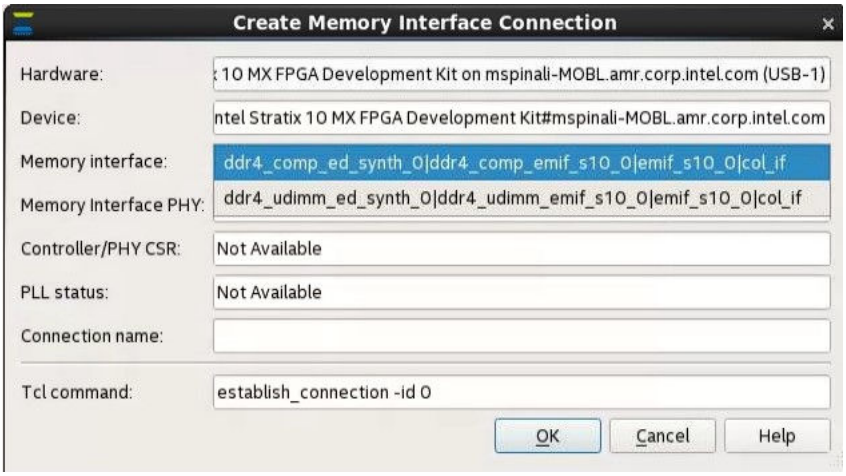
13.7.1.4.4. Establishing Communication to Connections

After you have completed linking the project, you can establish communication to the connections.

1. In the Tasks window,
   - Click **Create Memory Interface Connection** to create a connection to the external memory interface.
   - Click **Create Efficiency Monitor Connection** to create a connection to the efficiency monitor.
2. To create a communication channel to a connection, select the desired connection from the displayed pulldown menu of connections, and click **Ok**. The toolkit establishes a communication channel to the connection, creates a report folder for the connection, and creates a folder of tasks for the connection.

   Note: By default, the connection and the reports and tasks folders are named according to the hierarchy path of the connection. If you want, you can specify a different name for the connection and its folders.

Figure 122. Legacy EMIF Debug Toolkit - Create Memory Interface Connection



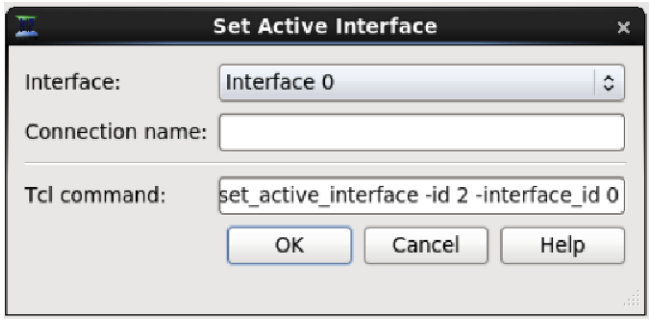
The above figure shows the first EMIF IP core from both columns in the pulldown menu of **Memory Interface**. Referring to the previous example in *Configuring Your EMIF IP for Use with the Debug Toolkit*, the DDR4 component interface is the first EMIF IP core in column 2 and the DDR4 UDIMM interface is the first EMIF IP core in column 3.

3. You can run any of the tasks in the folder for the connection; any resulting reports appear in the reports folder for the connection.

13.7.1.4.5. Selecting an Active Interface

If you have more than one external memory interface in an I/O column, you can select one instance as the active interface for debugging.

1. To select one of multiple EMIF instances in the same I/O column, select the active interface ID from the **Interface** pulldown menu in the **Set Active Interface** dialog box. This interface ID is the same ID that you have assigned to the given EMIF IP core in the **Calibration Debug Options** section of the **Diagnostics** tab.



Referring to the previous example in the *Configuring Your EMIF IP for Use With the Debug Toolkit* topic, interface 0 is associated with the first EMIF component interface, interface 1 is associated with the first HiLo interface, and interface 2 is associated with the first EMIF UDIMM interface.

2. If you want to generate reports for the new active interface, you must first recalibrate the interface.

13.7.1.5. Reports

The toolkit can generate a variety of reports, including summary, calibration, and margining reports for external memory interface connections. To generate a supported type of report for a connection, you run the associated task in the tasks folder for that connection.

Summary Report

The Summary Report provides an overview of the memory interface; it consists of the following tables:

- Summary table. Provides a high-level summary of calibration results. This table lists details about the connection, IP version, IP protocol, and basic calibration results, including calibration failures. This table also lists the estimated average read and write data valid windows, and the calibrated read and write latencies.
- Interface Details table. Provides details about the parameterization of the memory IP. This table allows you to verify that the parameters in use match the actual memory device in use.
- Ranks Masked from Calibration tables (DDR3 only). Lists any ranks that were masked from calibration when calibration occurred. Masked ranks are ignored during calibration.

Calibration Report

The Calibration Report provides detailed information about the margins observed during calibration, and the settings applied to the memory interface during calibration; it consists of the following tables:

- Calibration Status Per Group table: Lists the pass/fail status per group.
- DQ Pin Margins Observed During Calibration table: Lists the DQ read/write margins and calibrated delay settings. These are the expected margins after calibration, based on calibration data patterns. This table also contains DM/DBI margins, if applicable.
- DQS Pin Margins Observed During Calibration table: Lists the DQS margins observed during calibration.

- FIFO Settings table: Lists the VFIFO and LFIFO settings made during calibration.
- Latency Observed During Calibration table: Lists the calibrated read/write latency.
- Address/Command Margins Observed During Calibration table: Lists the margins on calibrated A/C pins, for protocols that support Address/Command calibration.

13.7.1.6. On-Die Termination Calibration

The **Calibrate Termination** feature lets you determine the optimal **On-Die Termination** and **Output Drive Strength** settings for your memory interface.

The **Calibrate Termination** function runs calibration with all available termination settings and selects the optimal settings based on the calibration margins.

The **Calibrate Termination** feature is available for DDR4, and RLDRAM 3 protocols.

13.7.1.7. Eye Diagram

The **Generate Eye Diagram** feature allows you to create read and write eye diagrams for each pin in your memory interface.

The **Generate Eye Diagram** feature uses calibration data patterns to determine margins at each V_{ref} setting on both the FPGA pins and the memory device pins. A full calibration is done for each Vref setting. Other settings, such as DQ delay chains, will change for each calibration. At the end of a `Generate Eye Diagram` command, a default calibration is run to restore original behavior

The **Generate Eye Diagram** feature is available for DDR4 and QDR-IV protocols.

13.7.1.8. Driver Margining for Intel Stratix 10 EMIF IP

The Driver Margining feature lets you measure margins on your memory interface using a driver with arbitrary traffic patterns.

Margins measured with this feature may differ from margins measured during calibration, because of different traffic patterns. Driver margining is not available if ECC is enabled.

To use driver margining, ensure that the following signals on the driver are connected to In-System Sources/Probes:

- Reset_n: An active low reset signal
- Pass: A signal which indicates that the driver test has completed successfully. No further memory transactions must be sent after this signal is asserted.
- Fail: A signal which indicates that the driver test has failed. No further memory transactions must be sent after this signal is asserted.
- PNF (Pass Not Fail): An array of signals that indicate the pass/fail status of individual bits of a data burst. The PNF should be arranged such that each bit index corresponds to `(Bit of burst * DQ width) + (DQ pin)`. A `1` indicates pass, `0` indicates fail. If the PNF width exceeds the capacity of one In-System Probe, specify them in `PNF[1]` and `PNF[2]`; otherwise, leave them blank.

If you are using the example design with a single EMIF, the In-System Sources/Probes can be enabled by adding the following line to your .qsf file:

```
set_global_assignment -name VERILOG_MACRO
"ALTERA_EMIF_ENABLE_ISSP=1"
```

13.7.1.8.1. Determining Margin

The Driver Margining feature lets you measure margins on your EMIF IP interface using a driver with arbitrary traffic patterns.

The Driver Margining feature is available only for DDR3 and DDR4 interfaces, when ECC is not enabled.

1. Establish a connection to the desired interface and ensure that it has calibrated successfully.
2. Select **Driver Margining** from the **Commands** folder under the target interface connection.
3. Select the appropriate **In-System Sources/Probes** using the drop-down menus.
4. If required, set additional options in the **Advanced Options** section:
   - Margining is performed on all ranks together.
   - **Step size** specifies the granularity of the driver margining process. Larger step sizes allow faster margining but reduced accuracy. It is recommended to omit this setting.
   - **Adjust delays after margining** causes delay settings to be adjusted to the center of the window based on driver margining results.
   - The **Margin Read**, **Write**, **Write DM**, and **DBI** checkboxes allow you to control which settings are tested during driver margining. You can uncheck boxes to allow driver margining to complete more quickly.
5. Click **OK** to run the tests.
   The toolkit measures margins for DQ read/write and DM. The process may take several minutes, depending on the margin size and the duration of the driver tests. The test results are available in the *Margin Report*.

13.7.1.9. Example Tcl Script for Running the Legacy EMIF Debug Toolkit

If you want, you can run the Legacy EMIF Debug Toolkit using a Tcl script. The following example Tcl script is applicable to all device families.

The following example Tcl script opens a file, runs the debug toolkit, and writes the resulting calibration reports to a file.

You should adjust the variables in the script to match your design. You can then run the script using the command `quartus_sh -t example.tcl`.

```
# Modify the following variables for your project
set project "ed_synth.qpf"
# Index of the programming cable.  Can be listed using "get_hardware_names"
set hardware_index 1
# Index of the device on the specified cable. Can be listed using
"get_device_names"
set device_index 1
# SOF file containing the EMIF to debug
set sof "ed_synth.sof"
# Connection ID of the EMIF debug interface.  Can be listed using
"get_connections"
set connection_id 2
```

```
# Output file
set report "toolkit.rpt"

# The following code opens a project and writes its calibration reports to a
file.
project_open $project
load_package ::quartus::external_memif_toolkit
initialize_connections
set hardware_name [lindex [get_hardware_names] $hardware_index]
set device_name [lindex [get_device_names -hardware_name $hardware_name]
$device_index]
link_project_to_device -device_name $device_name -hardware_name $hardware_name -
sof_file $sof
establish_connection -id $connection_id
create_connection_report -id $connection_id -report_type summary
create_connection_report -id $connection_id -report_type calib
write_connection_target_report -id $connection_id -file $report
```

13.7.1.10. Using the Legacy EMIF Debug Toolkit with Intel Stratix 10 HPS Interfaces

The Legacy External Memory Interface Debug Toolkit is not directly compatible with Intel Stratix 10 HPS interfaces.

To debug your Intel Stratix 10 HPS interface using the Legacy EMIF Debug Toolkit, you should create an identically parameterized, non-HPS version of your interface, and apply the toolkit to that interface. When you finish debugging this non-HPS interface, you can then apply any needed changes to your HPS interface, and continue your design development.

13.7.2. Debugging with the External Memory Interface Unified Calibration Debug Toolkit

The External Memory Interface Unified Calibration Debug Toolkit for Intel Stratix 10 FPGAs provides access to data collected by the Nios II sequencer during memory calibration, and provides analysis tools for evaluating the calibrated interface stability and assessing hardware conditions.

The toolkit provides the following reports:

- Interface and memory configuration, such as external memory protocol and interface width.
- Calibration results including status (pass/fail), failure stage (if applicable), delay settings and margins, V_{REF} settings and margins.

The available tasks and analysis tools enable the following:

- Requesting recalibration of the memory interface.
- Reading the probe data or writing the source data to the In-System Sources and Probes (ISSPs) instances in the design.
- Viewing the delay setting on any pin in the selected interface and updating it if necessary.
- Rerunning the traffic generator in the example design.
- Running V_{REF} Margining on the interface.
- Running Driver Margining on the interface.
- Calibrating or update the termination settings.

13.7.2.1. Prerequisites for Using the EMIF Unified Calibration Debug Toolkit

Before using the EMIF Unified Calibration Debug Toolkit, you must complete the following steps.

1. Using an EMIF IP version of 19.2.1 or higher, configure your design to use the EMIF Unified Calibration Debug Toolkit (refer to the following topics for details).
2. Compile your design.
3. Program your target device with the resulting SRAM object file (`.sof`).

13.7.2.2. Configuring a Design to use the EMIF Unified Calibration Debug Toolkit

You can use the Unified Calibration Debug Toolkit in a variety of scenarios.

- You are generating a new design and want to enable the toolkit with it.
- You have a design with the toolkit enabled, and you want to add more interfaces to it.
- You have an existing design and want to enable the toolkit with it.

13.7.2.2.1. Generating a Design Example with the Debug Toolkit

To enable the Debug Toolkit in the example design, open the parameter editor for your EMIF IP, and complete the following steps.

1. Open the parameter editor by selecting **External Memory Interface Intel Stratix 10 FPGA IP** from the **IP Catalog**.
2. Open the **Diagnostics** tab of the parameter editor.
3. Select **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port ➤ Add EMIF Debug Interface**.
4. Select **Enable In-System Sources and Probes**.
5. Parameterize the interface to your requirements,
6. Click **Generate Example Design**.

The system configures the generated design with the Debug Toolkit enabled and all components connected as required for a single interface.

13.7.2.2.2. Adding Interfaces to an Design Example

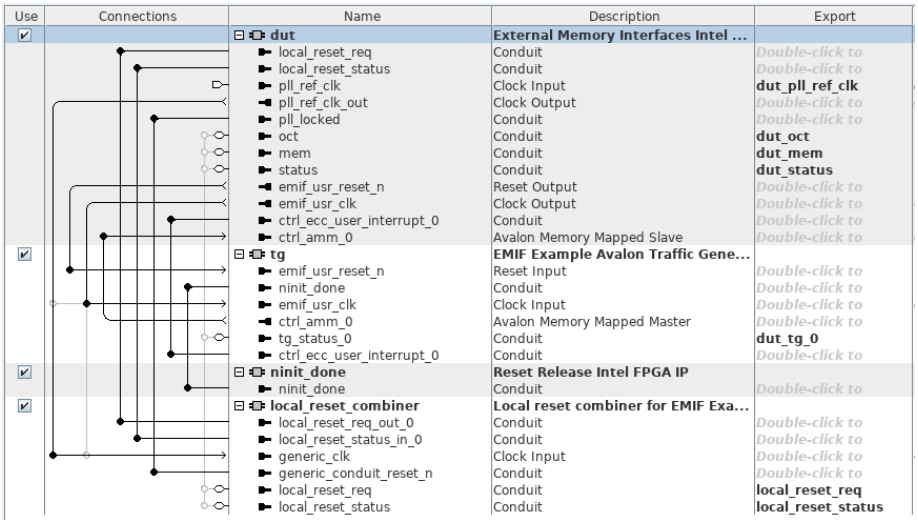
To create a design containing two or more interfaces, complete the following steps.

1. Repeat the steps of the *Generating a Design Example with the Debug Toolkit* procedure for as many interfaces as required in your design.
2. Choose one of the generated design examples to add all the other interfaces to. This integrated design is used in the final project.
3. Each of the generated designs contains an `/ip` directory, in which several files having `.ip` extension reside. For each additional interface, move the following files into the final project's `/ip` directory, and rename them to distinguish them from other existing files in that directory:
   a. `ed_synth_s10_0.ip`

b. `ed_synth_tg.ip`

c. `ed_synth_ninit_done.ip`

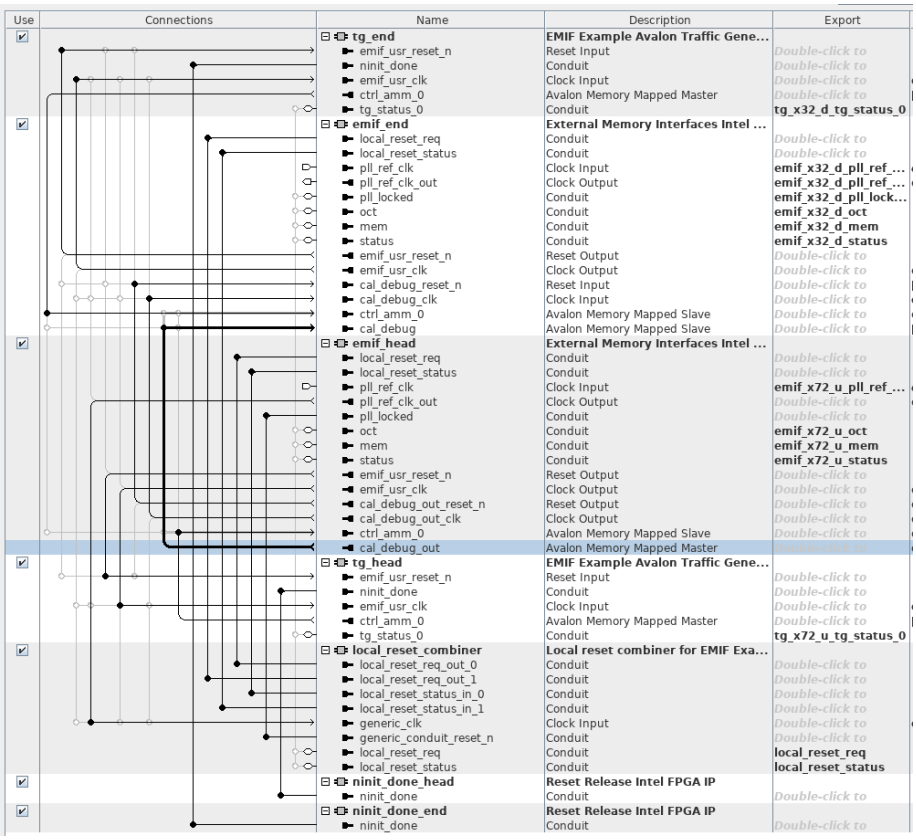
Figure 123. Existing Connections Before Adding a Second Design Example



4. Open the `ed_synth.qsys` file of the final project in the Platform Designer.
5. The IP variants that you have copied into the final project directory should now appear under the *Existing IP Variants* section of the **IP Catalog**. Add these IP variants to your system and connect them to each other, using the original design example as reference.
6. If no two EMIF IPs in your design share an I/O column, skip ahead to step 8; otherwise, proceed as follows:
   a. One EMIF IP in each column used must be at the head of the daisy-chain; to do this, set the following parameters in this EMIF IP:
      - Set **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** to **Add EMIF Debug Interface**.
      - Select **Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**.
   b. One EMIF IP in each column used must be at the end of the daisy-chain; to do this, set the following parameters in this EMIF IP:
      - Set **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** to **Export**.
      - Deselect **Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**.
   c. Configure every additional EMIF IP as follows:
      - Set **Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** to **Export**.
      - Select **Enable Daisy-Chaining for Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port**.

7. Connect any daisy-chained EMIF IPs in your design. To do this, make connections from a `cal_debug_out` interface on one EMIF to the `cal_debug` interface on another EMIF.

Figure 124. Connections for a Second EMIF Design Example



8. Ensure you have ported any pin assignments over to the final design.
9. After the system is connected, you can generate HDL and compile your design.

13.7.2.2.3. Enabling the EMIF Unified Calibration Debug Toolkit in an Existing Design

To enable Unified Toolkit support in an existing design, follow these steps.

1. Add the following line to the `.qsf` file: `set_global_assignment -name VERILOG_MACRO "ALTERA_EMIF_ENABLE_ISSP=1"`
2. For each EMIF instance in the design, select **Add EMIF Debug Interface** from the **Intel Quartus Prime EMIF Debug Toolkit/On-Chip Debug Port** drop-down menu.
3. Click **Generate HDL** and compile the design.

Note: If your original EMIF IP version is older than 19.2.1, or if your design was not generated based on the design example, you must regenerate the design beginning with a design example. Follow the instructions in Generating a Design Example with the Debug Toolkit and in Adding Interfaces to an Design Example.

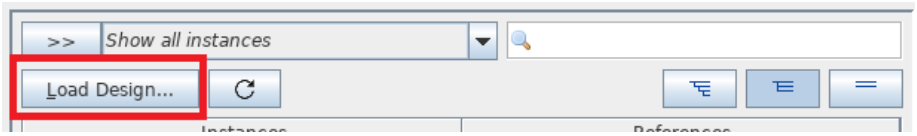
13.7.2.3. Launching the EMIF Debug Toolkit

Before launching the EMIF Debug Toolkit, ensure that you have configured your device with a programming file that has the EMIF Debug Toolkit enabled, as described in *Configuring the EMIF IP and Calibration IP for Use with the Debug Toolkit*.

To launch the EMIF Debug Toolkit, follow these steps:

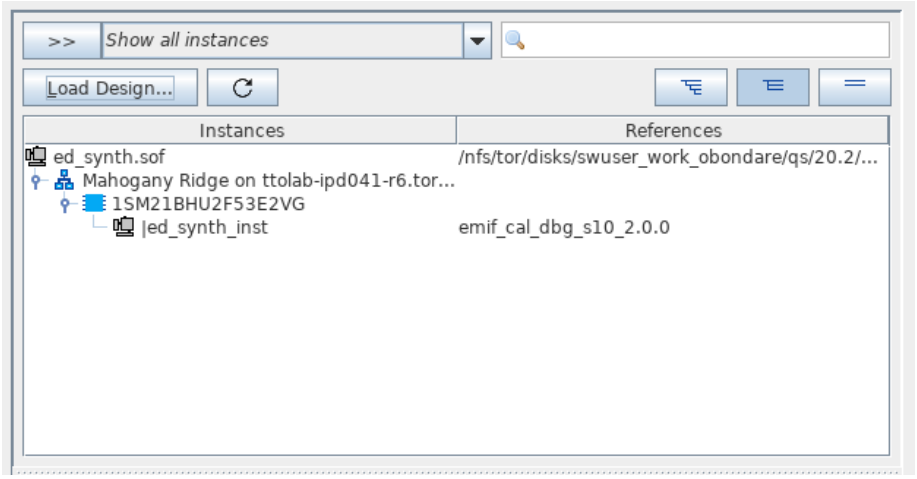
1. In the Intel Quartus Prime software, open the System Console by clicking **Tools ➤ System Debugging Tools ➤ System Console**.
2. In the System Console, load the `.sof` file with which you programmed the board in *Prerequisites for Using the EMIF Unified Calibration Debug Toolkit*.

Figure 125.



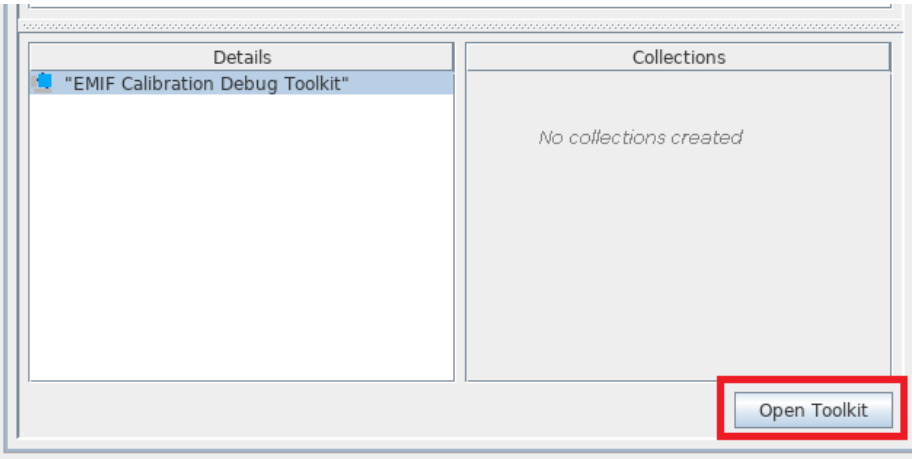
3. Select the correct instances to debug.

Figure 126.



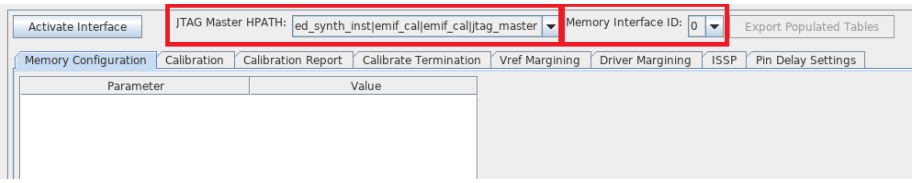
4. Open the toolkit.

Figure 127.



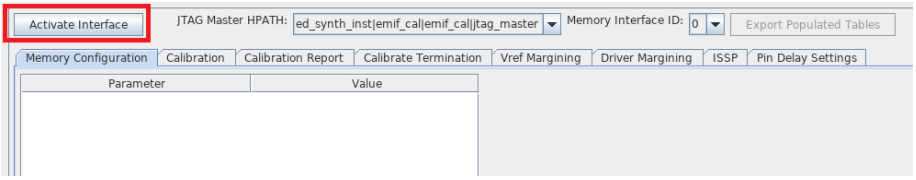
5. If there are multiple EMIF instances in the programmed design, select the column (path to JTAG master) and ID of the EMIF instance for which to activate the toolkit.

Figure 128.



6. Click *Activate Interface* to allow the toolkit to read the parameters and status of the selected interface, and to run analysis tasks.

Figure 129.



7. You must debug the interfaces one at a time; therefore, when trying to connect to another EMIF in the design, you must deactivate the current interface.

Figure 130.

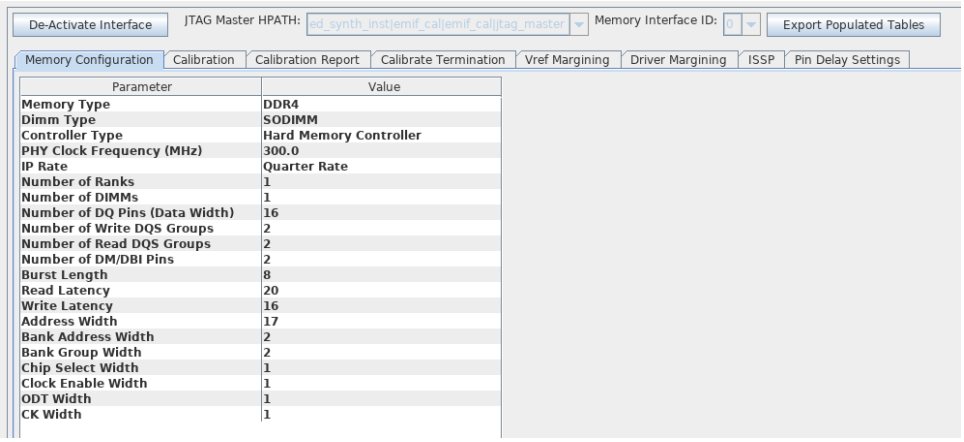
13.7.2.4. Using the EMIF Debug Toolkit

The main view of the EMIF Debug Toolkit contains several tabs: **Memory Configuration**, **Calibration**, **Calibration Report**, **Calibrate Termination**, **Vref Margining**, **Driver Margining**, **ISSP**, and **Pin Delay Settings**.

13.7.2.4.1. Memory Configuration Tab

The **Memory Configuration** tab shows the IP settings, which you defined when you parameterized the EMIF IP.

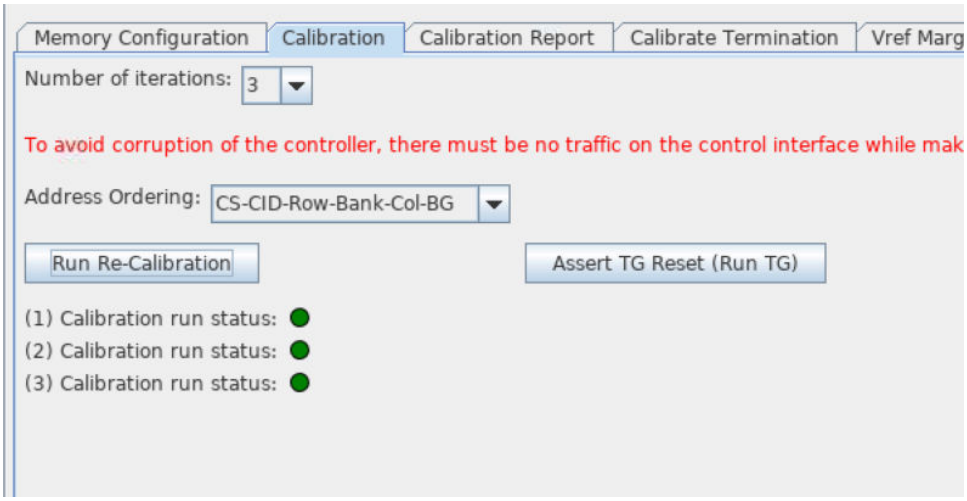
Figure 131. Memory Configuration Tab



13.7.2.4.2. Calibration Tab

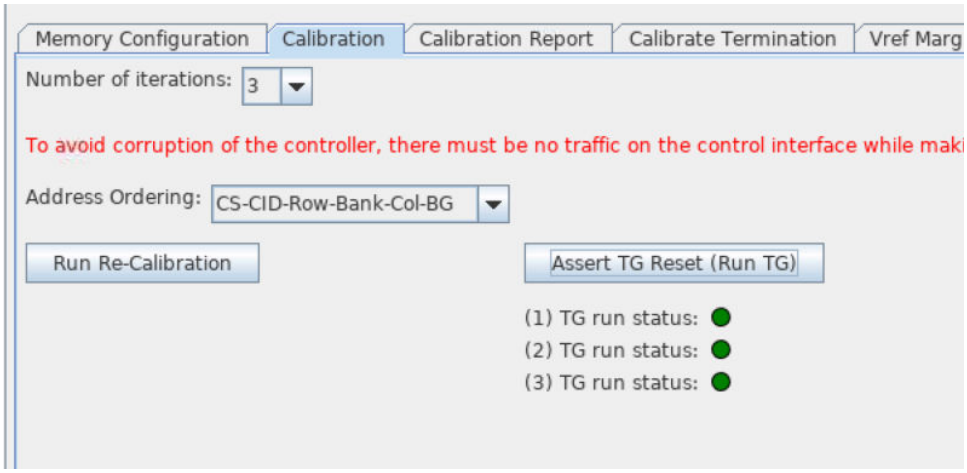
To re-run calibration, you can select the desired number of iterations from the **Number of iterations** pull-down menu, and then click on the **Run Re-Calibration** button.

Figure 132. Calibration Tab



To re-run the generated traffic, you can select the desired number of iterations from the **Number of iterations** pull-down menu, and then click on the **Assert TG Reset (Run TG)** button.

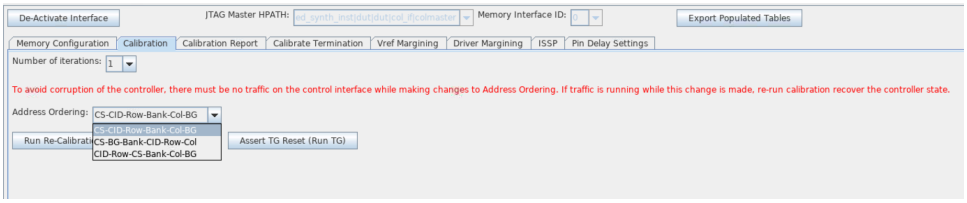
Figure 133. TG Run Status Indicator LEDs



Each run's status appears as an LED. A green LED indicates a pass, while a red LED indicates a fail.

To change address ordering, you can select one of the options from the **Address Ordering** drop-down menu.

Figure 134.



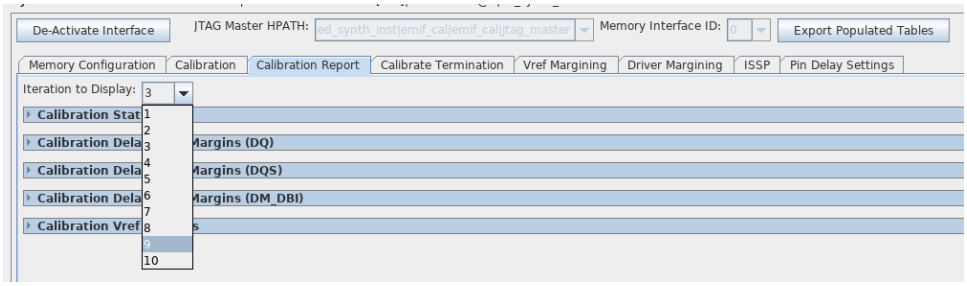
13.7.2.4.3. Calibration Report Tab

The **Calibration Report** tab shows the calibration status, as well as delay settings and margins discovered during calibration.

Choose the Iteration to View

You may choose to view the status, delay settings, or margins reports for any of the most recent calibration iterations (which were initiated through the toolkit). To choose the iteration to view, select from the **Iteration to Display** dropdown.

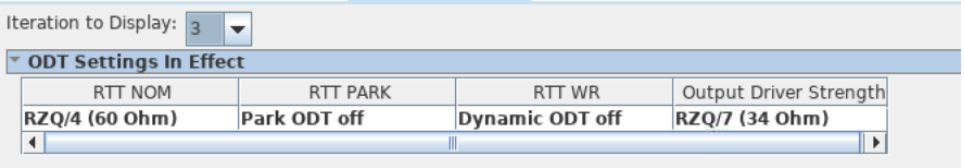
Figure 135. Selecting the Iteration to Display



ODT Settings in Effect

This report shows the current ODT settings for the latest calibration.

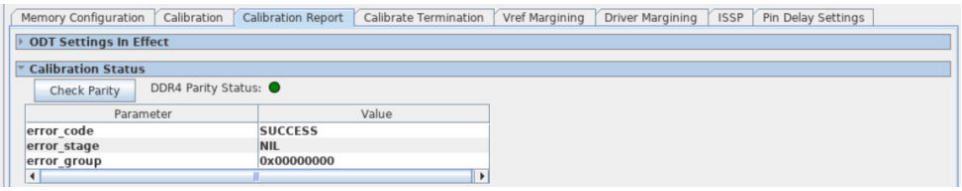
Figure 136. ODT Settings in Effect



Calibration Status Report

The **Calibration Status** report shows the calibration status and memory parity (ALERT_N) status. If a failure occurs, this report shows the first stage of calibration that failed, as well as which data groups failed this stage. Memory parity status observed during calibration is shown for DDR4 interfaces if you have enabled ISSPs in the design. The calibration status report window includes a memory parity status LED and a button that allows you to reread the memory parity status.

Figure 137. Calibration Status



Calibration Delays and Margins Reports

The **Calibration Delays** and **Margins** reports provide detailed information about the margins observed during calibration, and the settings applied on the calibration bus during calibration. To view the margins, click on the respective section for DQ, DQS, DM_DBI, V_{REF} or Address and Command.

Figure 138. DQ Calibration Delays and Margins

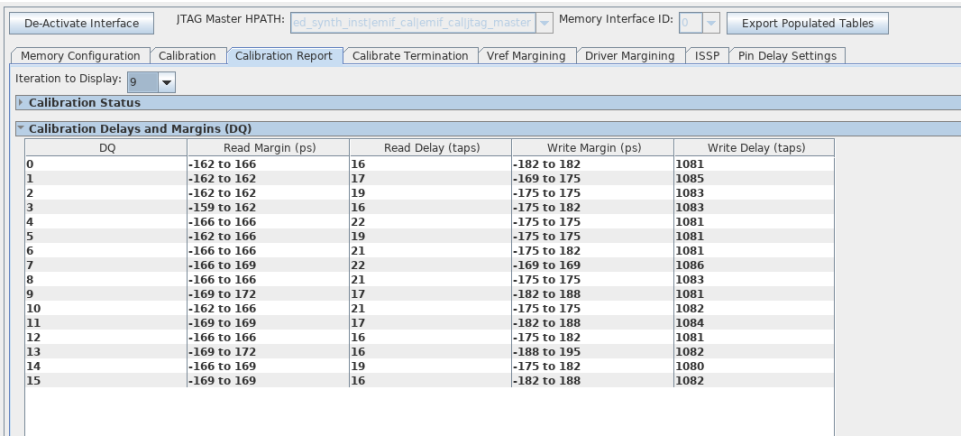


Figure 139. DQS Calibration Delays and Margins

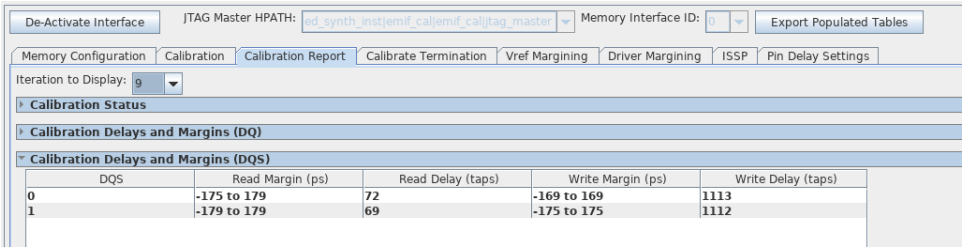


Figure 140. DM_DBI Calibration Delays and Margins

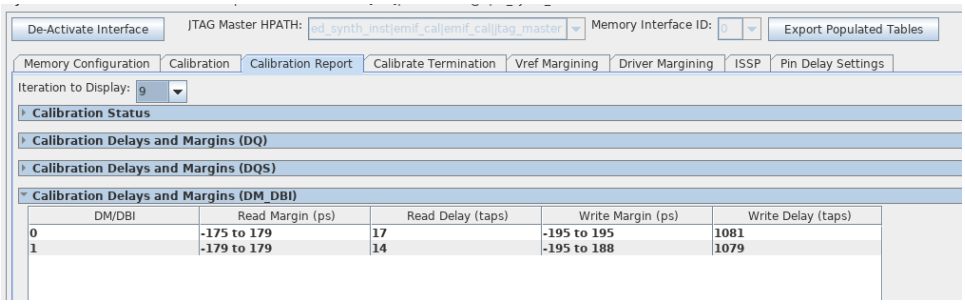


Figure 141. V_{REF} Calibration Settings

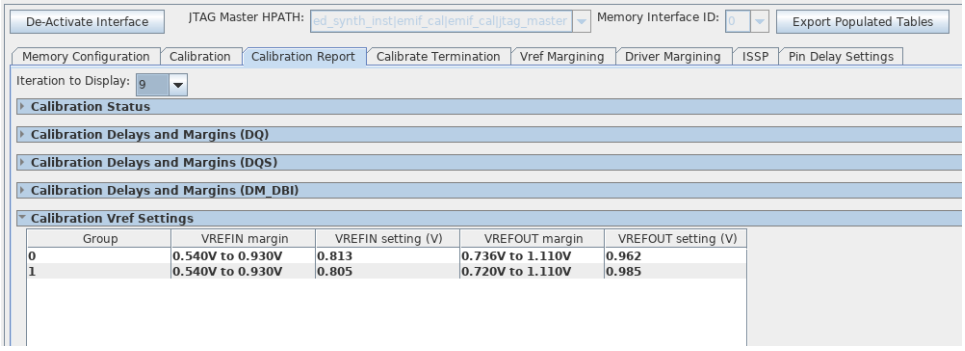
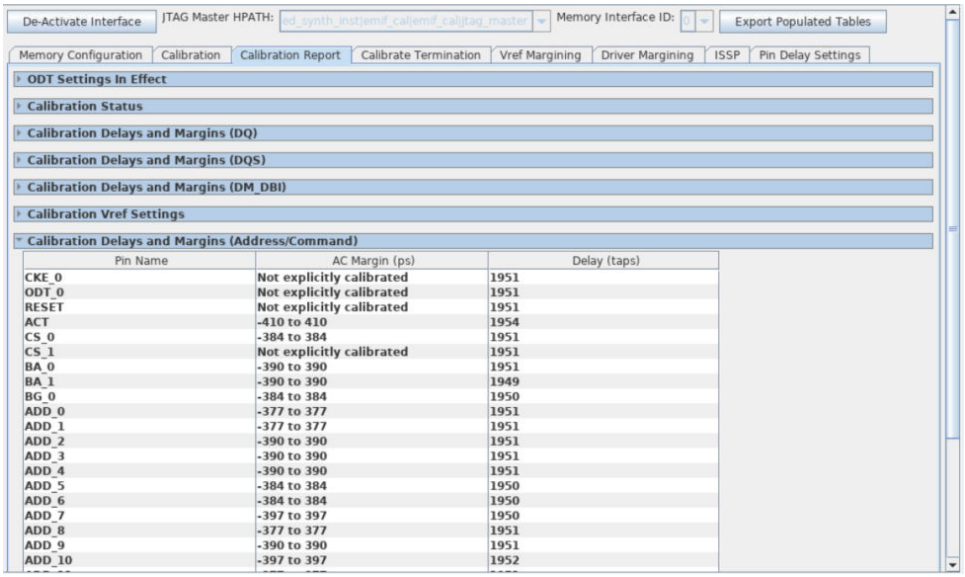


Figure 142. Address and Command Calibration Delays and Margins



Note: Reports can also be viewed graphically; for information, refer to *Viewing Diagrams in Eye Viewer*.

13.7.2.4.4. Calibrate Termination Tab

The **Calibrate Termination** tab allows you to update any of the following termination settings without having to recompile or reprogram the design: **RTT_NOM**, **RTT_PARK**, **RTT_WR**, **Output Drive Strength**.

The **Calibrate ODT** feature also lets you determine the optimal **On-Die Termination** and **Output Drive Strength** settings for your memory interface.

- Press **Calibrate ODT** to run calibration with the cross product of the termination settings that you want to sweep, and display the worst-case-margin for each combination of settings.
- If you select **Run TG for each combination of settings**, the traffic generator status is also displayed.

You can review the report and use it to apply the optimal termination settings (generally the ones with the largest margin), using the drop-downs for each setting.

To reset the ODT settings to those from the IP parameterization, press **Restore ODT settings from IP parameterization**.

Figure 143. Termination Settings

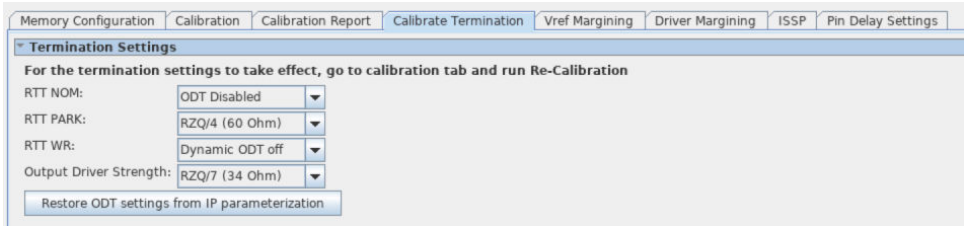
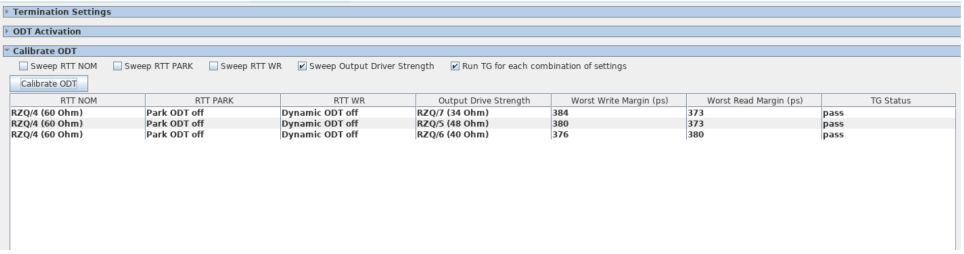
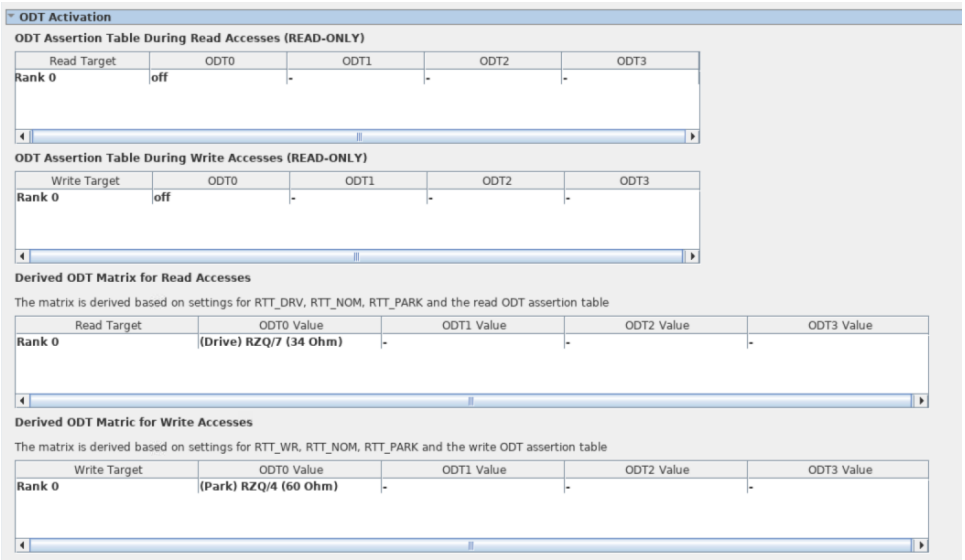


Figure 144. Calibrate ODT



The **ODT Activation** section displays the ODT assertion patterns in use and the ODT settings in effect during read and write.

Figure 145. ODT Assertion Table



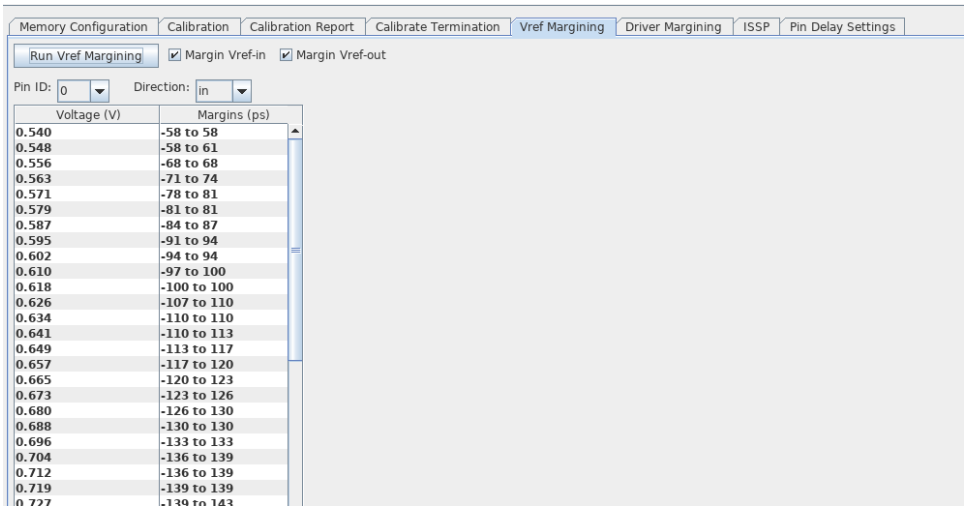
Note: The **Calibrate Termination** tab does not support LRDIMM interfaces.

13.7.2.4.5. Vref Margining Tab

The V_{REF} Margining feature sweeps different Vref-in and Vref-out settings. At each V_{REF} value, calibration finds the margin on each pin.

You can choose to apply this margining tool to both or only one of the directions (V_{REF}-in or V_{REF}-out) using the checkboxes near the **Run Vref Margining** button. The tool reports the passing delay margins for each pin, at each V_{REF} value, for each direction. The Pin ID corresponds to the DQ index on the interface (for example, Pin ID=0 refers to DQ0 on the memory interface).

Figure 146. V_{REF} Margining



Note: Reports can also be viewed graphically; for information, refer to *View Diagrams in Eye Viewer*.

13.7.2.4.6. Driver Margining Tab

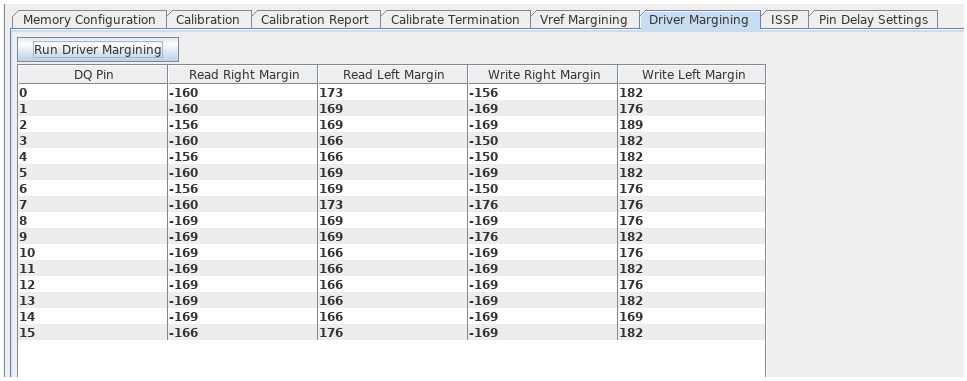
The Driver Margining feature lets you measure margins on your memory interface using a driver with predefined traffic patterns. Margins measured with this feature are expected to be smaller than those measured during calibration, because this traffic pattern is longer than the ones run during calibration, and is intended to stress the interface.

Driver Margining is supported only when a memory interface meets all of the following criteria:

- Has a TG IP (altera_emif_tg_avl) connected to it (that is, not the configurable traffic generator).
- Does not have ECC enabled.
- Has ISSPs enabled in the project's `.qsf` file.

To run driver margining, click the **Run Driver Margining** button at the top left corner of the tab. The toolkit then measures margins for DQ read, DQ write, and DM. The process usually takes a few minutes, depending on the margin size, the interface size, and the duration of the driver tests. The test results are displayed in the table when the test is completed.

Figure 147. Driver Margining Tab



DQ Pin	Read Right Margin	Read Left Margin	Write Right Margin	Write Left Margin
0	-160	173	-156	182
1	-160	169	-169	176
2	-156	169	-169	189
3	-160	166	-150	182
4	-156	166	-150	182
5	-160	169	-169	182
6	-156	169	-150	176
7	-160	173	-176	176
8	-169	169	-169	176
9	-169	169	-176	182
10	-169	166	-169	176
11	-169	166	-169	182
12	-169	166	-169	176
13	-169	166	-169	182
14	-169	166	-169	169
15	-166	176	-169	182

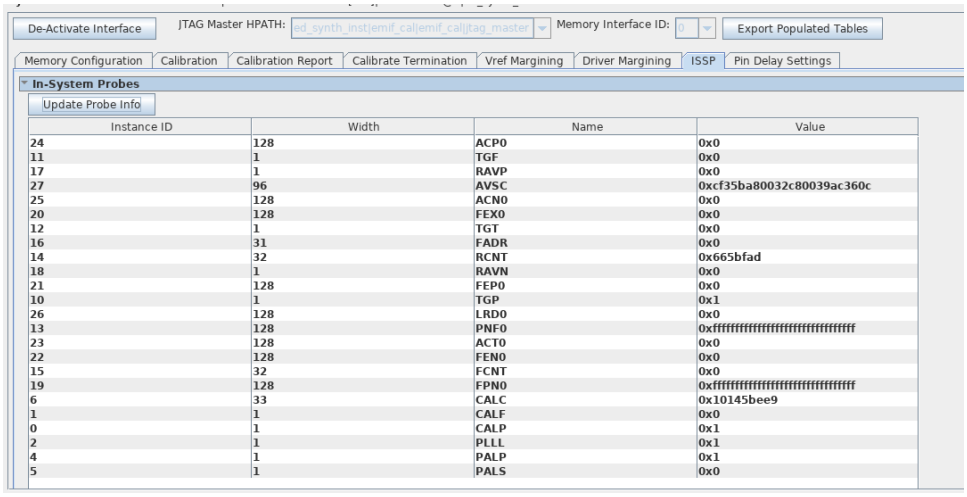
Note: Reports can also be viewed graphically; for information, refer to *Viewing Diagrams in Eye Viewer*.

13.7.2.4.7. ISSP Tab

The **ISSP** tab allows you to read probe data and set source values for the In-System Sources and Probes in the design.

To reread probe data from the ISSPs in the design, expand the **In-System Probes** section and click the **Update Probe Info** button.

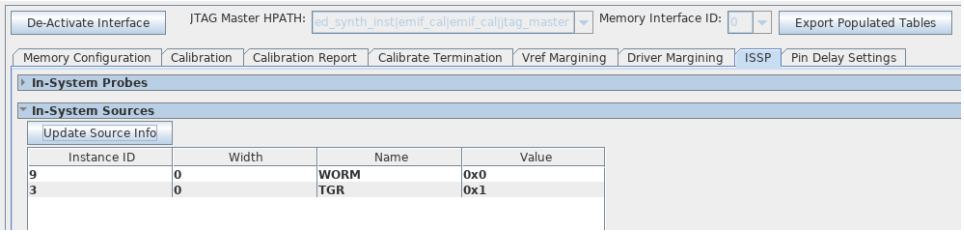
Figure 148. In-System Probes



Instance ID	Width	Name	Value
24	128	ACP0	0x0
11	1	TGF	0x0
17	1	RAVP	0x0
27	96	AVSC	0xcf35ba80032c80039ac360c
25	128	ACN0	0x0
20	128	FEX0	0x0
12	1	TGT	0x0
16	31	FADR	0x0
14	32	RCNT	0x665bfad
18	1	RAVN	0x0
21	128	FEP0	0x0
10	1	TGP	0x1
26	128	LRD0	0x0
13	128	PNF0	0xffffffffffffffffffffffffffffffff
23	128	ACT0	0x0
22	128	FEN0	0x0
15	32	FCNT	0x0
19	128	FPN0	0xffffffffffffffffffffffffffffffff
6	33	CALC	0x10145bee9
1	1	CALF	0x0
0	1	CALP	0x1
2	1	PLLL	0x1
4	1	PALP	0x1
5	1	PALS	0x0

To reread source data from the ISSPs in the design, expand the **In-System Sources** section and click the **Update Source Info** button.

Figure 149. In-System Sources



To overwrite the source data, select the *Instance Name* and change the *Writedata* value. Click the **Write Source Data** button to write the new source value.

Table 352. Definition of ISSPs in the EMIF Design Example

Instance Name	Description
PLLL	PLL Lock signal. A value of 1 means that the PLL is locked, a value of 0 means that the PLL cannot lock to the reference clock.
RCNT	Total read data count.
FCNT	Total fail count (data mismatch count).
FADR	First address where a data mismatch is reported.
RAVP	Read data valid from the data before the first failing address.
RAVN	Read data valid from the data after the first failing address.
PNF#	Persistent Pass Not Fail Flag. A 1 indicates pass, 0 indicates fail.
FPN#	PNF flag for the first data mismatch.
FEX#	The expected read data for the first failing read.
FEP#	The expected read data before the first failing read.
FEN#	The expected read data after the first failing read.
ACT#	The actual read data for the first failing read.
ACP#	The actual read data before the first failing read.
ACN#	The actual read data after the first failing read.
LRD#	The repeated read result. When there is an error, the driver reads again from the first failing address. This is the PNF flag for the repeated read.
AVSC	Avalon Stall Count - a concatenation of the following three 32-bit signals (MSB to LSB): • Count of read/write requests on the ctrl_amm interface. • Count of only read requests on the ctrl_amm interface. • Number of clocks counted since receiving the first read/write request.
PALP	Clock phase alignment lock status.
PALS	Clock phase alignment lock (secondary).
CALC	Calibration counter. Highest bit is a *done* signal — a value of 1 means that calibration has completed, and a value of 0 means that calibration is still in progress. The other 32 bits are a clock counter which tracks the number of clocks passed during calibration.
TGP	Traffic Generator Pass Flag. Pass=1.
TGF	Traffic Generator Fail Flag. Fail=1.

continued...

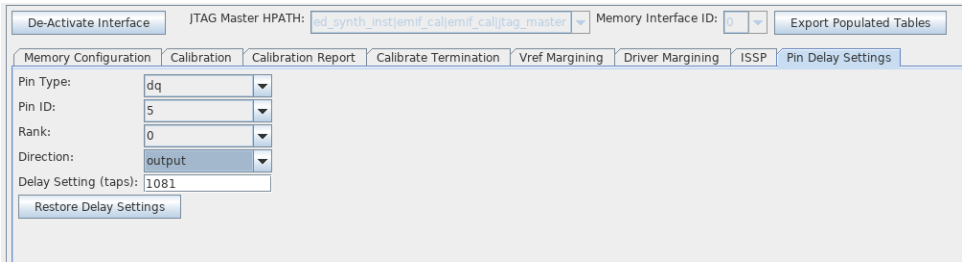
Instance Name	Description
TGT	Traffic Generator Timeout. Timeout=1.
TGR	Traffic Generator Reset. Active High. Toggle TGR to rerun the traffic generator.
RSTN	Global Reset for the design example. Active Low. Toggle RSTN to reset and recalibrate the interface.
WORM	Set to 1 to enable WORM mode. In WORM mode, if a data mismatch is encountered, the system stops as much of the traffic as possible and issues a read to the same address. In this mode, the persistent PNF is no longer meaningful as execution stops at the first data mismatch. By default, WORM mode is turned off.
PRTY	Indicates DDR4 memory parity status. A value of 0 indicates no error, and a value of 1 indicates an error. The value is not updated if the design is not DDR4, or if the **AC Parity Latency** parameter is disabled in the interface.

13.7.2.4.8. Pin Delay Settings Tab

The **Pin Delay Settings** tab lets you view and change delay values on specific pins.

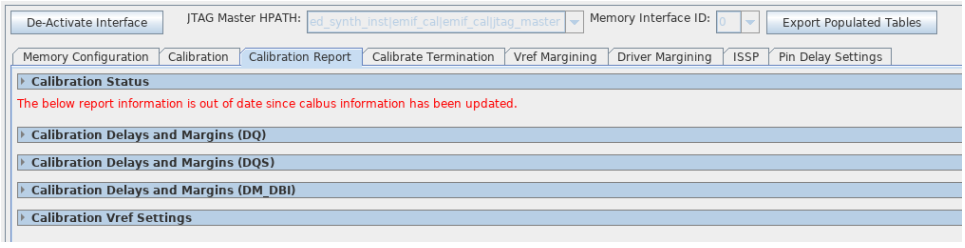
You can select the **Pin Type**, **Pin ID**, **Rank**, or **Direction** values of a delay that you are interested in, and the toolkit displays the delay value in the **Delay Setting (taps)** field.

Figure 150. Pin Delay Settings Tab



If you make changes to **Delay Setting (taps)** tab, the delay value is updated in hardware. After the value changes in hardware and no longer matches the setting found in calibration, a warning message appears, indicating that the margins in the **Calibration** tab are out-of-date.

Figure 151. Error Message Resulting from Changed Delay Settings



To restore the delay settings to the values found during the most recent calibration, click the **Restore Delay Settings** button on the **Pin Delay Settings** tab.

13.7.2.5. Exporting Tables

The **Export Populated Tables** button saves all the populated tables in the current toolkit view and all the pin delay settings read from the calbus bridge into `*.csv` files.

Figure 152. Export Populated Tables Button

All the `.csv` files are saved to the directory from where you originally launched the System Console.

Table 353. List of .csv Files

Name of .csv File	Corresponding Tab (Table) in the Toolkit GUI
`mem_config_table.csv`	Memory Configuration
`vref_margins_table.csv`	V_{REF} Margining
`rtt_nom_table.csv`	Calibrate Termination [RTT_NOM Margins]
`rtt_park_table.csv`	Calibrate Termination [RTT_PARK Margins]
`rtt_wr_table.csv`	Calibrate Termination [RTT_WR Margins]
`ods_table.csv`	Calibrate Termination [Output Driver Impedance Control Margins]
`cal_status_table.csv`	Calibration Report [Calibration Status]
`cal_results_delay_dq_table.csv`	Calibration Report [Calibration Delays and Margins (DQ)]
`cal_results_delay_dqs_table.csv`	Calibration Report [Calibration Delays and Margins (DQS)]
`cal_results_delay_dm_dbi_table.csv`	Calibration Report [Calibration Delays and Margins (DM_DBI)]
`cal_results_vref_table.csv`	Calibration Report [Calibration V_{REF} Settings]
`driver_margins_table.csv`	Driver Margining
`issp_probes_table.csv`	ISSP [In-System Probes]
`issp_sources_table.csv`	ISSP [In-System Sources]
`pin_delay.csv`	Pin Delay Settings

13.7.2.6. Viewing Diagrams in the Eye Viewer

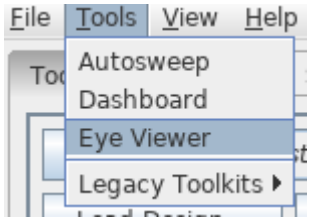
Some of the tables displayed in the toolkit can be viewed in a graphical format.

The following tabs and reports can be displayed:

- Calibration report: Report types are DQ, DQS, or DM/DBI, and split between input and output delays.
- Vref margining: Report types are Per-DQ, Merged for each DQS group, or Merged (an average across all the DQ margins), and split between input and output margins.
- Driver margining: Reports either input or output margins on each DQ pin.

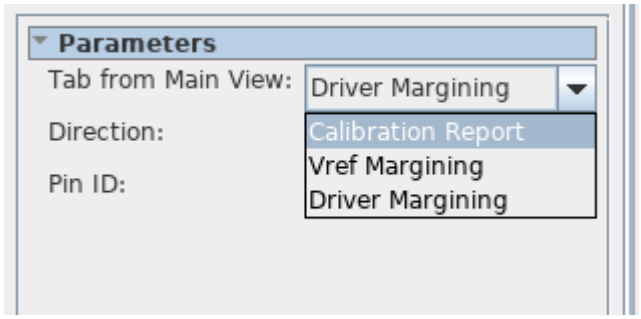
To view these reports, follow these steps:

1. Select **Tools ➤ Eye Viewer**.

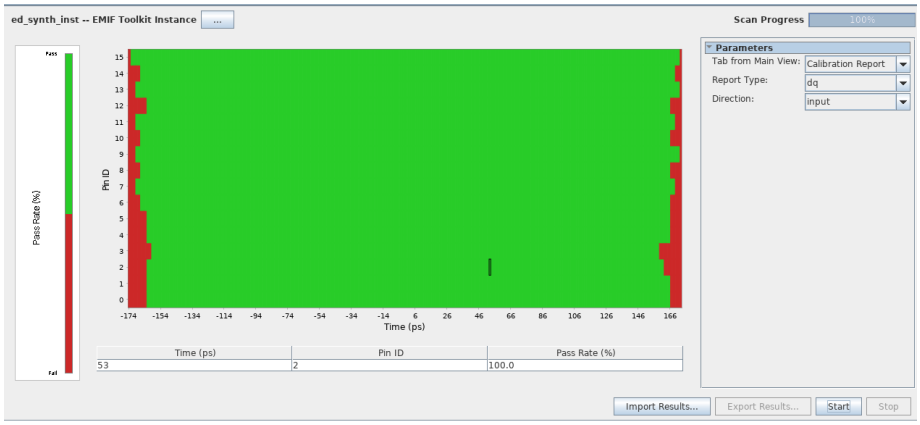


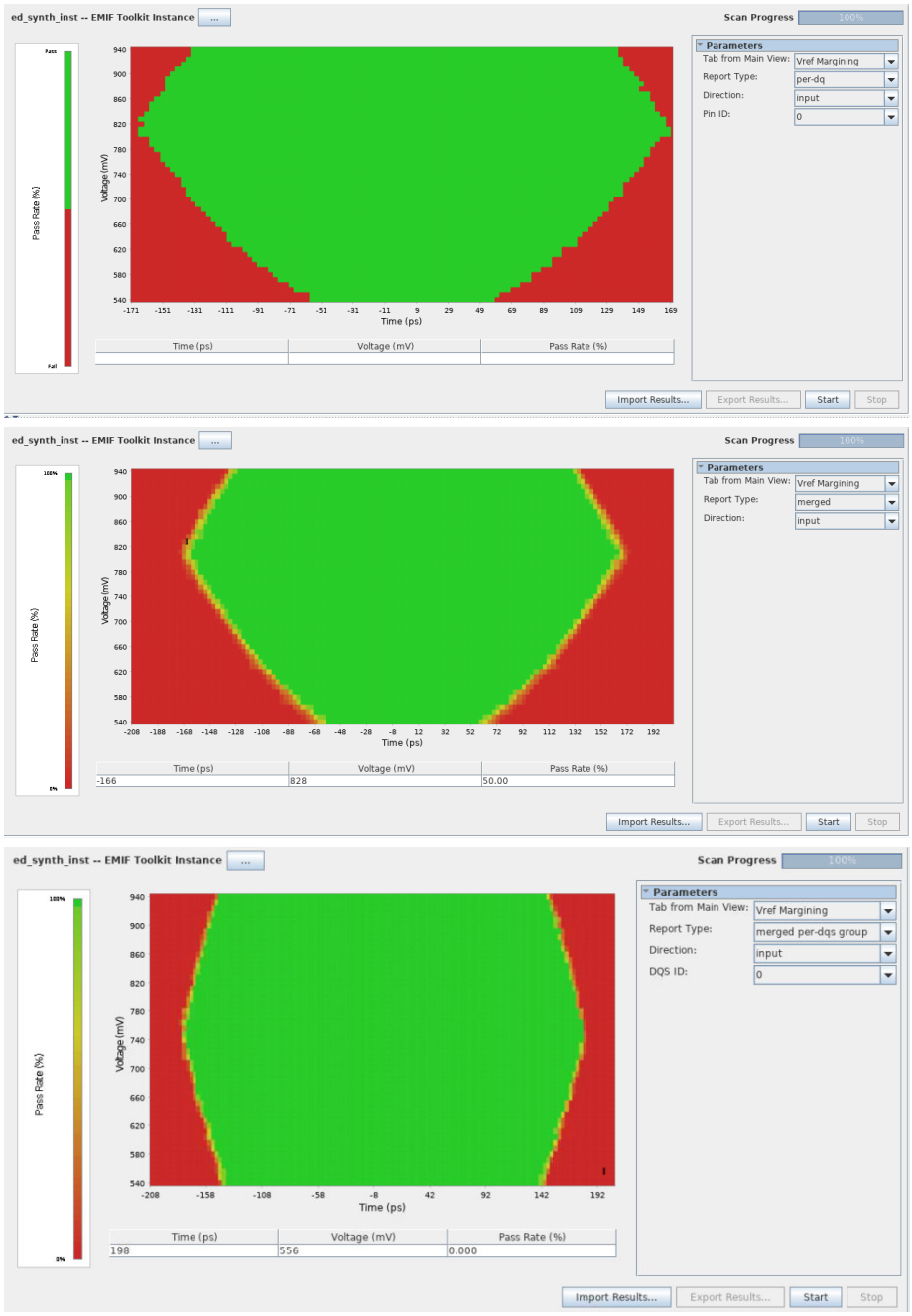
2. Select the desired toolkit instance.

3. In the **Parameters** section, select the desired tab from the drop-down menu.



4. Click the **Start** button at the bottom right, to draw the report based on the selected parameters. (The following diagrams demonstrate how different reports may look, and are taken from various designs.)





13.7.2.7. Guidelines for Debugging Calibration Issues

The following topics provide general guidelines for debugging calibration-related issues for DDR4 and QDR-IV interfaces.

General Hardware Debugging for Calibration Issues

1. Begin with the design example generated by the Intel Quartus Prime software as a starting point to debug your issue. Review and update the memory timing parameter, CAS, and Write CAS latency based on the speed bin of the targeted memory component and the operating frequency of your interface. Incorrect memory timing parameter, CAS, or Write CAS latency can cause data corruption in the memory component.
2. Verify that the design has the correct pin locations and I/O standard. Although the Fitter may place some unassigned pins automatically, you should provide the pin location assignments and I/O standard for all the EMIF pins in your design. Check the Fitter report to ensure that all the pins are placed correctly in the design.
3. Ensure that the PCB has correct termination resistors on the address and command signals. Refer to the *Board Design Guidelines* section for your memory protocol in this user guide for more information on suggested termination values.
4. Each EMIF instance has its own RZQ pin. Ensure that every RZQ pin on the FPGA side is connected to GND through a 240 ohm, 1% resistor.
5. If you are using discrete memory components, ensure that every ZQ pin on the memory component side is connected to GND through a 240 ohm, 1 % resistor.
6. Ensure that the EMIF IP is instantiated with the correct I/O PLL reference clock frequency and I/O standard. The reference clock must be stable and running at the expected frequency during calibration, after calibration, and during user mode. Probe the memory clock frequency to confirm that the memory clock is toggling at the expected frequency after configuring the device.
7. Check the relevant voltage rails for absolute value and for worst case noise. Suggested rails are V_{CC},V_{CCP}, V_{CCIO_PIO}, V_{CCPT},V_{CCA_PLL}, V_{REF}, V_{TT} and the power supplies at the memory device.
8. Ensure that the reset signal to the IP is driven correctly. The reset request is sent by transitioning the local_reset_req signal from low to high, then keeping the signal at the high state for a minimum of 2 EMIF core clock cycles, then transitioning the signal from high to low.
9. Check to determine whether the calibration problem exists on more than one board.
10. Determine whether the issue exists at lower interface frequencies. If the board passes at lower frequencies, evaluate the I/O Timing to ensure that the PCB and associated system is capable of running at your targeted frequency.
11. Repeat the calibration multiple times without reconfiguring the device, to see whether the calibration can recover by recalibrating the interface.
12. Rerun the calibration by reconfiguring the device to see whether the calibration can recover after reconfiguring the device.

13.7.2.7.1. Debugging Calibration Failure Using Information from the Calibration report

The following topics provide recommendations for debugging calibration failure after using the Debug Toolkit to determine which stage of calibration is failing.

You should complete the steps in the *General Hardware Debugging for Calibration Issues* section before proceeding with these recommendations.

13.7.2.7.2. Debugging Address and Command Leveling Calibration Failure

1. In each rank, verify that CS#, CAS#/A15, and DQS/DQSn are connected correctly from the FPGA to the memory device.
   - In a non-clamshell configuration, the algorithm only checks if the DQS0/DQS0n in each rank are toggling.
   - In a clamshell configuration, the algorithm checks if all the DQS/DQSn are toggling.
2. Try nondefault I/O settings for address and command and memory clock. Perform board simulation with IBIS models to determine the best settings for your design.

13.7.2.7.3. Debugging Address and Command Deskew Failure

1. Determine which pins are failing. And then:
   - If only some pins are failing, determine whether there is a connectivity problem on the failing net. Also check whether the failing net has the proper termination to V_{TT}. Refer to the *Board Design Guidelines* section for your memory protocol in this user guide for recommended termination and decoupling requirements.
   - If all the pins are failing, verify connectivity on the PAR pin and the ALERT# pin. All the address and command pins fail this calibration stage if the memory device is not receiving the PARITY bit, or if the FPGA is not receiving the ALERT# signal from the memory device, or if the FPGA is not receiving the ALERT# signal from the memory device. Verify that the ALERT# signal is pulled up to 1.2V.
2. Verify whether the memory clock is toggling at the correct frequency.
3. Verify that the memory device is powered.
4. Try with nondefault I/O settings for address and command and memory clock. Perform board simulation with IBIS models to determine the best settings for your design.

13.7.2.7.4. Debugging DQS Enable Failure

1. Verify that the correct resistor is connected between the RZQ pin of the FPGA and GND.
2. Verify that the correct resistor is connected between the ZQ pin of the memory component and GND.
3. Verify that there is no connectivity problem preventing the memory component from receiving the back-to-back READ commands correctly.
4. Verify that there is no connectivity problem preventing the DQS/DQSn pins on the FPGA from receiving the DQS/DQSn signals correctly.
5. Verify that the address and command pins are correctly connected between the FPGA and the memory device or DIMM. (Note that passing the address and command leveling and deskew does not necessarily mean that these signals are connected properly (i.e no swap of signals).

13.7.2.7.5. Debugging Read Deskew Calibration Failure

1. Ensure that you specify the correct memory timing parameter, CAS, and Write CAS latency when generating the EMIF IP. An incorrect parameter value can cause data corruption.
2. Determine which pins are failing.
   - If only certain DQ pins are failing, verify that there is no connectivity problem on the PCB.
   - If the same set of pins are failing on multiple PCBs, check for a possible problem with the board layout—for example, cross talk.
3. Create design with smaller DQ width (that is, with only the failing DQS group) to reduce possible cross talk between adjacent I/O lanes.
4. Probe the stability of the V_{TT} power rail when running the calibration. An unstable V_{TT} power rail can cause the wrong command to be received by the memory component.
5. Probe the stability of the V_{CCIO} power rail when running calibration.
6. Test the design at lower frequencies and determine whether there is a frequency at which it passes.
7. Retest the failing board after eliminating the dependence on ODT signals. The following settings in the EMIF IP eliminate the dependence on ODT signals:
   - Dynamic ODT (Rtt_WR) value = Dynamic ODT off.
   - ODT R_{tt} nominal value = ODT Disable.
   - Output drive strength setting = RZQ/7 (34 ohm)
   - R_{tt} Park = RZQ /3 (80 Ohm)

   If you have enabled the Unified Calibration Debug Toolkit in your design, you can change the above settings on the **Calibrate Terminations** tab, without recompiling your design

Figure 153. Changing Termination Settings with the Unified Calibration Debug Toolkit

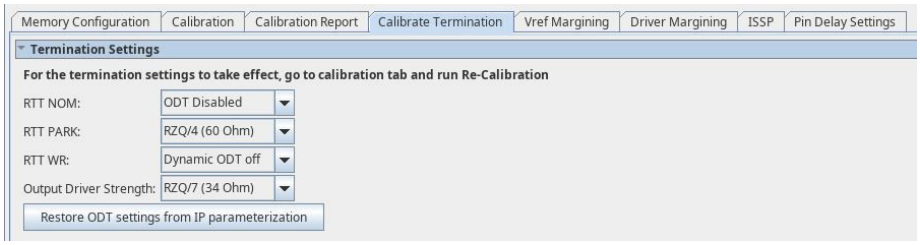
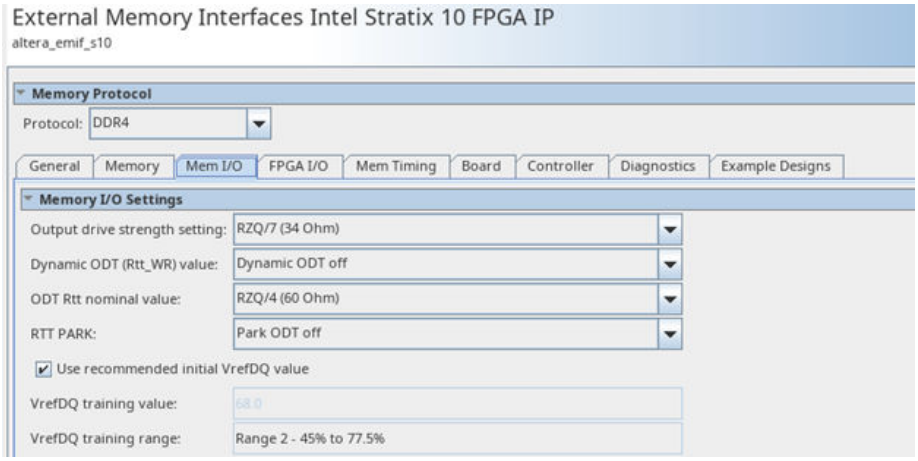


Figure 154. Changing Termination Setting when Regenerating EMIF IP – Recompilation Required



13.7.2.7.6. Debugging V_{REFIN} Calibration Failure

1. Ensure that the V_{CCIO} of the failing group is powered up to V_{CCIO}=1.2V at the FPGA side.
2. Regenerate the EMIF IP with other Initial V_{REFIN} values. It defaults to 61% when using the default FPGA I/O settings.

Figure 155. Changing the Initial V_{REFIN} Value

13.7.2.7.7. Debugging LFIFO Calibration Failure

LFIFO calibration failure is unexpected as it is performed at the end of the calibration to optimize the read latency.

If the earlier tests are passing with larger read latency, LFIFO calibration should not fail when trying to minimize the read latency.

13.7.2.7.8. Debugging Write Leveling Failure

1. Check the pin assignments for address and command pins. If the FPGA cannot write to the memory device correctly, the FPGA cannot get the correct data for comparison.
2. Compare the calibrated setting and margin for DQS enable for the failing group with other passing groups. If the DQS enable is not calibrated correctly, the FPGA cannot get correct data from the memory device.
3. Ensure the parameter editor specifies correct memory timing parameter, CAS, and Write CAS latency parameters. Incorrect parameter values can cause data corruption in the memory device.

13.7.2.7.9. Debugging Write Deskew Calibration Failure

If write deskew calibration is failing, perform the same checks as for *Debugging Read Deskew Calibration Failure*.

13.7.2.7.10. Debugging V_{REFOUT} Calibration Failure

1. Ensure the address and command pins are connected correctly and that every calibrated pin has sufficient margin.
2. Ensure that the V_{REFCA} pins on the DDR memory component are powered up to 0.6V.

13.7.3. On-Chip Debug Port for Intel Stratix 10 EMIF IP

The EMIF On-Chip Debug Port allows user logic to access the same calibration data used by the EMIF Toolkit, and allows user logic to send commands to the sequencer. You can use the EMIF On-Chip Debug Port to access calibration data for your design and to send commands to the sequencer just as the EMIF Toolkit would. The following information is available:

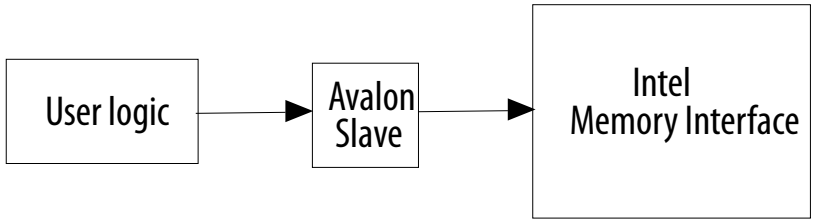
- Pass/fail status for each DQS group
- Read and write data valid windows for each group

In addition, user logic can request the following commands from the sequencer:

- Destructive recalibration of all groups
- Masking of groups and ranks
- Generation of per-DQ pin margining data as part of calibration

The user logic communicates through an Avalon-MM slave interface as shown below.

Figure 156. User Logic Access



13.7.3.1. EMIF On-Chip Debug Port

Access to on-chip debug is provided through software running on a Nios processor connected to the external memory interface.

If you enable the `Use Soft Nios Processor for On-Chip Debug` option, the system instantiates a soft Nios processor, and software files are provided as part of the EMIF IP.

Instructions on how to use the software are available in the following file: : `<variation_name>/altera_emif_arch_nd_<version number>/<synth|sim>/<variation_name>_altera_emif_arch_nd_<version number>_<unique ID>_readme.txt`.

13.7.3.2. Access Protocol

The On-Chip Debug Port provides access to calibration data through an Avalon-MM slave interface. To send a command to the sequencer, user logic sends a command code to the command space in sequencer memory. The sequencer polls the command space for new commands after each group completes calibration, and continuously after overall calibration has completed.

The communication protocol to send commands from user logic to the sequencer uses a multistep handshake with a data structure as shown below, and an algorithm as shown in the figure which follows.

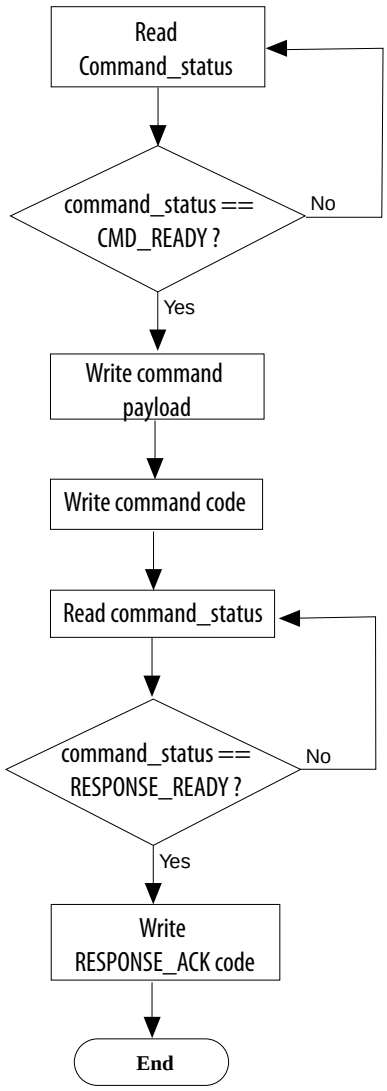
```
typedef struct_debug_data_struct {
...
// Command interaction
alt_u32 requested_command;
alt_u32 command_status;
alt_u32 command_parameters[COMMAND_PARAM_WORDS];...
}
```

To send a command to the sequencer, user logic must first poll the `command_status` word for a value of `TCLDBG_TX_STATUS_CMD_READY`, which indicates that the sequencer is ready to accept commands. When the sequencer is ready to accept commands, user logic must write the command parameters into `command_parameters`, and then write the command code into `requested_command`.

The sequencer detects the command code and replaces `command_status` with `TCLDBG_TX_STATUS_CMD_EXE`, to indicate that it is processing the command. When the sequencer has finished running the command, it sets `command_status` to `TCLDBG_TX_STATUS_RESPONSE_READY` to indicate that the result of the command is available to be read. (If the sequencer rejects the requested command as illegal, it sets `command_status` to `TCLDBG_TX_STATUS_ILLEGAL_CMD`.)

User logic acknowledges completion of the command by writing `TCLDBG_CMD_RESPONSE_ACK` to `requested_command`. The sequencer responds by setting `command_status` back to `STATUS_CMD_READY`. (If an illegal command is received, it must be cleared using `CMD_RESPONSE_ACK`.)

Figure 157. Debugging Algorithm Flowchart



13.7.4. Legacy Efficiency Monitor and Protocol Checker

For designs created prior to the Intel Quartus Prime software version 20.3, you can use the Legacy Efficiency Monitor and Protocol Checker to measure traffic efficiency on the Avalon-MM bus between the controller and user logic. For newer designs created in the Intel Quartus Prime software version 20.3 or later, use the new Efficiency Monitor.

The Legacy Efficiency Monitor and Protocol Checker measures read latencies, and checks the legality of Avalon commands passed from the master.

For Intel Stratix 10 devices, the Legacy Efficiency Monitor and Protocol Checker is available for the following configurations:

- DDR4 with hard PHY and hard controller
- QDR-IV with hard PHY and soft controller

The Legacy Efficiency Monitor and Protocol Checker is not available for PHY-only designs.

Efficiency Monitor

The Efficiency Monitor counts command transfers and wait times on the controller input and passes that information to the EMIF Debug Toolkit over an Avalon slave port. This summary of read and write throughput may be useful to you when experimenting with advanced controller settings, such as command and data reordering.

Protocol Checker

The Protocol Checker checks the legality of commands on the controller's input interface against Avalon interface specifications. If the Protocol Checker detects an illegal command, it sets a flag in a register on an Avalon slave port.

Read Latency Counter

The Read Latency Counter measures the minimum and maximum wait times for read commands to be serviced on the Avalon bus. Each read command is time-stamped and placed into a FIFO buffer upon arrival. The Read Latency Counter determines latency by comparing the time stamp to the current time when the master receives the first beat of the returned read data.

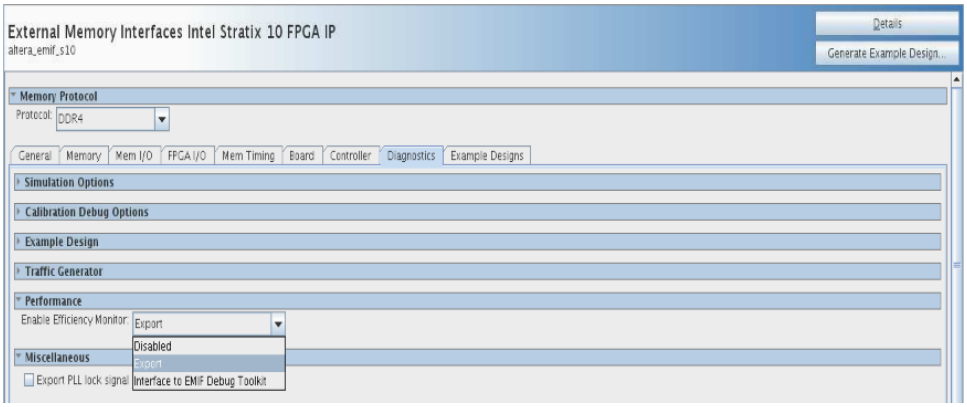
Note: Be aware that including the Legacy Efficiency Monitor and Protocol Checker when you generate your IP may make it more difficult to achieve timing closure.

13.7.4.1. Including the Legacy Efficiency Monitor and Protocol Checker in Your Generated IP

To include the Legacy Efficiency Monitor and Protocol Checker when you generate your IP, follow these steps.

1. On the **Diagnostics** tab of the parameter editor, turn on **Enable the Efficiency Monitor**.
   - If you want to see the results compiled by the Efficiency Monitor using the EMIF Debug Toolkit, select **Interface to EMIF Debug Toolkit**.
   - If you want to communicate directly to the Efficiency Monitor, select **Export**. (Refer to *Communicating Directly to the Efficiency Monitor and Protocol Checker* for a memory map of registers within the Efficiency Monitor and Protocol Checker.)

Figure 158. Enabling the Legacy Efficiency Monitor and Protocol Checker



13.7.4.2. Running the Legacy Efficiency Monitor with the External Memory Debug Toolkit

To see the results compiled by the Legacy Efficiency Monitor using the EMIF Debug Toolkit, follow these steps.

1. To launch the EMIF Debug Toolkit, select **Tools ➤ System Debugging Tools ➤ External Memory Interface Toolkit**.
2. To view the statistics, perform the following:
   a. Initialize connections.
   b. Link the project to the device.
   c. Create the memory interface connection.
   d. Create the Efficiency Monitor connection.

The following images illustrate the Legacy Efficiency Monitor statistics and Protocol Checker Summary statistics available in the EMIF Toolkit.

Figure 159. Legacy Efficiency Monitor Statistics in the EMIF Toolkit

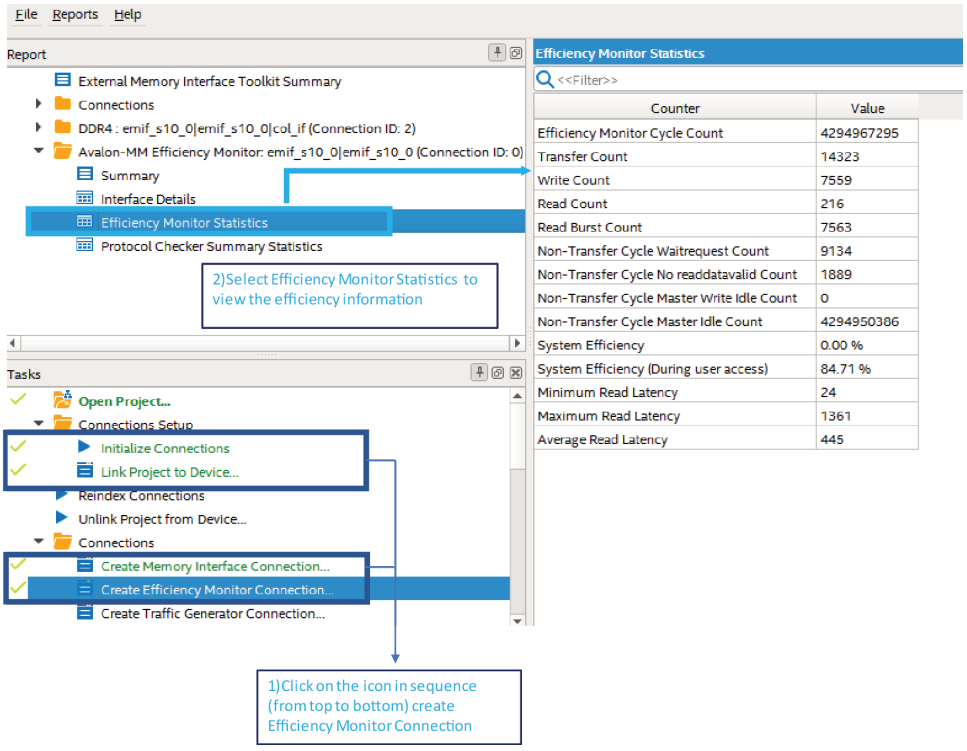
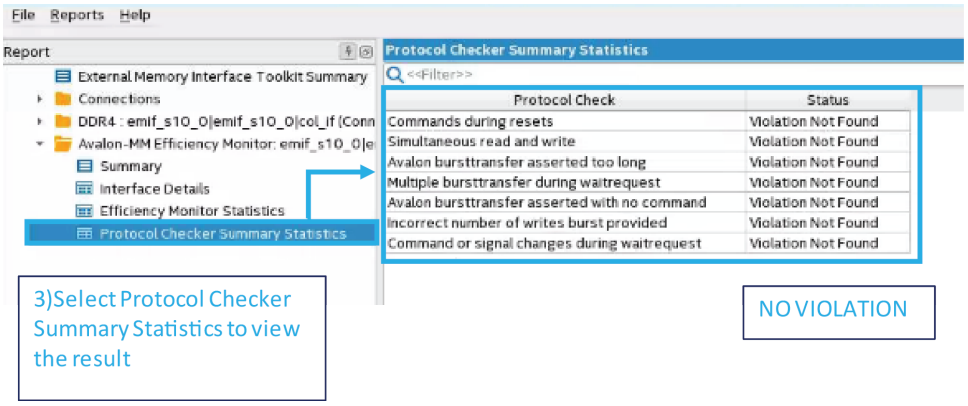


Figure 160. Protocol Checker Summary Statistics in the EMIF Toolkit



13.7.4.3. Communicating Directly to the Legacy Efficiency Monitor and Protocol Checker

When you export the Legacy Efficiency Monitor, a CSR Avalon slave interface is added to enable communication directly to the Efficiency Monitor and Protocol Checker without using the EMIF Debug Toolkit. You can create user logic to retrieve the efficiency statistic of the interface. The following table lists the memory map of the registers inside the Legacy Efficiency Monitor and Protocol Checker.

Before reading data in the CSR, you must issue a read command to address 0x01 to take a snapshot of the current data.

Table 354. Avalon CSR Slave and JTAG Memory Map

Address	Bit	Name	Default	Access	Description
0x01	31:0	Reserved	0	Read Only	Used internally by the EMIF Debug Toolkit to identify Efficiency Monitor type. This address must be read prior to reading the other CSR contents.
0x02	31:0	Reserved	—	—	Used internally by the EMIF Debug Toolkit to identify Efficiency Monitor version.
0x08	0		—	Write Only	Write a 0 to reset.
	7:1	Reserved	—	—	Reserved for future use.
	8		—	Write Only	Write a 0 to reset.
	15:9	Reserved	—	—	Reserved for future use.
	16		—	Read/Write	Starting and stopping statistics gathering.
	23:17	Reserved	—	—	Reserved for future use.
	31:24	Efficiency Monitor Status	—	Read Only	• bit 0: Efficiency Monitor stopped • bit 1: Waiting for start of pattern • bit 2: Running • bit 3: Counter saturation
0x10	15:0	Efficiency Monitor address width	—	Read Only	Address width of the Efficiency Monitor.
	31:16	Efficiency Monitor data width	—	Read Only	Data width of the Efficiency Monitor.
0x11	15:0	Efficiency Monitor byte enable	—	Read Only	Byte enable width of the Efficiency Monitor.
	31:16	Efficiency Monitor burst count width	—	Read Only	Burst count width of the Efficiency Monitor.
0x14	31:0	Cycle counter	—	Read Only	Clock cycle counter for the Efficiency Monitor. Lists the number of clock cycles

continued...

Address	Bit	Name	Default	Access	Description
					elapsed before the Efficiency Monitor stopped.
0x18	31:0	Transfer counter	—	Read Only	Counts any read or write data transfer cycle.
0x1C	31:0	Write counter	—	Read Only	Counts write requests, including those during bursts.
0x20	31:0	Read counter	—	Read Only	Counts read requests.
0x24	31:0	Read total counter	—	Read Only	Counts read requests (total burst requests).
0x28	31:0	NTC waitrequest counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to slave wait request high.
0x2C	31:0	NTC noreaddatavalid counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to slave not having read data.
0x30	31:0	NTC master write idle counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to master not issuing command or pause in write burst.
0x34	31:0	NTC master idle counter	—	Read Only	Counts Non Transfer Cycles (NTC) due to master not issuing command anytime.
0x40	31:0	Read latency minimum	—	Read Only	The lowest read latency value.
0x44	31:0	Read latency maximum	—	Read Only	The highest read latency value.
0x48	31:0	Read latency total [31:0]	—	Read Only	The lower 32 bits of the total read latency.
0x49	31:0	Read latency total [63:32]	—	Read Only	the upper 32 bits of the total read latency.
0x50	7:0	Illegal command	—	Read Only	Bits used to indicate which illegal command has occurred.

continued...

Address	Bit	Name	Default	Access	Description
					Each bit represents a unique error.
	31:8	Reserved	—	—	Reserved for future use.

13.7.5. New Efficiency Monitor

You can instantiate the new Efficiency Monitor as part of the generated design example. The Efficiency Monitor is a block with control and status registers, that you can use to measure efficiency on the Avalon interface.

You can enable, disable, or reset the Efficiency Monitor through control registers in real time. The Efficiency Monitor also provides status registers containing detailed efficiency information.

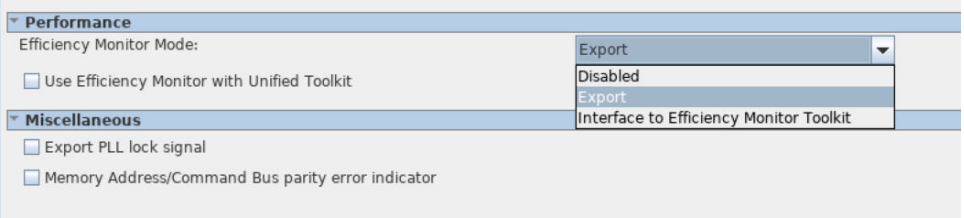
13.7.5.1. Enabling the Efficiency Monitor in a Design Example

To enable the Efficiency Monitor, follow these steps.

In the **Performance** group on the **Diagnostics** tab in the parameter editor, set **Efficiency Monitor Mode** to one of the following values:

- **Export**: Allows you to connect your own RTL logic to control the Efficiency Monitor and read status registers.
- **Interface to Efficiency Monitor Toolkit**: Allows use of a unified toolkit GUI in the System Console. To use the Efficiency Monitor with the Unified Debug Toolkit instead of the legacy External Memory Interface Debug Toolkit, check the **Use Efficiency Monitor with Unified Toolkit** option. Refer to *Opening the Efficiency Monitor Toolkit* for more information.

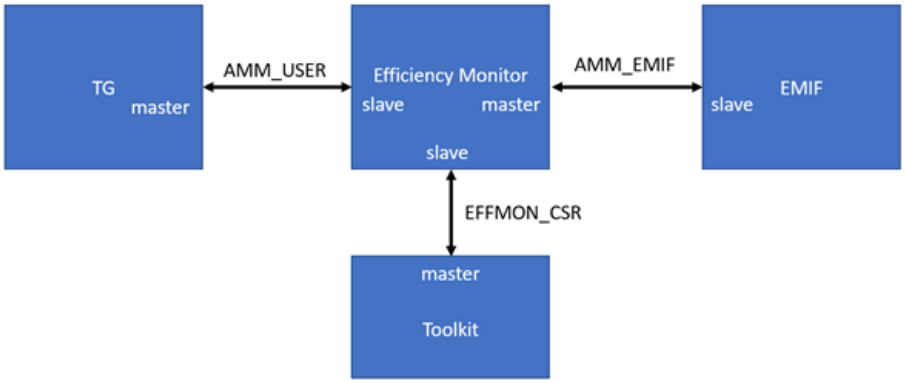
Figure 161. Efficiency Monitor Mode Setting



13.7.5.2. Efficiency Monitor Block Descriptions

The Efficiency Monitor accepts Avalon signals from a master and sends commands downstream to a slave without modifying anything on the interface.

Figure 162. Sample Efficiency Monitor Topology



amm_emif interface

The Efficiency Monitor passes traffic downstream to the external memory interface on this interface.

amm_user interface

The traffic generator (or custom user logic) initiates traffic and passes it to the Efficiency Monitor over this interface.

effmon_csr interface

This interface consists of several configuration and status registers. The **Efficiency Monitor Mode** parameter controls whether this interface is exported for you to provide a custom master, or connected internally so that the System Console can act as master on this interface.

13.7.5.3. Control and Status Registers

Status registers hold a record of the transactions that happen on the Avalon interface and contain useful information for the efficiency calculation. You can enable, disable, or reset the recording of transactions on the status registers, through the control registers.

The following table summarizes the available registers.

Table 355. Control and Status Registers

Symbol Address	Register Name	Readable or Writeable	Register Description
0x0	EFFMON_START	Readable and Writeable	• Write a value of 1 to enable the Efficiency Monitor. • Write a value of 0 to disable the Efficiency Monitor.
0x4	EFFMON_READ_COUNTER	Readable	Number of read commands issued.
0x8	EFFMON_WRITE_COUNTER	Readable	Number of write commands issued.

continued...

Symbol Address	Register Name	Readable or Writeable	Register Description
0xC	EFFMON_CYCLE_COUNTER	Readable	Number of clock cycles after the first command (read or write) issued on the interface. (This counter stops at EFFMON_CYCLE_COUNTER_MAX.)
0x10	EFFMON_COUNTER_SATURATION	Readable	• A value of 1 indicates that EFFMON_CYCLE_COUNTER has reached EFFMON_CYCLE_COUNTER_MAX, and further data is not collected until all status registers are cleared. • A value of 0 indicates the counter has not saturated.
0x14	EFFMON_RDLAT_MIN	Readable	Minimum read latency. Read latency is measured from the clock cycle at which a read command is issued to the clock cycle where the corresponding readdatavalid signal is asserted.
0x18	EFFMON_RDLAT_MAX	Readable	Maximum read latency. Read latency is measured from the clock cycle at which a read command is issued to the clock cycle where the corresponding readdatavalid signal is asserted.
0x1C	EFFMON_RDLAT_TOTAL_L	Readable	Total read latency (lower 32 bits). Read Latency is measured from the clock cycle at which a read command is issued to the clock cycle where the corresponding readdatavalid signal is asserted.
0x20	EFFMON_RDLAT_TOTAL_H	Readable	Total read latency (upper 32 bits). Read Latency is measured from the clock cycle at which a read command is issued to the clock cycle where the corresponding readdatavalid signal is asserted.
0x24	EFFMON_READDATAVALID_COUNTER	Readable	Total number of clock cycles in which readdatavalid is asserted.
0x28	EFFMON_TRANSFER_COUNTER	Readable	Indicates the number of cycles where amm_write and amm_ready are asserted or amm_readdatavalid is asserted.
0x2C	EFFMON_COMMAND_WAIT_COUNTER	Readable	Indicates the total number of cycles in which the issuance of a read or write command was stalled due to waitrequest being asserted.
0x30	EFFMON_NO_READDATAVALID_COUNTER	Readable	Indicates the number of cycles where readdatavalid is low after a read command has been issued.
0x34	EFFMON_MASTER_IDLE_COUNTER	Readable	Indicates the number of cycles where there is no read or write from the master after the first command (read or write) has been issued on the interface.
0x38	EFFMON_MASTER_WRIDLE_COUNTER	Readable	Indicates the number of cycles in which the master is unable to provide valid write data and is forced to deassert WRITE within a multi-word burst.
0x3C	EFFMON_STATUS_CLEAR	Readable and Writeable	Write a value of 1 to clear all the status registers. (This value is set back to 0 automatically, after the status registers are cleared.)
0x40	EFFMON_CYCLE_COUNTER_SNAPSHOT	Readable	Stores a snapshot of EFFMON_CYCLE_COUNTER, as it was at the time of the last transaction on the interface (such as a read, a write, or a read_data_valid). This is the value used as the denominator for efficiency calculations in the toolkit GUI.

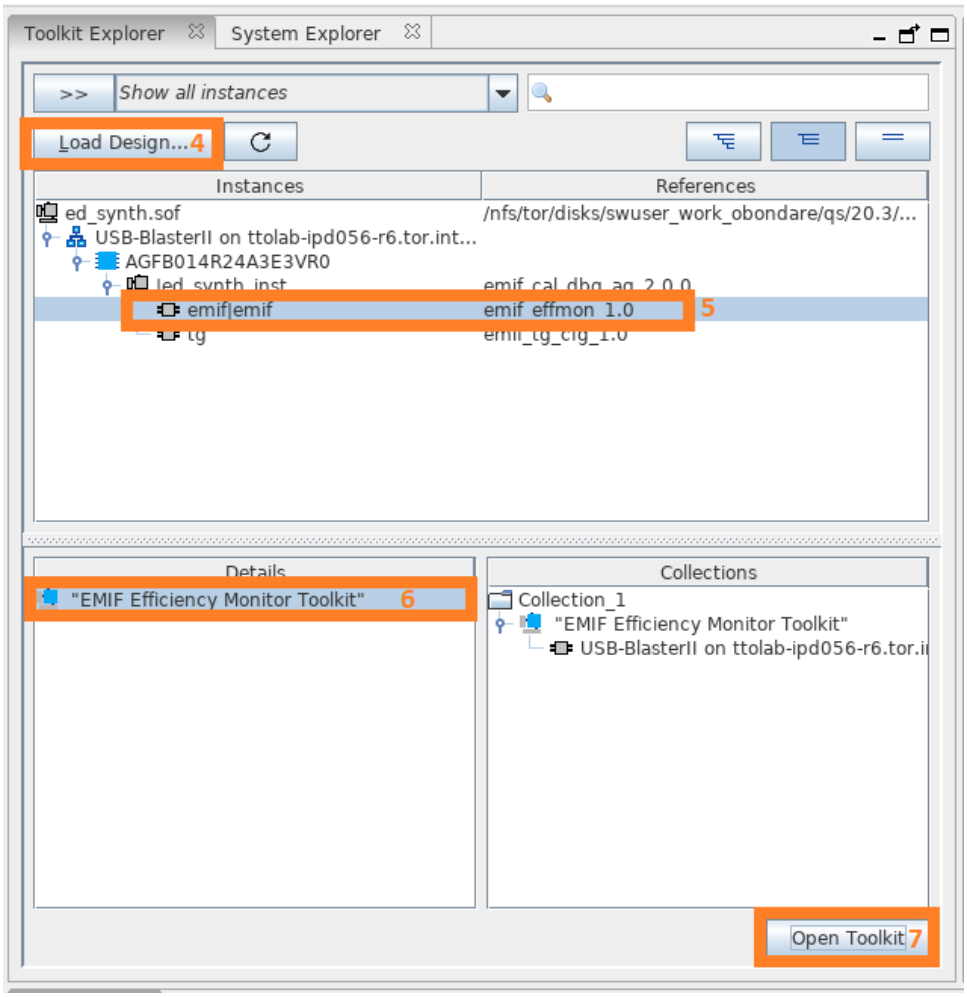
13.7.5.4. Opening the Efficiency Monitor

The Efficiency Monitor GUI runs on the System Console. You can launch the System Console from the **Tools** menu in the Intel Quartus Prime software.

Connecting the Efficiency Monitor

1. Compile a design with an Efficiency Monitor, as described in *Enabling the Efficiency Monitor in a Design Example*.
2. Program the `.sof` file onto a device.
3. Launch the System Console—either through the Intel Quartus Prime software, or directly from the command line.
4. Load the `.sof` file of your design.
5. Select the `emif_effmon` toolkit instance.
6. Select **EMIF Efficiency Monitor Toolkit** in the **Details** section of the **Toolkit Explorer** in the System Console.
7. Click **Open Toolkit** to launch the Efficiency Monitor toolkit.

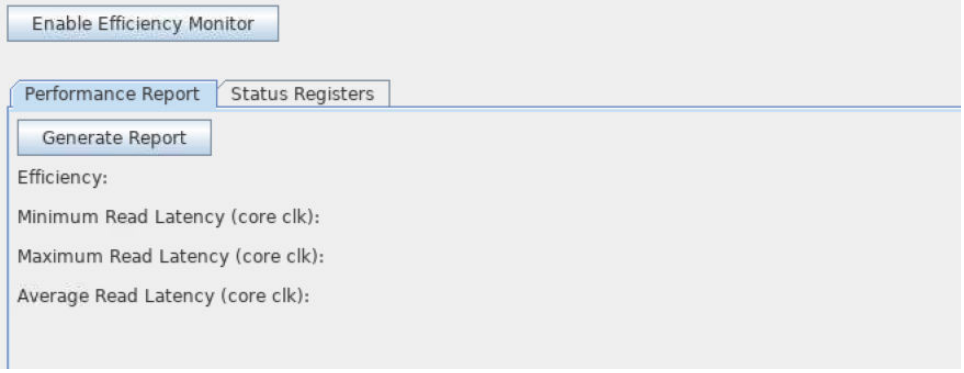
Figure 163. Connecting the Efficiency Monitor



Starting and Stopping the Efficiency Monitor

To start collecting information on the Avalon interface, click **Enable Efficiency Monitor**.

Figure 164. Enable Efficiency Monitor



To stop the Efficiency Monitor, click **Disable Efficiency Monitor**.

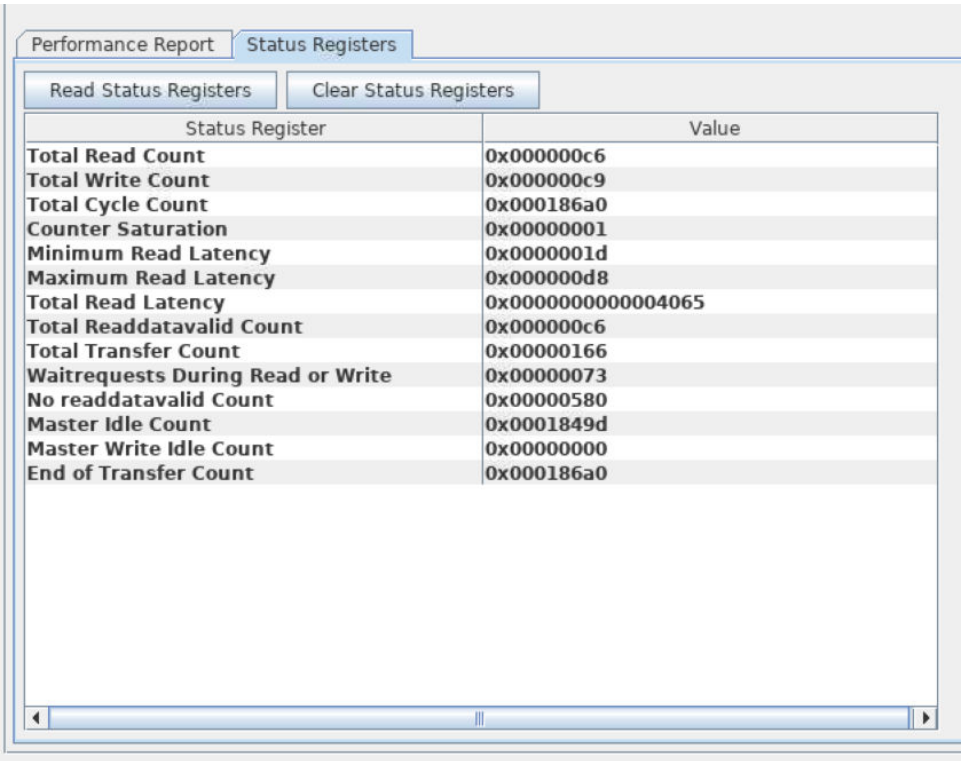
Figure 165. Disable Efficiency Monitor

Status Registers

The **Status Registers** tab lets you:

- Read all status registers from the device, by clicking **Read Status Registers**.
- Clear all status registers, by clicking **Clear Status Registers**.

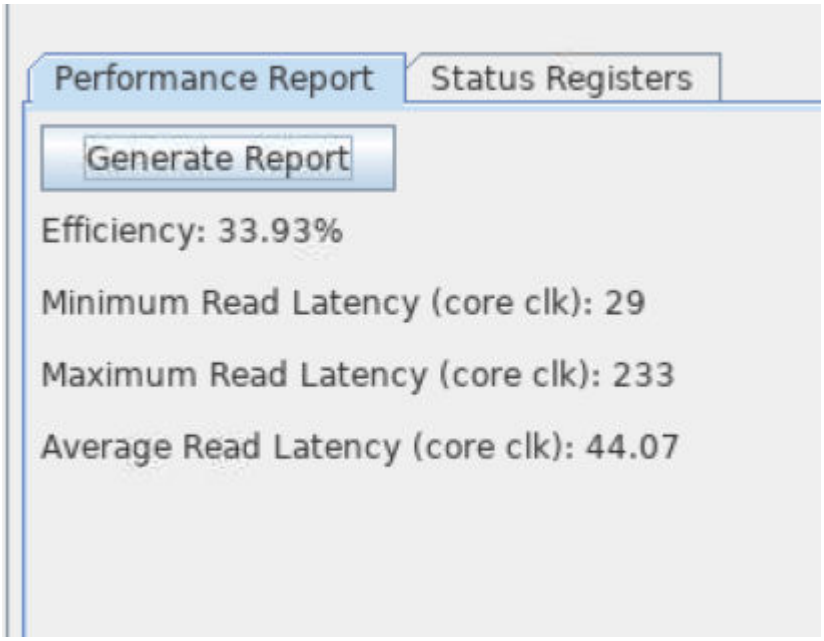
Figure 166. Status Registers



Performance Report

The **Performance Report** tab displays efficiency as a percentage, and the read latency report.

Figure 167. Performance Report Tab



The reported values are calculated using values in the status registers, as follows:

- Efficiency = (EFFMON_TRANSFER_COUNTER ÷ EFFMON_END_OF_TRANS_COUNTER) × 100%
- Minimum Read Latency = EFFMON_RDLAT_MIN
- Maximum Read Latency = EFFMON_RDLAT_MAX
- Average Read Latency = EFFMON_RDLAT_TOTAL ÷ EFFMON_READDATAVALID_COUNTER

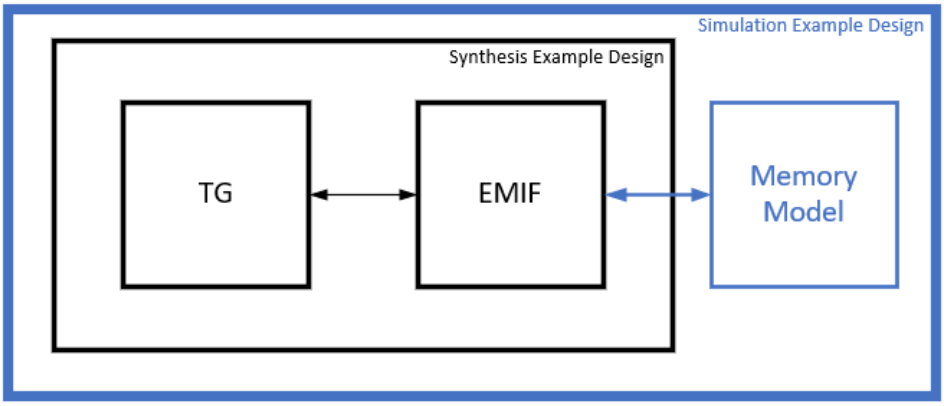
13.8. Using the Default Traffic Generator

A Traffic Generator (TG) IP instance is present in any EMIF Design Example; its purpose is to connect an EMIF IP instance via the `ctrl_amm_*` port(s) and send sample traffic (writes and reads) through the EMIF to the memory. The TG traffic pattern is parameterizable, and it is configured to start once TG comes out of reset.

The Traffic Generator also compares the written data and the read data, and sets one of the following status bits:

- **traffic_gen_pass (TGP ISSP):** Indicates that all write and read commands were sent to the EMIF, all read responses were received, and all writes and reads matched as expected.
- **traffic_gen_fail (TGF ISSP):** Indicates that all write and read commands were sent to the EMIF, all read responses were received, but one or more write-read mismatches have occurred.
- **traffic_gen_timeout (TGT ISSP):** Indicates that one or more of the expected read responses were not received.

Figure 168. EMIF Design Example Overview



For general information about the generated EMIF design example, refer to the External Memory Interfaces Intel Stratix 10 FPGA IP Design Example User Guide.

You can use the traffic generator for a variety of analysis and debugging applications, including the following:

- Verifying that an external memory interface is configured and working correctly, in simulation and in hardware.
- Evaluating the stability of the interface, as well as the calibration results. (Refer to the Driver Margining Tab topic.)
- Isolating hardware issues such as single pin failures.
- Distinguishing between read failures and write failures.
- Running infinite traffic for hardware debugging.
- Measuring the efficiency of the interface.

13.8.1. Reading the Default Traffic Generator Status

To observe the overall traffic generator (TG) status, you should route each of the following top-level signals to external pins connected to LEDs or to test points for monitoring with an oscilloscope: `traffic_gen_pass`, `traffic_gen_fail`, and `traffic_gen_timeout`.Alternatively, you can enable In-System Sources and Probes (ISSPs) in the design, which you can read using Signal Tap, the System Console, or the Calibration Debug Toolkit.

The traffic generator provides detailed failure information, as described below.

Pass-Not-Fail (PNF) bits

The width of the `pnf_per_bit` bus equals the data width on the Avalon Control interface. Each PNF bit represents the status of each data bit, as gathered from comparison between the data written to a particular address and the read response from the same address.

`pnf_per_bit[`*x*`]` is high provided that no write-read mismatches have occurred on bit *x*. PNF bits are persistent, meaning that once a bit is set low due to a data mismatch, it remains low until the next TG reset.

The PNF bits map to the control interface data bits in a 1:1 manner. To understand the mapping to data pins on the memory side, consider the example of a 32-bit DDR4, quarter-rate interface. This interface has a control data width of 256, where the following are true:

- pnf[0] maps to dq[0] for the first beat of the memory bus burst
- pnf[1] maps to dq[1] for the first beat of the memory bus burst
- ...
- pnf[31] maps to dq[31] for the first beat of the memory bus burst
- pnf[32] maps to dq[0] for the second beat of the memory bus burst
- ...
- pnf[64] maps to dq[0] for the third beat of the memory bus burst
- ...
- pnf[96] maps to dq[0] for the fourth beat of the memory bus burst
- ...
- pnf[128] maps to dq[0] for the fifth beat of the memory bus burst
- pnf[160] maps to dq[0] for the sixth beat of the memory bus burst
- ...
- pnf[192] maps to dq[0] for the seventh beat of the memory bus burst
- ...
- pnf[224] maps to dq[0] for the eighth beat of the memory bus burst

A similar mapping approach applies to any other supported interface memory bus width.

Information about First Observed Failure

The traffic generator has registers that store the address of the first data mismatch, the expected data, the read data, etcetera. These registers can be read through ISSPs or by adding them to a Signal Tap waveform. For a detailed description of all ISSPs that are present in the example design, refer to ISSPs Tab.

Write-Once-Read-Many (WORM) Mode

When enabled, WORM mode causes the traffic generator to provide information about failures by performing an additional read from the address where the failure occurred. You can then read the TG status registers and analyze the results accordingly:

- If both reads produced the same `readdata` value, then the error was likely in the write path.
- If each read produced a different `readdata` value, then the error is likely in the read path.

To enable WORM mode, set the WORM ISSP HIGH. When the TG is reset while the WORM bit is set to HIGH, TG runs in WORM mode. Refer to ISSPs Tab for a list of ISSPs that are present in a design example. These ISSPs store the data observed while TG runs in this mode.

13.8.2. Running Infinite Traffic using the Default Traffic Generator

By default, the traffic generator runs through one iteration of its tests. For general debugging, you may find it preferable to let the tests run continuously.

To configure the tests to run continuously, follow these steps:

1. Locate the `ed_synth.tg.v` file in the `<project_directory>/ip/ ed_synth/ed_synth_tg/synth` directory and open the file in a text editor.
2. Search for `.TEST_DURATION ("SHORT")`, and change it to `.TEST_DURATION ("INFINITE")`,
3. Save your change, recompile the design, and rerun the simulation for the change to take effect.

13.8.3. Changing the Reset Trigger of the Default Traffic Generator

As generated, the design example project responds to an active-high reset pulse on the `local_reset_req` signal.

If you prefer to have a level-sensitive, typically active-low reset signal as was common with earlier device families, you can invert the design example reset signal by making the following RTL changes to the `ed_synth.v` file:

- Add the following two lines in the wire declaration section:

```
wire reset_invert;
assign reset_invert = !local_reset_req;
```

- Where the reset block is instantiated, change the `local_reset_req` to connect to the inverted reset signal called `reset_invert`, as follows:

```
ed_synth_local_reset_combiner local_reset_combiner (
.clk
(emif_fm0_0_pll_ref_clk_out_clk),
.reset_n
(emif_fm0_0_pll_locked_pll_locked),
.local_reset_req
(local_reset_req),
.local_reset_req
(reset_invert),
.local_reset_req_out_0
(local_reset_combiner_local_reset_req_out_0_local_reset_req),
.local_reset_done
(local_reset_done),
.local_reset_done_in_0
(emif_fm0_0_local_reset_status_local_reset_done)
);
```

In addition, it is a good idea — though not mandatory — to also run analysis and elaboration, to help show project structure and verify assignments.

13.8.4. Observing Generated Traffic with Signal Tap

When using Signal Tap to observe the traffic generated by the Default Traffic Generator, the following are the recommended signals to tap.

Table 356. Signals to Tap Using Signal Tap

Pins: All	local_reset_req
	local_reset_done
	local_cal_success
	local_cal_fail
	traffic_gen_pass
	traffic_gen_fail
	traffic_gen_timeout
Signal Tap : pre-synthesis	Pre-synthesis and search for signal names with wildcards as appropriate
Pass-not-fail signals	pnf_per_bit
	pnf_per_bit_persist
Avalon bus signals	amm_read_0
	amm_readdatavalid_0
	amm_ready_0
	amm_write_0
	amm_address_0
	amm_burstcount_0
	amm_byteenable_0
	amm_readdata_0
	amm_writedata_0
For the Signal Tap clock, Signal Tap : Pre-synthesis	emif_usr_clk

13.9. Using the Configurable Traffic Generator (TG2)

The generated EMIF design example includes a traffic generator block with control and status registers, that you can use to send sample traffic through the external memory interface to the memory device.

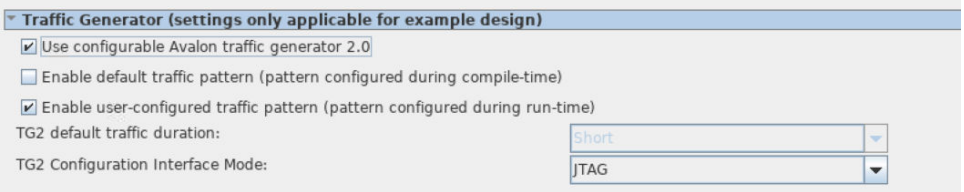
In the Configurable Traffic Generator (TG2) (altera_tg_avl_2), you can configure the traffic pattern in real time through control registers—meaning that you do not have to recompile the design to change or relaunch the traffic pattern. This traffic generator provides fine control over the type of traffic that it sends on the EMIF control interface. Additionally, it provides status registers that contain detailed failure information.

13.9.1. Enabling the Traffic Generator in a Design Example

You can enable the traffic generator from the **Diagnostics** tab in the EMIF parameter editor.

To enable the traffic generator, turn on **Use configurable Avalon traffic generator 2.0** on the **Diagnostics** tab.

Figure 169.

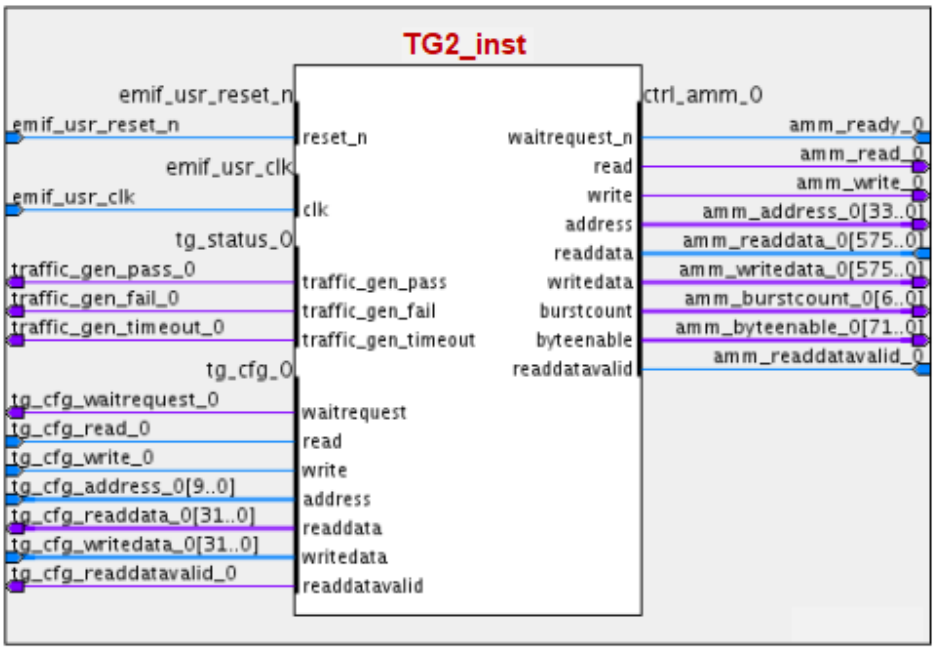


- You may choose to disable the **default traffic pattern** stage or the **user-configured traffic** stage, but you must have at least one stage enabled. For information on these stages, refer to Default Traffic Pattern and User-Configured Traffic Pattern.
- The **TG2 test duration** parameter applies only to the default traffic pattern. You may choose a test duration of *short*, *medium*, or *infinite*.
- You may choose either of two values for the **TG2 Configuration Interface Mode** parameter:
  - *JTAG*: Allows use of a GUI in the system console. For more information, refer to Traffic Generator Configuration User Interface.
  - *Export*: Allows use of custom RTL logic to control the traffic pattern.

13.9.2. Traffic Generator Block Description

The traffic generator sends traffic through the external memory interface using the ctrl_amm interface.

Figure 170. Traffic Generator Block Diagram



ctrl_amm interface

The TG2 traffic generator replaces user logic as a master on the ctrl_amm interface, and sends traffic to the external memory interface.

tg_status interface

The tg_status interface mimics simple traffic generator status; you may assign these pins to LEDs or leave them unused.

tg_cfg interface

The **TG2 Configuration Interface Mode** parameter determines whether this interface is exported to allow a custom configuration master, or internally connected such that system-console becomes the master on this interface. This interface consists of several configuration and status registers, described in the User-Configured Traffic Pattern and Traffic Generator Status sections.

13.9.3. Default Traffic Pattern

The traffic generator's default traffic pattern consists of three stages, which run sequentially.

If you select the **Enable default traffic pattern** parameter, the following three traffic stages run when the traffic generator comes out of the reset state:

Table 357.

Traffic Stage	Description
Single RW stage	The traffic generator sends a single write instruction, followed by a single read instruction, and compares the results. This loop of a single write followed by a single read is issued three times.
Block RW stage	The traffic generator sends a block of write instructions followed by the same number of read instructions—this sequence is called a *loop*. The number of loops performed, as well as the number of writes and reads performed within each loop, is determined by the value that you choose for the **TG2 test duration** parameter. The traffic generator executes this stage once for each of the three address modes (Sequential, Random, and Random-Sequential).
Byte-enable stage	• The traffic generator randomly generates a byte-enable value and performs a block of writes to a start address, in Sequential Address Mode. • The traffic generator then uses the inverted write and inverted byte-enable value, and performs a second block of writes, starting at the same address. • Finally, the traffic generator issues reads from the same start address with all bytes enabled, and compares the read data to the write data and inverted write data, where applicable.

To run the default traffic pattern, the traffic generator uses the same infrastructure as the user-configured traffic stage; that is, for each part of the default traffic pattern, the traffic generator sets the configuration registers to pre-set default values. The registers used to configure this traffic pattern are described in more detail in the User-Configured Traffic Pattern topic.

13.9.4. Configuration and Status Registers

You can configure the user traffic pattern by writing to configuration registers that influence the resulting traffic pattern.

Configuration registers that govern the resulting traffic pattern affect one of the following aspects of the pattern:

- Test duration / Instruction pattern
- Address pattern
- Data pattern

Note: This section describes the registers that configure a traffic pattern as seen on the ctrl_amm interface.

Table 358. Configuration and Status Registers

Symbol Address	Register Name	Register Width	Number of Registers	Readable or Writeable	Register Section	Register Description
0x0	TG_VERSION	32	1	Readable	N/A	Version number of the traffic generator address map.
0x4	TG_START	1	1	Writeable	N/A	Perform a write to this register to start the traffic generator (any value).
0x8	TG_LOOP_COUNT	32	1	Readable and Writeable	Test Duration/ Instruction Pattern	The number of read/write loops to run. A loop is defined as a block of writes followed by a block of reads. If this value is set to 0, the traffic generator will run infinite loops.
0xC	TG_WRITE_COUNT	12	1	Readable and Writeable	Test Duration/ Instruction Pattern	The number of unique writes to perform in each loop.
0x10	TG_READ_COUNT	12	1	Readable and Writeable	Test Duration/ Instruction Pattern	Number of unique reads to perform in each loop.
0x14	TG_WRITE_REPEAT_COUNT	16	1	Readable and Writeable	Test Duration/ Instruction Pattern	Number of times to repeat each write operation.
0x18	TG_READ_REPEAT_COUNT	16	1	Readable and Writeable	Test Duration/ Instruction Pattern	Number of times to repeat each read operation.
0x20	TG_CLEAR	4	1	Readable and Writeable	Status	Clears the failure status registers. Allows clearing these registers independently from one another by writing a 1 to the following bits. BIT0 - Clears the recorded PNF data. BIT1 - Clears the recorded number of Avalon reads. BIT2 - Clears the recorded data of the first failure (address, expected data, and actual data).

continued...

Symbol Address	Register Name	Register Width	Number of Registers	Readable or Writeable	Register Section	Register Description
						BIT3 - Clears the recorded data of address overflow due to burst length (last address written to, failure status).
0x1C	TG_BURST_LENGTH	7	1	Readable and Writeable	Test Duration/ Instruction Pattern	Avalon burst length.
0x38	TG_RW_GEN_IDLE_COUNT	16	1	Readable and Writeable	Test Duration/ Instruction Pattern	Number of cycles for which the traffic generator remains idle between a write block and the next read block.
0x3C	TG_RW_GEN_LOOP_IDLE_COUNT	16	1	Readable and Writeable	Test Duration/ Instruction Pattern	Number of cycles for which the traffic generator remains idle between a read block and the next write block.

0x40	TG_SEQ_START_ADDR_WR	32	12	Readable and Writeable	Address Pattern	Start address for writes; used as a seed address in Random and Fixed Modes. Consists of 12 registers, 2 for each address field.[CS1] Each pair of adjacent registers represents the lower 32 bits and upper 32 bits of a start address for the corresponding field. For example:
						`TG_SEQ_START_ADDR_WR = start_addr_field0[31:0]`
						`TG_SEQ_START_ADDR_WR+1 = start_addr_field0[63:32]`
						`…`
						`TG_SEQ_START_ADDR_WR +10=start_addr_field_5[31:0]`
						`TG_SEQ_START_ADDR_WR +11=start_addr_field_5[63:32]`

0x80	TG_ADDR_MODE_WR	2	6	Readable and Writeable	Address Pattern	Address mode for writes. Consists of 6 registers, where each register specifies the write address mode for the corresponding address field. Available address modes include (see *Address Generator Modes* for details):
						`TG_ADDR_MODE == 0: Fixed`
						`TG_ADDR_MODE == 1: Random`
						`TG_ADDR_MODE == 2: Sequential`
						`TG_ADDR_MODE == 3: Unused Field`

0xC0	TG_RETURN_TO_START_ADDR	1	1	Readable and Writable	Address Pattern	If set to 1, specifies to return to start address in each loop. If set to 0,

continued...

Symbol Address	Register Name	Register Width	Number of Registers	Readable or Writeable	Register Section	Register Description
						specifies to resume the address pattern from where the previous loop left off.
0x84	TG_RAND_SEQ_ADDRS_RD			Readable and Writeable	Address Pattern	Number of times to increment sequentially on the random base address before generating a new random write address for reads.
0x88	TG_PASS	1	1	Read Only	Status	A value of 1 indicates that the traffic generator passed at the end of all test stages.
0x8C	TG_FAIL	1	1	Read Only	Status	A value of 1 indicates that the traffic generator failed at the end of all test stages.
0x90	TG_FAIL_COUNT_L	32	1	Read Only	Status	The number of failed reads (lower 32 bits).
0x94	TG_FAIL_COUNT_H	32	1	Read Only	Status	The number of failed reads (upper 32 bits).
0x98	TG_FIRST_FAIL_ADDR_L	32	1	Read Only	Status	The address of the first failed read (lower 32 bits).
0x9C	TG_FIRST_FAIL_ADDR_H	32	1	Read Only	Status	The address of the first failed read (upper 32 bits).
0xA0	TG_TOTAL_READ_COUNT_L	32	1	Read Only	Status	The number of read operations executed - sent and received (lower 32 bits).
0xA4	TG_TOTAL_READ_COUNT_H	32	1	Read Only	Status	The number of read operations executed - sent and received (upper 32 bits).
0xA8	TG_TEST_COMPLETE	1	1	Read Only	Status	A value of 1 indicates that the traffic generator run has completed.
0xAC	TG_INVERT_BYTEEN	1	1	Readable and Writeable	Data/Byte-Enable Pattern	If set to 1, specifies to invert byte-enable values and write_data.
0xB4	TG_USER_WORM_EN	1	1	Readable and Writeable	Test Duration/ Instruction Pattern	If set to 1, enables WORM mode.
0xB8	TG_TEST_BYTEEN	1	1	Readable and Writeable	Data/Byte-Enable Pattern	If set to 1, specifies to change the comparison pass/fail condition, such that for each byte: • If byte-enable is high, compare readdata with writedata at this byte. • If byte-enable is low, compare readdata with inverted writedata at this byte.
0xC4	TG_NUM_DATA_GEN	5	1	Read Only	Data/Byte-Enable Pattern	Number of data generators in the design.

continued...

Symbol Address	Register Name	Register Width	Number of Registers	Readable or Writeable	Register Section	Register Description
0xC8	TG_NUM_BYTEEN_GEN	5	1	Read Only	Data/Byte-Enable Pattern	Number of byte-enable generators in the design.
0xDC	TG_RDATA_WIDTH	32	1	Read Only	Data/Byte-Enable Pattern	Width of read_data, write_data, and PNF signals.
0xEC	TG_ERROR_REPORT	32	1	Read Only	Status	Reports illegal configurations of the traffic generator. Value is 0 when no error is present. (Details about error codes can be found below.)
0xF0	TG_DATA_RATE_WIDTH_RATIO	4	1	Read Only	Data/Byte-Enable Pattern	Data rate width ratio is the ratio between the data width at the ctrl_amm interface and the data width at the memory interface.
0x100	TG_SEQ_ADDR_INCR	8	6	Readable and Writeable	Address Pattern	Sequential address increment for both the read and write addresses. This value is only used if the field mode is set to Sequential. Consists of 6 registers, where each register specifies the sequential address increment for both the read and write address generators of the corresponding address field. For field 0 this value must be greater than or equal to the value of TG_BURST_LENGTH.
0x140	TG_SEQ_START_ADDR_RD	32	12	Readable and Writeable	Address Pattern	Start address for reads; used as a seed address in Random and Fixed modes. Organized as 2*6=12 registers to reserve space for 64-bit start addresses across all address fields. The start addresses are organized such that each pair of 2 adjacent start addresses represent the lower 32 bits and upper 32 bits of a start address.

0x180	TG_ADDR_MODE_RD	2	6	Readable and Writeable	Address Pattern	Address mode for reads. Consists of 6 registers, where each register specifies the read address mode for the corresponding address field. Available address modes include (see *Address Generator Modes* for details):
						`TG_ADDR_MODE == 0: Fixed`
						`TG_ADDR_MODE == 1: Random`
						`TG_ADDR_MODE == 2:`

continued...

Symbol Address	Register Name	Register Width	Number of Registers	Readable or Writeable	Register Section	Register Description
						`Sequential TG_ADDR_MODE == 3: Unused Field`
0x1C0	TG_PASS	1	1	Read Only	Status	A value of 1 indicates that the traffic generator passed at the end of all test stages
0x1C4	TG_FAIL	1	1	Read Only	Status	A value of 1 indicates that the traffic generator failed at the end of all test stages.
0x1C8	TG_FAIL_COUNT_L	32	1	Read Only	Status	The number of failed reads (lower 32 bits).
0x1CC	TG_FAIL_COUNT_H	32	1	Read Only	Status	The number of failed reads (upper 32 bits).
0x1D0	TG_FIRST_FAIL_ADDR_L	32	1	Read Only	Status	The address of the first failed read (lower 32 bits).
0x1D4	TG_FIRST_FAIL_ADDR_H	32	1	Read Only	Status	The address of the first failed read (upper 32 bits).
0x1D8	TG_TOTAL_READ_COUNT_L	32	1	Read Only	Status	The number of read operations executed - sent and received (lower 32 bits).
0x1DC	TG_TOTAL_READ_COUNT_H	32	1	Read Only	Status	The number of read operations executed - sent and received (upper 32 bits).
0x1E0	TG_TEST_COMPLETE	1	1	Read Only	Status	A value of 1 indicates that the traffic generator run has completed.
0x1E4	TG_INVERT_BYTEEN	1	1	Readable and Writable	Data/Byte-Enable Pattern	If set to 1, specifies to invert byteenable values and write_data.
0x1EC	TG_USER_WORM_EN	1	1	Readable and Writable	Test Duration/ Instruction Pattern	If set to 1, enables WORM mode.
0x1F0	TG_TEST_BYTEEN	1	1	Readable and Writable	Data/Byte-Enable Pattern	If set to 1, specifies to change the comparison pass/fail condition, such that for each byte: • If byte-enable is high, compare readdata with writedata at this byte. • If byte-enable is low, compare readdata with inverted writedata at this byte.
0x1F8	TG_NUM_DATA_GEN	5	1	Read Only	Data/Byte-Enable Pattern	Number of data generators in the design.
0x1FC	TG_NUM_BYTEEN_GEN	5	1	Read Only	Data/Byte-Enable Pattern	Number of byte-enable generators in the design.
0x200	TG_RDATA_WIDTH	32	1	Read Only	Data/Byte-Enable Pattern	Width of read_data, write_data, and PNF signals.

continued...

Symbol Address	Register Name	Register Width	Number of Registers	Readable or Writeable	Register Section	Register Description
0x204	TG_ERROR_REPORT	32	1	Read Only	Status	Reports illegal configurations of the traffic generator. Value is 0 when no error is present. (See details about error codes below.)
0x208	TG_DATA_RATE_WIDTH_RATIO	4	1	Read Only	Data/Byte-Enable Pattern	Data rate width ratio is the ratio between the data width at the ctrl_amm interface and the data width at the memory interface.
0x240	TG_PNF	32	ceil(TG_RDATA_WIDTH/32)	Read Only	Status	Persistent Pass Not Fail (PNF) signal. Bus Width = TG_RDATA_WIDTH.
0x340	TG_FAIL_EXPECTED_DATA	32	ceil(TG_RDATA_WIDTH/32)	Read Only	Status	The expected data on the first failure. Bus Width = TG_RDATA_WIDTH. (See details below.)
0x440	TG_FAIL_READ_DATA	32	ceil(TG_RDATA_WIDTH/32)	Read Only	Status	The received data on the first failure. Bus Width = TG_RDATA_WIDTH. (See details below.)
0x540	TG_DATA_SEED	32	TG_NUM_DATA_GEN	Readable and Writeable	Data/Byte-Enable Pattern	Seed or starting value for each data generator (DG). This consists of TG_NUM_DATA_GEN entries. To set the seed value for the first DG[0], use the specified symbol address. For DG[1], increment the symbol address by 4. For DG[2], increment it by 8, etc.
0x580	TG_BYTEEN_SEED	32	TG_NUM_BYTEEN_GEN	Readable and Writeable	Data/Byte-Enable Pattern	Seed or starting value for each byte-enable generator (BEG). This consists of TG_NUM_BYTEEN_GEN entries. To set the seed value for the first BEG[0], use the specified symbol address. For BEG[1], increment the symbol address by 4. For BEG[2], increment it by 8, etc.
0x5C0	TG_PPPG_SEL	6	TG_NUM_DATA_GEN	Readable and Writeable	Data/Byte-Enable Pattern	Select pattern for a Data Generator. This consists of TG_NUM_DATA_GEN entries. Select from the available pattern modes: 0: Fixed. 1: PRBS7 2: PRBS15 3: PRBS31 4: Rotating

continued...

Symbol Address	Register Name	Register Width	Number of Registers	Readable or Writeable	Register Section	Register Description
0x600	TG_BYTEEN_SEL	6	TG_NUM_BYTEEN_GEN	Readable and Writeable	Data/Byte-Enable Pattern	Select pattern for a Byte-Enable Generator. This consists of TG_NUM_BYTEEN_GEN. Select from the available pattern modes: 0: Fixed. 1: PRBS7 2: PRBS15 3: PRBS31 4: Rotating
0x640	TG_ADDR_FIELD_RELATIVE_FREQ	16	6	Readable and Writeable	Address Pattern	Frequency setting for both reads and writes. Consists of 6 registers, where each register specifies after how many read or write operations an address field generates a new address. To understand relative frequencies of address fields, refer to *Address Generator Relative Frequencies*.
0x680	TG_ADDR_FIELD_MSB_INDEX	6	5	Readable and Writeable	Address Pattern	Most significant bit (MSB) position setting for both reads and writes. Consists of 5 registers, where each register specifies the index of the MSB of the corresponding field. This ties each of the 6 address generators to a bit range of the generated address. Field number 5 is implied to have an MSB index of AMM_WORD_ADDRESS_WIDTH-1 and need not be specified. The width of an address field is derived from these MSB indices. To understand MSB indices of address fields, refer to *Address Generator MSB Indices*.
0x6C0	TG_BURSTLENGTH_OVERFLOW_OCCURRED	1	1	Read Only	Status	A value of 1 indicates that an attempt was made to write outside of the address space. This occurs when the current address plus the burst length is greater than the total address space. This is an invalid operation and the burst length is clipped to prevent an invalid operation on the Avalon interface.
0x700	TG_BURSTLENGTH_FAIL_ADDR_L	32	1	Read Only	Status	The address at which the burst length overflow was attempted (lower 32 bits). The value at this register is

continued...

Symbol Address	Register Name	Register Width	Number of Registers	Readable or Writeable	Register Section	Register Description
						valid only if TG_BURSTLENGTH_OVERFLOW_OCCURED is 1.
0x704	TG_BURSTLENGTH_FAIL_ADDR_H	32	1	Read Only	Status	The address at which the burst length overflow was attempted (upper 32 bits). The value at this register is valid only if TG_BURSTLENGTH_OVERFLOW_OCCURED is 1.
0x740	TG_WORM_MODE_TARGETTED_DATA	32	ceil(TG_RDATA_WIDTH/32)	Readable	Status Checker	Received data from the targeted read. Targeted read data is set when WORM mode is enabled and the result of the second read to the first fail address occurs. Bus Width = TG_RDATA_WIDTH.

In the table above, some configuration settings and status information can fit within one 32-bit register, while others are broken into several registers. The *Starting Address* column indicates the address of the first register in the set while the *Number of Registers* column indicates the number of registers located after the start address.

For example: TG_PPPG_SEL occupies 8 registers when TG_NUM_DATA_GEN=8, so the data can be accessed by reading from addresses 0x5C0, 0x5C4, … 0x5E0.

13.9.5. User Pattern

The following topics describe available traffic patterns.

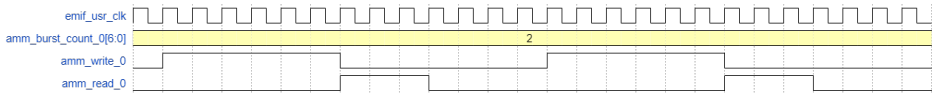
13.9.5.1. Test Duration / Instruction pattern

In the traffic generator, a loop refers to a set of writes followed by a set of reads.

Example test pattern:

```
TG_LOOP_COUNT=2     TG_WRITE_REPEAT_COUNT=1     TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=3     TG_READ_REPEAT_COUNT=1     TG_RW_GEN_LOOP_IDLE_COUNT=4
TG_READ_COUNT=3     TG_BURST_LENGTH=2
```

Figure 171. Timing Diagram for Example Test Pattern



13.9.5.2. Address Pattern

The traffic generator generates addresses based on a configured pattern: An address is generated for each unique write instruction, and then the same address is used for the corresponding unique read instruction. Repeated writes and reads reuse the last unique address.

An address generator occupies a user configurable range of bits and is assigned a user configurable mode. There is a maximum of 6 address generators available. The address pattern is configured by specifying modes, positions, and relative frequencies for 6 address generators.

13.9.5.2.1. Address Generator Modes

Each of the six address generators can be configured to one of four modes.

- **Fixed**: The address generator holds a constant value that is specified in the field's corresponding TG_SEQ_START_ADDR_WR and TG_SEQ_START_ADDR_RD registers.
- **Random**: The address generator starts at the value of the corresponding field's TG_SEQ_START_ADDR_WR and TG_SEQ_START_ADDR_RD registers and generates a pseudorandom address for each instruction. (The pseudorandom address is generated by a Linear Feedback Shift Register (LFSR) and is guaranteed to not repeat for the entirety of the address generator width.)
- **Sequential**: The address generator starts at the value of the corresponding field's TG_SEQ_START_ADDR_WR and TG_SEQ_START_ADDR_RD registers and increments by the value specified in the corresponding field's TG_SEQ_ADDR_INCR register each instruction.
- **Unused**: The address generator is deactivated and does not generate addresses. The address generator output is tied off to zero. Setting a field to this mode allows fewer than 6 address generators to be used.

Note: If a field is set to Random mode, the Start Address cannot be set to all 1s.

13.9.5.2.2. Address Generator MSB Indices

You can specify a most significant bit (MSB) index with the TG_ADDR_FIELD_MSB_INDEX registers, to tie each address generator to a bit range in the generated address.

Because the MSB index for the uppermost field is implied to be at the MSB of the overall address, it is automatically assigned to AMM_WORD_ADDRESS_WIDTH-1 and does not have a configuration register. Writing to the word address (TG_ADDR_FIELD_MSB_INDEX + *n*) specifies the MSB index for field *n*. The system derives the address field widths from the values of the TG_ADDR_FIELD_MSB_INDEX registers. The difference between a field's MSB index setting and the previous field's MSB index setting is the field width.

For example, if field 0 has an MSB index of 5 and field 1 has an MSB index of 9, field 0 will span bits 0-5 (inclusive) and field 1 will span bits 6-9 (inclusive), giving field 0 a width of 6 and field 1 a width of 4.

13.9.5.2.3. Address Generator Effective Width

The effective address width is the number of address bits that are controlled by the 6 address generators.

The effective address width is limited by three parameters and can be calculated as follows:

```
effective_width = wordAddrWidth - log2(wordAddrDivBy) - ceil_log2(burstlength)
```

Where:

- **wordAddrWidth** is the word address width on the ctrl_amm interface.
- **wordAddrDivBy** is the smallest value by which the address on the ctrl_amm interface is divisible to meet the alignment requirement for AMM word address. Generated word address must be divisible by this value. For a half rate (HR) EMIF IP instance without data masking enabled, wordAddrDivBy is 2. In all other cases it is 1.
- **burstlength** is the value that you specify in TG_BURST_LENGTH. If not divisible by 2, the ceiling of the log is taken.

Some of the least significant bits (LSBs) of the overall generated address are used implicitly due to AMM protocol requirements. As a result, these LSBs must be tied to zero, which imposes a restriction on the width of field 0 of the address:

```
field0Width >= log2(wordAddrDivBy) + ceil_log2(burstlength)
```

If this restriction is not met, the appropriate bit in TG_ERROR_REPORT is set to 1 and data mismatches may occur in the generated traffic pattern (refer to Table 365 on page 475 for information on error codes).

13.9.5.2.4. Address Generator Relative Frequencies

The *relative frequency* of an address field refers to the number of read or write operations for which an address generator maintains a constant output before generating a new value.

The TG_ADDR_FIELD_RELATIVE_FREQ set of registers allow address fields 0 to 5 to have their frequency specified.

You can specify the frequency of a field, *n*, by writing an integer, *k*, to address TG_ADDR_FIELD_RELATIVE_FREQ + *n*, where *n* is the desired field index. This setting causes address generator *n* to output a new value every *k* read or write operations.

For example, writing the value 8 to register TG_ADDR_FIELD_RELATIVE_FREQ+2 specifies that field 2 will generate a new address every 8th read or write operation.

13.9.5.2.5. Address Pattern Examples - Basic Mode

The examples shown in this topic include the generated address on both the Avalon address (`amm_address_0`) and the memory address (`mem_addr`).

The difference in widths between `amm_address_0` and `mem_addr` is based on the number of symbols per word.

The following points apply to the four examples that follow:

- A value of *X* indicates that a register is not used, making its value irrelevant.
- The address width (31) is the SYMBOL ADDRESS, as output from the traffic generator. In the design used for these examples, the AMM_WORD_ADDRESS_WIDTH is 26 bits. To account for this difference, the traffic generator shifts all addresses by the difference (5 bits). The examples below use this shifted address, but the external memory interface does not see this shift on its side of the ctrl_amm interface.
- The provided waveform is only a snippet of the full instruction pattern, to demonstrate the write instructions and the corresponding addresses. Not all read blocks are shown, due to space restrictions.

The width of the Avalon address is based on the following:

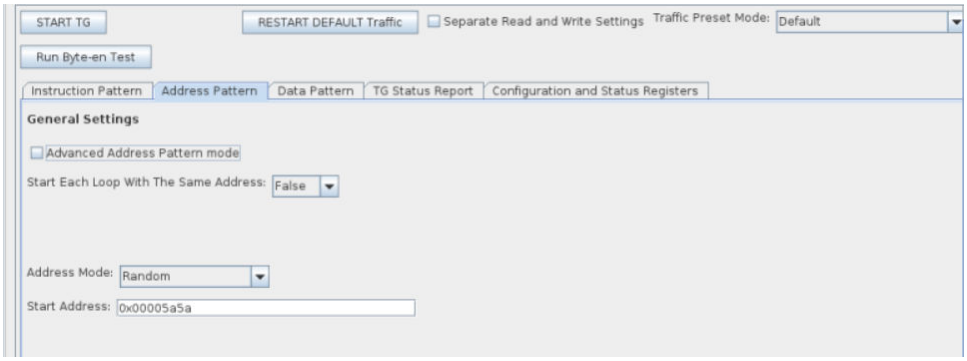
- The data width on the memory side.
- Whether a configured EMIF IP is quarter rate, half rate, or full rate.
- Whether the memory interface is double data rate or quarter data rate.

Example 1: Random Address Mode

Consider the following instruction pattern:

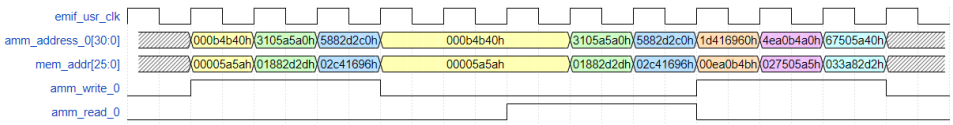
```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=2
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=0
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

Figure 172. Setting the Address Pattern in the Traffic Generator Configuration Interface



This configuration can be performed in basic mode. For the equivalent traffic pattern in advanced mode see example 1 in *Address Pattern Examples - Advanced Mode*.

Figure 173. Random Address Mode

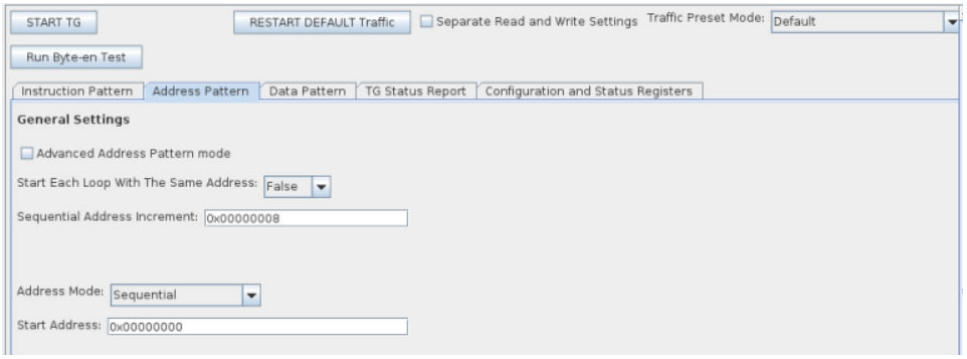


Example 2: Sequential Address Mode

Consider the following instruction pattern:

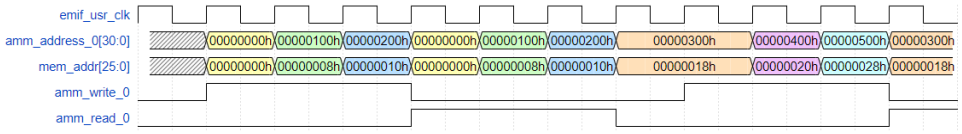
```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=1
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

Figure 174. Setting the Address Pattern in the Traffic Generator Configuration Interface



This configuration can be performed in basic mode. For the equivalent traffic pattern in advanced mode see example 2 in *Address Pattern Examples - Advanced Mode*.

Figure 175. Sequential Address Mode

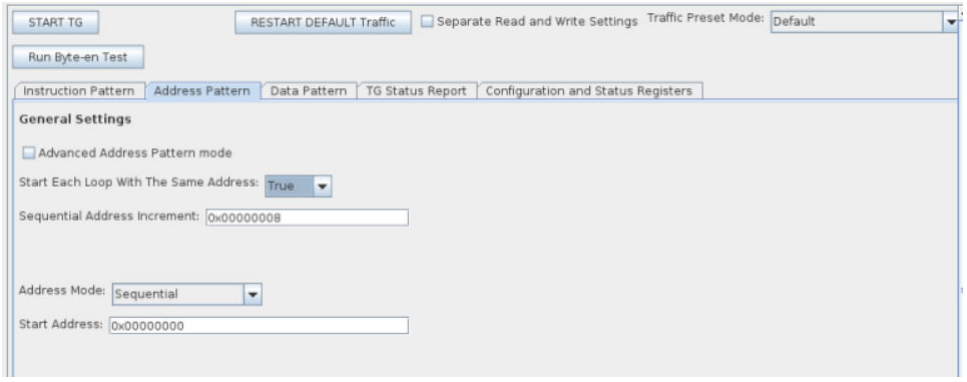


Example 3: Sequential Address Mode with TG_RETURN_TO_START_ADDR=1

Consider the following instruction pattern:

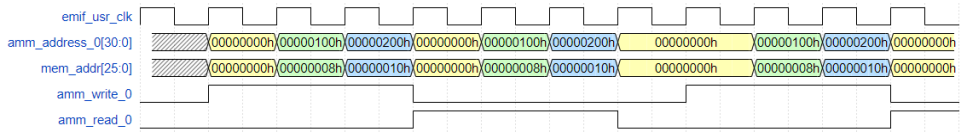
```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=1
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

Figure 176. Setting the Address Pattern in the Traffic Generator Configuration Interface



This configuration can be performed in basic mode. For the equivalent traffic pattern in advanced mode see example 3 in *Address Pattern Examples - Advanced Mode*.

Figure 177. Sequential Address Mode with TG_RETURN_TO_START_ADDR=1

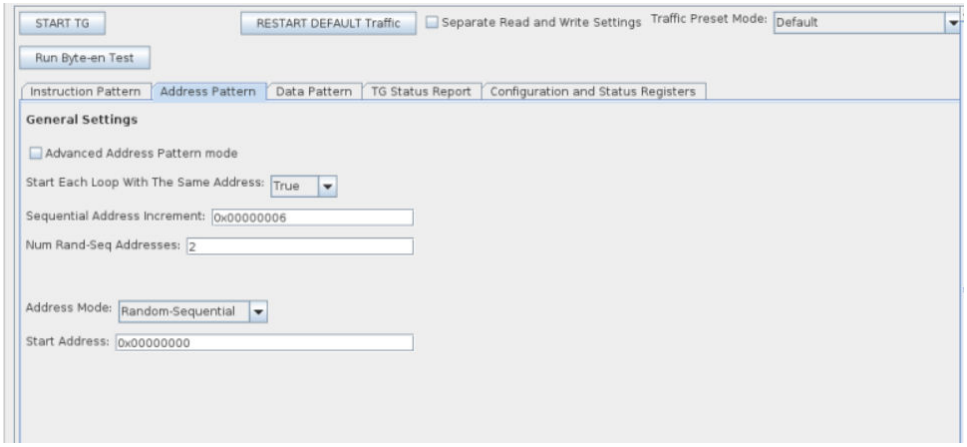


Example 4: Random Sequential Address Mode

Consider the following instruction pattern:

```
TG_LOOP_COUNT=1 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=1
TG_WRITE_COUNT=8 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=1
TG_READ_COUNT=8 TG_BURST_LENGTH=1
```

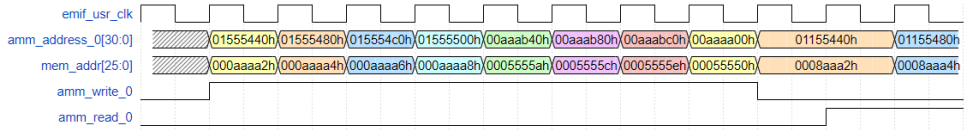
Figure 178. Setting the Address Pattern in the Traffic Generator Configuration Interface



The goal of this address pattern is to create a random sequential pattern with the bottom 4 bits of the address incrementing sequentially by 6 every cycle, while the upper bits are randomly generated every-other cycle.

This configuration can be performed in basic mode. It automatically calculates the value for TG_ADDR_FIELD_MSB_INDEX+1 to ensure that a sufficient number of bits are reserved for field *0* based on the sequential address increment, num rand-seq addresses, burst length, and word-address-divisible-by values. For the equivalent traffic pattern in advanced mode, refer to example 4 in *Address Pattern Examples - Advanced Mode*.

Figure 179. Random-Sequential Address Mode



13.9.5.2.6. Address Pattern Examples - Advanced Mode

The examples in this topic include the generated address on both the Avalon address (`amm_address_0`) and the memory address (`mem_addr`).

The difference in widths between `amm_address_0` and `mem_addr` is based on the configured EMIF IP variant.

The following points apply to the examples that follow:

- A value of *X* indicates that a register is not used, making its value irrelevant.
- The address width (31) is the SYMBOL ADDRESS, as output from the traffic generator. In the design used for these examples, the AMM_WORD_ADDRESS_WIDTH is 26 bits. To account for this difference, the traffic generator shifts all addresses by the difference (5 bits). The examples below use this shifted address, but the external memory interface does not see this shift on its side of the ctrl_amm interface.
- The provided waveform is only a snippet of the full instruction pattern, to demonstrate the write instructions and the corresponding addresses. Not all read blocks are shown, due to space restrictions.

The width of the Avalon address is based on the following:

- The data width on the memory side.
- Whether a configured EMIF IP is quarter rate, half rate, or full rate.
- Whether the memory interface is double data rate or quarter data rate.

Example 1: Random Address Mode

Consider the following instruction pattern:

```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=2
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=0
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

Table 359. Address Pattern

Write Start Addresses: TG_SEQ_START_ADDR_WR =0x5a5a TG_SEQ_START_ADDR_WR+1=0x0000 TG_SEQ_START_ADDR_WR+2=X … TG_SEQ_START_ADDR_WR+11=X	Read Start Addresses: TG_SEQ_START_ADDR_RD =0x5a5a TG_SEQ_START_ADDR_RD+1=0x0000 TG_SEQ_START_ADDR_RD+2=X … TG_SEQ_START_ADDR_RD+11=X
Write Address Modes: TG_ADDR_MODE_WR=1 TG_ADDR_MODE_WR+1=3 … TG_ADDR_MODE_WR+5=3	Read Address Modes: TG_ADDR_MODE_RD=1 TG_ADDR_MODE_RD+1=3 … TG_ADDR_MODE_RD+5=3
Sequential Address Increments: TG_SEQ_ADDR_INCR=X TG_SEQ_ADDR_INCR+1=X … TG_SEQ_ADDR_INCR+5=X	Return to Start Address: TG_RETURN_TO_START_ADDR=0
Relative Frequencies: TG_ADDR_FIELD_RELATIVE_FREQ=1 TG_ADDR_FIELD_RELATIVE_FREQ+1=X … TG_ADDR_FIELD_RELATIVE_FREQ+5=X	MSB Indices: TG_ADDR_FIELD_MSB_INDEX=AMM_WORD_ADDRESS_WIDTH-1 TG_ADDR_FIELD_MSB_INDEX+1=X … TG_ADDR_FIELD_MSB_INDEX+4=X

Figure 180. Setting the Address Pattern in the Traffic Generator Configuration Interface

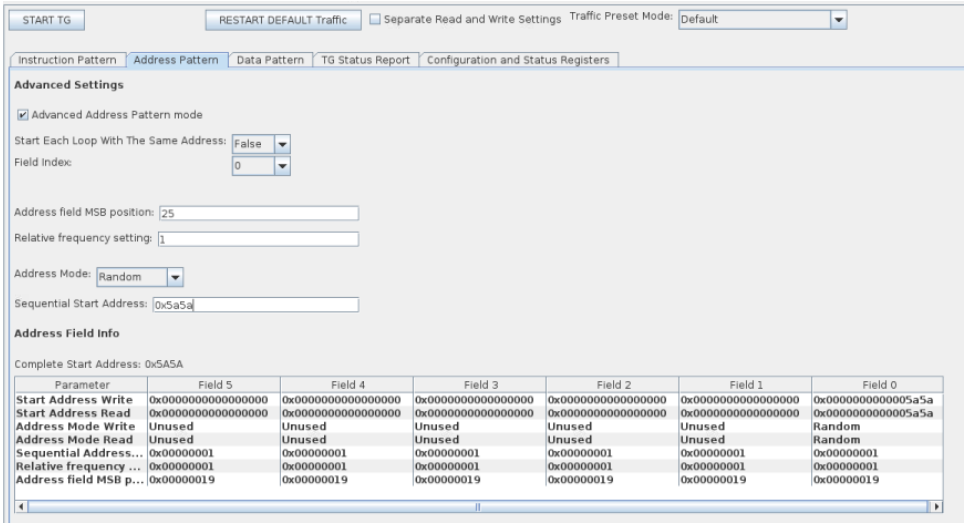
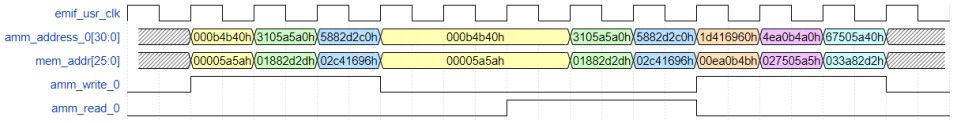


Figure 181. Random Address Mode



Example 2: Sequential Address Mode

Consider the following instruction pattern:

```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=1
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

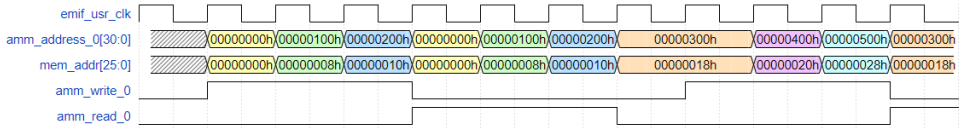
Table 360. Address Pattern

Write Start Addresses: TG_SEQ_START_ADDR_WR =’0 TG_SEQ_START_ADDR_WR+1=’0 TG_SEQ_START_ADDR_WR+2=X … TG_SEQ_START_ADDR_WR+11=X	Read Start Addresses: TG_SEQ_START_ADDR_RD =’0 TG_SEQ_START_ADDR_RD+1=’0 TG_SEQ_START_ADDR_RD+2=X … TG_SEQ_START_ADDR_RD+11=X
Write Address Modes: TG_ADDR_MODE_WR=2 TG_ADDR_MODE_WR+1=3 … TG_ADDR_MODE_WR+5=3	Read Address Modes: TG_ADDR_MODE_RD=2 TG_ADDR_MODE_RD+1=3 … TG_ADDR_MODE_RD+5=3
Sequential Address Increments: TG_SEQ_ADDR_INCR=8 TG_SEQ_ADDR_INCR+1=X … TG_SEQ_ADDR_INCR+5=X	Return to Start Address: TG_RETURN_TO_START_ADDR=0

continued...

Relative Frequencies: TG_ADDR_FIELD_RELATIVE_FREQ=1 TG_ADDR_FIELD_RELATIVE_FREQ+1=X … TG_ADDR_FIELD_RELATIVE_FREQ+5=X	MSB Indices: TG_ADDR_FIELD_MSB_INDEX=AMM_WORD_ADDRESS_WIDTH-1 TG_ADDR_FIELD_MSB_INDEX+1=X … TG_ADDR_FIELD_MSB_INDEX+4=X

Figure 182. Sequential Address Mode



Example 3: Sequential Address Mode with TG_RETURN_TO_START_ADDR_1

Consider the following instruction pattern:

```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=1
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

Table 361. Address Pattern

Write Start Addresses: TG_SEQ_START_ADDR_WR =’0 TG_SEQ_START_ADDR_WR+1=’0 TG_SEQ_START_ADDR_WR+2=X … TG_SEQ_START_ADDR_WR+11=X	Read Start Addresses: TG_SEQ_START_ADDR_RD =’0 TG_SEQ_START_ADDR_RD+1=’0 TG_SEQ_START_ADDR_RD+2=X … TG_SEQ_START_ADDR_RD+11=X
Write Address Modes: TG_ADDR_MODE_WR=2 TG_ADDR_MODE_WR+1=3 … TG_ADDR_MODE_WR+5=3	Read Address Modes: TG_ADDR_MODE_RD=2 TG_ADDR_MODE_RD+1=3 … TG_ADDR_MODE_RD+5=3
Sequential Address Increments: TG_SEQ_ADDR_INCR=8 TG_SEQ_ADDR_INCR+1=X … TG_SEQ_ADDR_INCR+5=X	Return to Start Address: TG_RETURN_TO_START_ADDR=1
Relative Frequencies: TG_ADDR_FIELD_RELATIVE_FREQ=1 TG_ADDR_FIELD_RELATIVE_FREQ+1=X … TG_ADDR_FIELD_RELATIVE_FREQ+5=X	MSB Indices: TG_ADDR_FIELD_MSB_INDEX+1=AMM_WORD_ADDRESS_WIDTH-1 TG_ADDR_FIELD_MSB_INDEX+1=X … TG_ADDR_FIELD_MSB_INDEX+4=X

Figure 183. Setting the Address Pattern in the Traffic Generator Configuration Interface

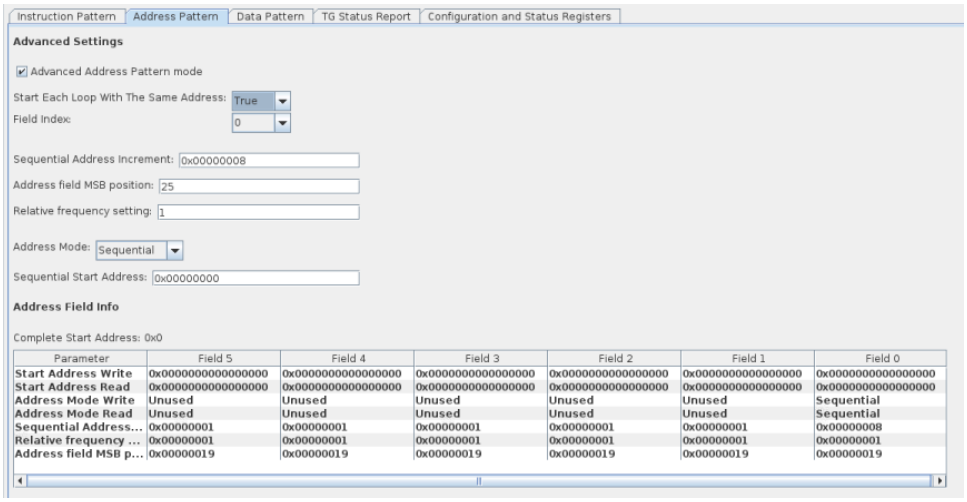
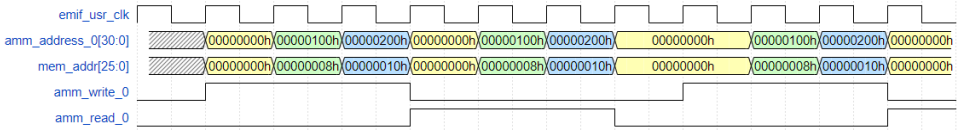


Figure 184. Sequential Address Mode with TG_RETURN_TO_START_ADDR=1



Example 4: Random Sequential Address Mode

Consider the following instruction pattern:

```
TG_LOOP_COUNT=1 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=1
TG_WRITE_COUNT=8 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=1
TG_READ_COUNT=8 TG_BURST_LENGTH=1
```

Table 362. Address Pattern

Write Start Addresses: TG_SEQ_START_ADDR_WR =0x0000 TG_SEQ_START_ADDR_WR+1=0x0000 TG_SEQ_START_ADDR_WR+2=0xaaaa TG_SEQ_START_ADDR_WR+3=0x0000 TG_SEQ_START_ADDR_WR+4=X … TG_SEQ_START_ADDR_WR+11=X	Read Start Addresses: TG_SEQ_START_ADDR_RD =0x0000 TG_SEQ_START_ADDR_RD+1=0x0000 TG_SEQ_START_ADDR_RD+2=0xaaaa TG_SEQ_START_ADDR_RD+3=0x0000 TG_SEQ_START_ADDR_RD+4=X … TG_SEQ_START_ADDR_RD+11=X
Write Address Modes: TG_ADDR_MODE_WR=2 TG_ADDR_MODE_WR+1=1 TG_ADDR_MODE_WR+2=3 … TG_ADDR_MODE_WR+5=3	Read Address Modes: TG_ADDR_MODE_RD=2 TG_ADDR_MODE_RD+1=1 TG_ADDR_MODE_RD+2=3 … TG_ADDR_MODE_RD+5=3
Sequential Address Increments: TG_SEQ_ADDR_INCR=2 TG_SEQ_ADDR_INCR+1=X … TG_SEQ_ADDR_INCR+5=X	Return to Start Address: TG_RETURN_TO_START_ADDR=0

continued...

Relative Frequencies: TG_ADDR_FIELD_RELATIVE_FREQ=1 TG_ADDR_FIELD_RELATIVE_FREQ+1=4 TG_ADDR_FIELD_RELATIVE_FREQ+2=X TG_ADDR_FIELD_RELATIVE_FREQ+3=X TG_ADDR_FIELD_RELATIVE_FREQ+4=X TG_ADDR_FIELD_RELATIVE_FREQ+5=X	MSB Indices: TG_ADDR_FIELD_MSB_INDEX=3 TG_ADDR_FIELD_MSB_INDEX+1=AMM_WORD_ADDRESS_WIDTH-1 TG_ADDR_FIELD_MSB_INDEX+4=X TG_ADDR_FIELD_MSB_INDEX+4=X TG_ADDR_FIELD_MSB_INDEX+4=X

Figure 185. Setting the Address Pattern in the Traffic Generator Configuration Interface

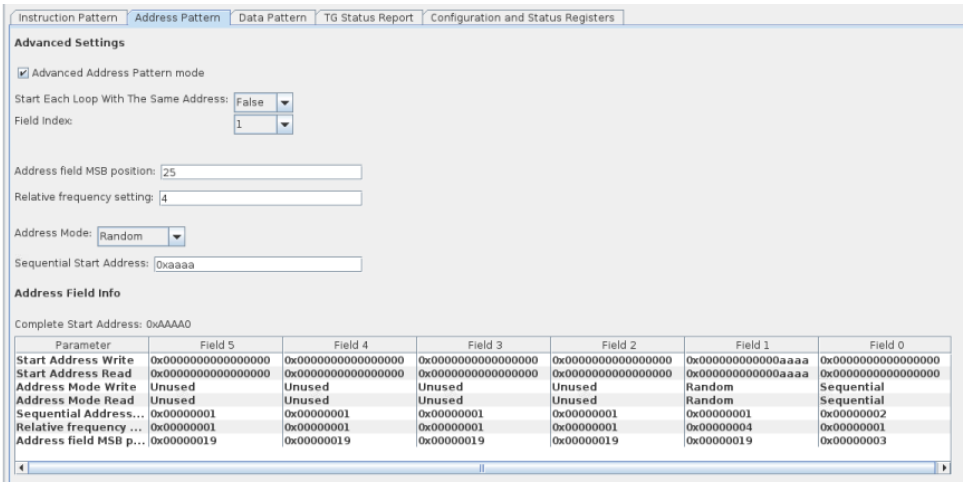
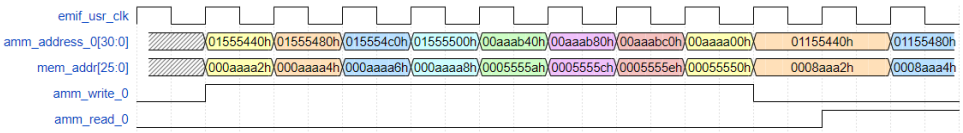


Figure 186. Random-Sequential Address Mode



Example 5: Using Multiple Address Fields for Traversing Memory Heirarchy

Consider the following instruction pattern:

```
TG_LOOP_COUNT=0 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=1 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=0
TG_READ_COUNT=0 TG_BURST_LENGTH=1
```

Address pattern:

This address pattern uses multiple fields to traverse the memory hierarchy to isolate a specific bank group for signal integrity testing. You specify the mapping between the Avalon control interface and the memory address and command interface using the **Address Ordering** parameter on the **Controller** tab of the parameter editor.

For example, consider a quarter-rate EMIF IP variant configured such that CTRL_DDR4_ADDR_ORDER_ENUM = DDR4_CTRL_ADDR_ORDER_CS_R_B_C_BG and the external memory is of the following format:

Table 363.

Hierarchy Level	Parameter	Value of Parameter
Rank	MEM_DDR4_DISCRETE_CS_WIDTH (cs)	1
Bank Group	MEM_DDR4_BANK_GROUP_WIDTH (BG)	2
Bank	MEM_DDR4_BANK_ADDR_WIDTH (BA)	2
Row	MEM_DDR4_ROW_ADDR_WIDTH (R)	15
Column	MEM_DDR4_COL_ADDR_WIDTH (C)	10

For this parameterization, the address bits map to the memory address and command pins on the EMIF ctrl_amm interface as follows:

25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	BA	BA	C	C	C	C	C	C	C	BG	BG

In the above example, the EMIF control interface address only uses 7 column bits. Due to the quarter-rate user logic and the double data rate interface, each instruction on the ctrl_amm interface causes a burst length of 8 on the memory side. We do not explicitly address those 3 bits on the ctrl_amm interface.

In this example the goal is to generate traffic for a randomly chosen bank in bank group 3. To write to every row and column combination in this bank requires $2^{16}x2^{7}=2^{23}$ unique writes on the control interface. Each TG_ADDR_FIELD_RELATIVE_FREQ register is 16 bits wide, meaning that the maximum relative frequency setting is 2^{16}-1. This maximum relative frequency does not allow access to every row and column in a bank, but does allow sufficient random coverage to test the signal integrity of each bank.

```
Write Start Addresses:
TG_SEQ_START_ADDR_WR  =0x0003
TG_SEQ_START_ADDR_WR+1=0x0000
TG_SEQ_START_ADDR_WR+2=0x0000
TG_SEQ_START_ADDR_WR+3=0x0000
TG_SEQ_START_ADDR_WR+4=0x0000
TG_SEQ_START_ADDR_WR+5=0x0000
TG_SEQ_START_ADDR_WR+6=0x0000
TG_SEQ_START_ADDR_WR+7=0x0000
TG_SEQ_START_ADDR_WR+8=X
        …
TG_SEQ_START_ADDR_WR+11=X
```

```
Read Start Addresses:
TG_SEQ_START_ADDR_RD  =0x0003
TG_SEQ_START_ADDR_RD+1=0x0000
TG_SEQ_START_ADDR_RD+2=0x0000
TG_SEQ_START_ADDR_RD+3=0x0000
TG_SEQ_START_ADDR_RD+4=0x0000
TG_SEQ_START_ADDR_RD+5=0x0000
TG_SEQ_START_ADDR_RD+6=0x0000
TG_SEQ_START_ADDR_RD+7=0x0000
TG_SEQ_START_ADDR_RD+8=X
        …
TG_SEQ_START_ADDR_RD+11=X
```

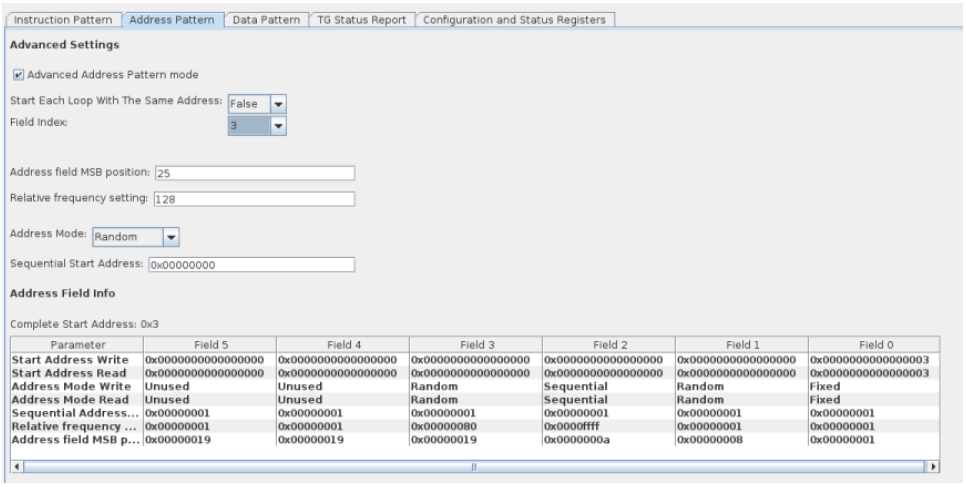
```
Write Address Modes:
TG_ADDR_MODE_WR=0
TG_ADDR_MODE_WR+1=2
TG_ADDR_MODE_WR+2=1
TG_ADDR_MODE_WR+3=2
TG_ADDR_MODE_WR+4=3
TG_ADDR_MODE_WR+5=3
```

```
Read Address Modes:
TG_ADDR_MODE_RD=0
TG_ADDR_MODE_RD+1=2
TG_ADDR_MODE_RD+2=1
TG_ADDR_MODE_RD+3=2
TG_ADDR_MODE_RD+4=3
TG_ADDR_MODE_RD+5=3
```

continued...

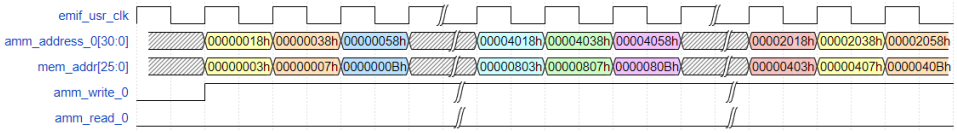
Sequential Address Increments: TG_SEQ_ADDR_INCR=X TG_SEQ_ADDR_INCR+1=1 TG_SEQ_ADDR_INCR+2=X TG_SEQ_ADDR_INCR+3=1 TG_SEQ_ADDR_INCR+4=X TG_SEQ_ADDR_INCR+5=X	Return to Start Address: TG_RETURN_TO_START_ADDR=0
Relative Frequencies: TG_ADDR_FIELD_RELATIVE_FREQ=1 TG_ADDR_FIELD_RELATIVE_FREQ+1=1 TG_ADDR_FIELD_RELATIVE_FREQ+2=216-1 TG_ADDR_FIELD_RELATIVE_FREQ+3=27 TG_ADDR_FIELD_RELATIVE_FREQ+4=X TG_ADDR_FIELD_RELATIVE_FREQ+5=X	MSB Indices: TG_ADDR_FIELD_MSB_INDEX=1 TG_ADDR_FIELD_MSB_INDEX+1=8 TG_ADDR_FIELD_MSB_INDEX+2=10 TG_ADDR_FIELD_MSB_INDEX+3=AMM_WORD_ADDRESS_WIDTH-1 TG_ADDR_FIELD_MSB_INDEX+4=X

Figure 187. Setting this Address Pattern Configuration in the Traffic Generator Configuration Interface



This address pattern can be performed in advanced mode only.

Figure 188. Multiple Address Fields for Traversing Memory Hierarchy



The first three writes represent the start of traffic where field 1 is incrementing every cycle. The first write after the time skip represents write number 2^7, as this is the first time that field 3 is incremented by 1 due to a relative frequency setting of 2^7. The first write after the second time skip represents write number 2^{16}, as this is the first time that a new value is generated for field 2 due to a relative frequency setting of 2^{16}-1

Example 6: Using All Address Fields

Consider the following instruction pattern:

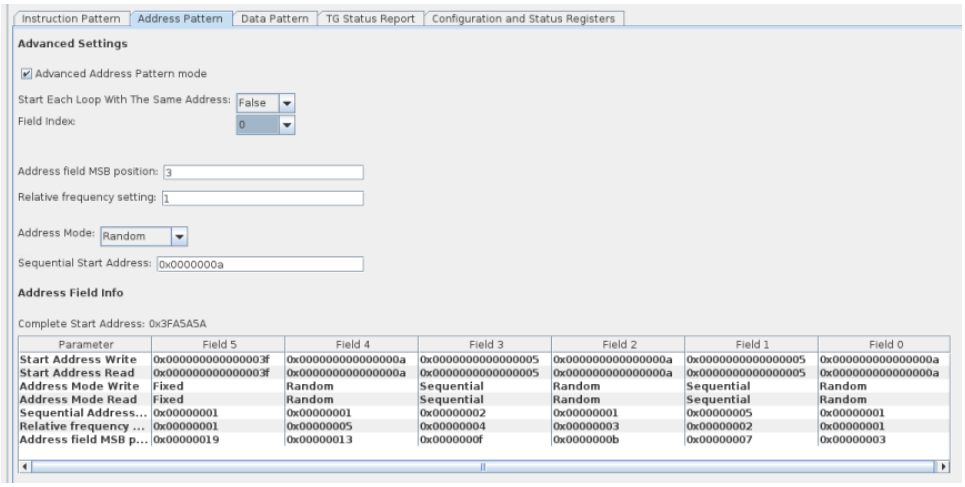
```
TG_LOOP_COUNT=2 TG_WRITE_REPEAT_COUNT=1 TG_RW_GEN_IDLE_COUNT=0
TG_WRITE_COUNT=3 TG_READ_REPEAT_COUNT=1 TG_RW_GEN_LOOP_IDLE_COUNT=0
TG_READ_COUNT=3 TG_BURST_LENGTH=1
```

Address pattern:

This address pattern illustrates the abilities of the advanced mode, by tying different address bits to a variety of different address generators, each with a different relative frequency.

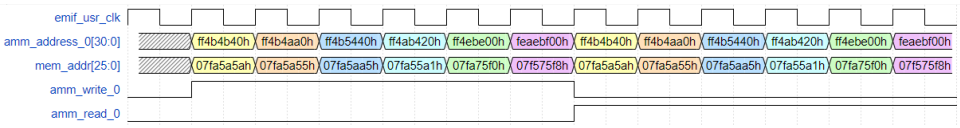
Write Start Addresses: TG_SEQ_START_ADDR_WR =0x000a TG_SEQ_START_ADDR_WR+1=0x0000 TG_SEQ_START_ADDR_WR+2=0x0005 TG_SEQ_START_ADDR_WR+3=0x0000 TG_SEQ_START_ADDR_WR+4=0x000a TG_SEQ_START_ADDR_WR+5=0x0000 TG_SEQ_START_ADDR_WR+6=0x0005 TG_SEQ_START_ADDR_WR+7=0x0000 TG_SEQ_START_ADDR_WR+8=0x000a TG_SEQ_START_ADDR_WR+9=0x0000 TG_SEQ_START_ADDR_WR+10=0x007f TG_SEQ_START_ADDR_WR+11=0x0000	Read Start Addresses: TG_SEQ_START_ADDR_RD =0x000a TG_SEQ_START_ADDR_RD+1=0x0000 TG_SEQ_START_ADDR_RD+2=0x0005 TG_SEQ_START_ADDR_RD+3=0x0000 TG_SEQ_START_ADDR_RD+4=0x000a TG_SEQ_START_ADDR_RD+5=0x0000 TG_SEQ_START_ADDR_RD+6=0x0005 TG_SEQ_START_ADDR_RD+7=0x0000 TG_SEQ_START_ADDR_RD+8=0x000a TG_SEQ_START_ADDR_RD+9=0x0000 TG_SEQ_START_ADDR_RD+10=0x007f TG_SEQ_START_ADDR_RD+11=0x0000
Write Address Modes: TG_ADDR_MODE_WR=1 TG_ADDR_MODE_WR+1=2 TG_ADDR_MODE_WR+2=1 TG_ADDR_MODE_WR+3=2 TG_ADDR_MODE_WR+4=1 TG_ADDR_MODE_WR+5=0	Read Address Modes: TG_ADDR_MODE_RD=1 TG_ADDR_MODE_RD+1=2 TG_ADDR_MODE_RD+2=1 TG_ADDR_MODE_RD+3=2 TG_ADDR_MODE_RD+4=1 TG_ADDR_MODE_RD+5=0
Sequential Address Increments: TG_SEQ_ADDR_INCR=X TG_SEQ_ADDR_INCR+1=5 TG_SEQ_ADDR_INCR+2=X TG_SEQ_ADDR_INCR+3=2 TG_SEQ_ADDR_INCR+4=X TG_SEQ_ADDR_INCR+5=X	Return to Start Address: TG_RETURN_TO_START_ADDR=0
Relative Frequencies: TG_ADDR_FIELD_RELATIVE_FREQ=1 TG_ADDR_FIELD_RELATIVE_FREQ+1=2 TG_ADDR_FIELD_RELATIVE_FREQ+2=3 TG_ADDR_FIELD_RELATIVE_FREQ+3=4 TG_ADDR_FIELD_RELATIVE_FREQ+4=5 TG_ADDR_FIELD_RELATIVE_FREQ+5=X	MSB Indices: TG_ADDR_FIELD_MSB_INDEX=3 TG_ADDR_FIELD_MSB_INDEX+1=7 TG_ADDR_FIELD_MSB_INDEX+2=11 TG_ADDR_FIELD_MSB_INDEX+3=15 TG_ADDR_FIELD_MSB_INDEX+4=19

Figure 189. How to set this address pattern configuration in the Traffic Generator Configuration Interface



This address pattern can be performed in advanced mode only.

Figure 190. Multiple Address Fields



13.9.5.3. Data Pattern and Byte Enable

An independent data generator controls each DQ pin's pattern within each DQS group. The pattern is then duplicated across DQS groups.

Example:

This example assumes the following conditions:

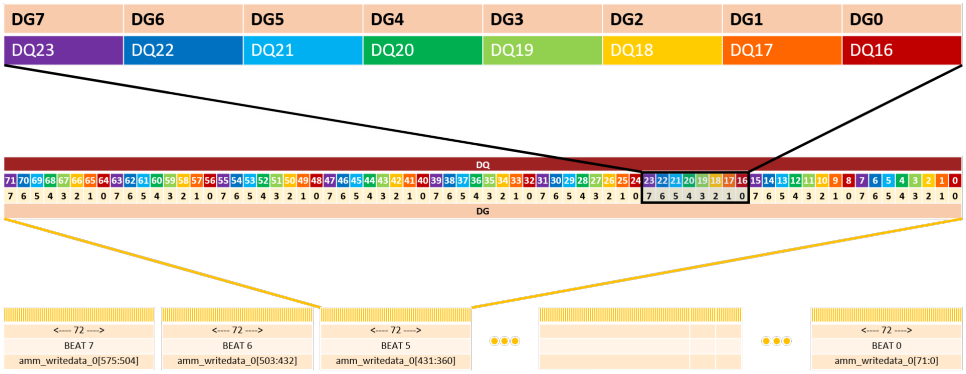
- DQ Width: x72 (without ECC enabled)
- Memory Protocol: DDR4
- DQ/DQS: 8
- Rate: Quarter-rate

Because there are 8 DQ/DQS, there are 8 data generators, meaning that DQ0, DQ8, DQ16 … DQ56, and DQ64 share the same data generator.

In this example, each data transfer on the ctrl_amm interface is 576 bits wide. For the purpose of this example, let's focus on beat 5 of the transfer, as seen on the memory side:

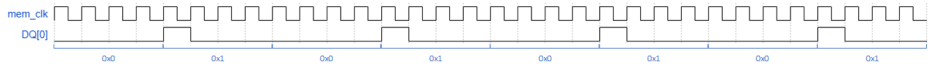
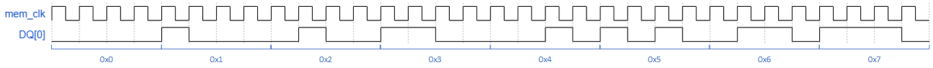
- DQ[0] corresponds to beat 5 of the memory bus burst and it takes bit5 from DG0
- DQ[1] corresponds to beat 5 of the memory bus burst and it takes bit5 from DG1
- ...
- DQ[3] corresponds to beat 5 of the memory bus burst and it takes bit5 from DG3
- ...
- DQ[8] corresponds to beat 5 of the memory bus burst and it takes bit5 from DG0
- DQ[9] corresponds to beat 5 of the memory bus burst and it takes bit5 from DG1

Figure 191. Data Pattern Example



For each data generator, it is possible to configure a start seed (32 bits wide), and a pattern mode.

Table 364. Modes

Mode	Description
Fixed	The data pattern is constant. Only the lowest TG_DATA_RATE_WIDTH_RATIO bits are used – each one representing one beat on that pin. Example: A seed of 0x76543210 for DG0, where TG_DATA_RATE_WIDTH_RATIO=8 results in the following pattern seen on DQ0, sequentially: 
PRBS7	Pseudorandom Binary Sequence. Uses the 8 least-significant bits (LSB) of the input seed, and the monic polynomial: $x^7 + x^6 + 1$ A seed of '0 produces no unique patterns.
PRBS15	Pseudorandom Binary Sequence. Uses the 16 least-significant bits of the input seed, and the monic polynomial: $x^{15} + x^{14} + 1$ A seed of '0 produces no unique patterns.
PRBS31	Pseudorandom Binary Sequence. Uses all 32 bits of the input seed, and the monic polynomial: $x^{31} + x^{28} + 1$ A seed of '0 produces no unique patterns.
Rotating (custom)	The data pattern on the pin is 32 bits long, as specified by you. The pattern appears on the pin least-significant bit to most-significant bit. Example: A seed of 0x76543210 for DG0 results in the following pattern seen on DQ0, sequentially: 

There is one byte-enable generator for each byte in the interface. The byte-enable generator options are identical to the data generator options.

13.9.6. Traffic Generator Status

The traffic generator reports its status in two ways: ISSPs and configuration interface.

Status Registers

The traffic generator reports status in two ways:

- ISSPs
- Configuration interface

Reading PNF Registers or ISSPs

The Pass Not Fail (PNF) registers show the status for each bit that has been read on the ctrl_amm interface.

```
pnf[x] = ~(amm_readdata_0[x]^amm_expected_readdata_0[x])
```

The PNF signal is "sticky", which means that once a PNF bit is set to 0 due to a read miscompare, it does not return to a value of 1 on any consecutive reads, until the PNF is cleared. The PNF, TG_FAIL_EXPECTED_DATA, and TG_FAIL_READ_DATA registers are normally wider than a single register on the tg_cfg interface. As a result, the bus width is split up across NPNF_reg registers, where:

$$N_{PNF\_reg} = ceil(TG\_RDATA\_WIDTH / 32)$$

To determine the address of the *N*th register:

$$TG\_PNF[N_{PNF\_reg}] = (Symbol\ Address\ of\ TG\_PNF) + 4*N_{PNF\_reg}$$

The maximum PNF width is 511 bits, so the bus width is split up across N_{PNF_ISSP} ISSPs:

$$N_{PNG\_ISSP} = ceil(TG\_RDATA\_WIDTH / 511)$$

PNF ISSPs have the index in their name, starting with PNF0.

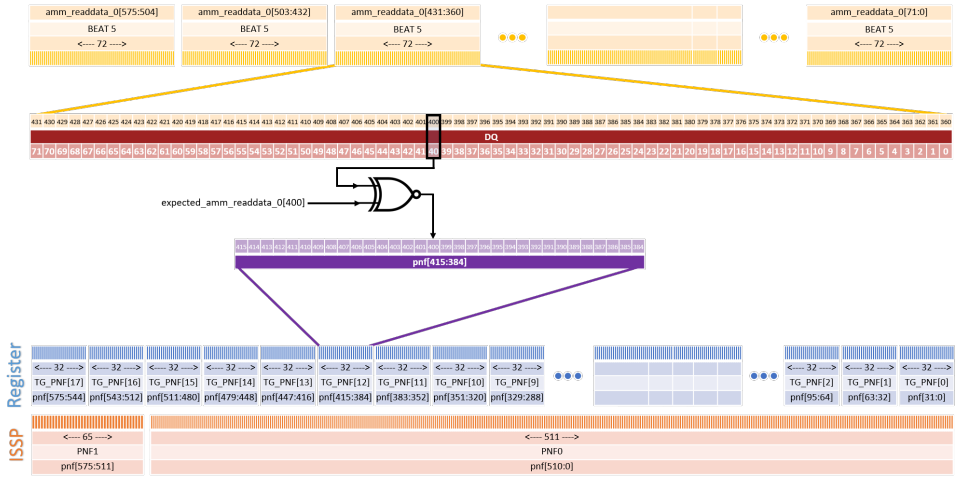
Example:

This example assumes the following conditions:

- DQ Width: x72 (without ECC enabled)
- Memory Protocol: DDR4
- DQ/DQS: 8
- Rate: Quarter-rate

Due to the memory protocol and rate, TG_DATA_RATE_WIDTH_RATIO = 8.

Therefore TG_RDATA_WIDTH = 72 * 8 = 576. This means that there are 18 TG_PNF registers, and two PNF ISSPs (PNF0, PNF1), as illustrated below:



Clearing Failure Information Between Runs

You can set the TG_CLEAR register to clear failure information between successive runs of the traffic generator.

Configuration Error Codes

The TG_ERROR_REPORT register codes flag when the traffic pattern may be the direct cause for data mismatches. You may still choose to run the traffic pattern, despite errors. This may be useful when not looking for a passing comparison.

The following table outlines the error codes:

Table 365. Error Codes

Code Value	Code Name	Code Description
0x1	ERR_MORE_READS_THAN_WRITES	More read operations are scheduled than write operations. Data mismatches might occur.
0x2	ERR_BURSTLENGTH_GT_SEQ_ADDR_INCR	The Avalon burst length exceeds the sequential address increment. Data mismatches might occur.
0x4	ERR_ADDR_DIVISIBLE_BY_GT_SEQ_ADDR_INCR	The sequential address increment is smaller than the minimum required. Data mismatches might occur.
0x8	ERR_SEQ_ADDR_INCR_NOT_DIVISIBLE	The sequential address increment is not divisible by the necessary step. Data mismatches might occur.
0x10	ERR_READ_AND_WRITE_START_ADDRS_DIFFER	The sequential address increment is not divisible by the necessary step. Data mismatches might occur.
0x20	ERR_ADDR_MODES_DIFFERENT	Read and write settings for address generation modes are different. Data mismatches might occur.
0x40	ERR_REPEATS_SET_TO_ZERO	Invalid read or write repeat count. The number of read or write repeats must be at least 1. Data mismatches might occur.
0x80	ERR_BOTH_BURST_AND_REPEAT_MODE_ACTIVE	Invalid read or write repeat count. The number of read or write repeats must be at least 1. Data mismatches might occur.
0x100	ERR_BURSTLENGTH_AND_WORD_ADDR_DIVISIBLE_FIELD_0	The width of field 0 must be greater than the number of bits required for burst length plus the number of LSB bits for the smallest possible step. Data mismatches may occur.
0x200	ERR_RELATIVE_FREQ_ZERO	The relative frequency of a field is zero. Data mismatches may occur.

13.9.7. Starting Traffic with the Traffic Generator

You can signal the traffic generator to start traffic in a variety of ways, which are described below.

Default Traffic

If you select the **Enable default traffic pattern** parameter when you parameterize the design example, the default traffic pattern begins automatically when the traffic generator comes out of the reset state.

To trigger the same traffic manually after this point, you can simply reset the traffic generator, or issue a write command to the TG_RESTART_DEFAULT_TRAFFIC register (symbol address 0xB0).

User Traffic

To launch traffic in user mode, issue a write to the TG_START register (symbol address 0x4).

To run user mode simulation with a custom traffic pattern, edit the `tg_def_sim_master_user_param` parameter in `altera_emif_avl_tg_2_sim_master_defs.sv`. Ensure that a value of 1 is issued to TG_START (symbol address 0x4) at the end of the pattern, to start traffic.

You can run user traffic infinitely by writing a value of 0 to TG_LOOP_COUNT (symbol address 0x8) and a value of 1 to TG_START (symbol address 0x4). To stop user traffic when it is running infinitely, write a non-zero value to TG_LOOP_COUNT (symbol address 0x8).

Write Once Read Many (WORM) Mode

In WORM mode, if a data mismatch is encountered, the traffic generator stops at the first data mismatch and issues a second read to the same address. The purpose of this mode is to distinguish between write failures and read failures. You can enable this mode with either default traffic or user traffic.

To enable WORM mode, write a value of 1 to TG_USER_WORM_EN (symbol address 0xB4) or enable ISSPs in the design and write a value of 1 to the WORM in-system source.

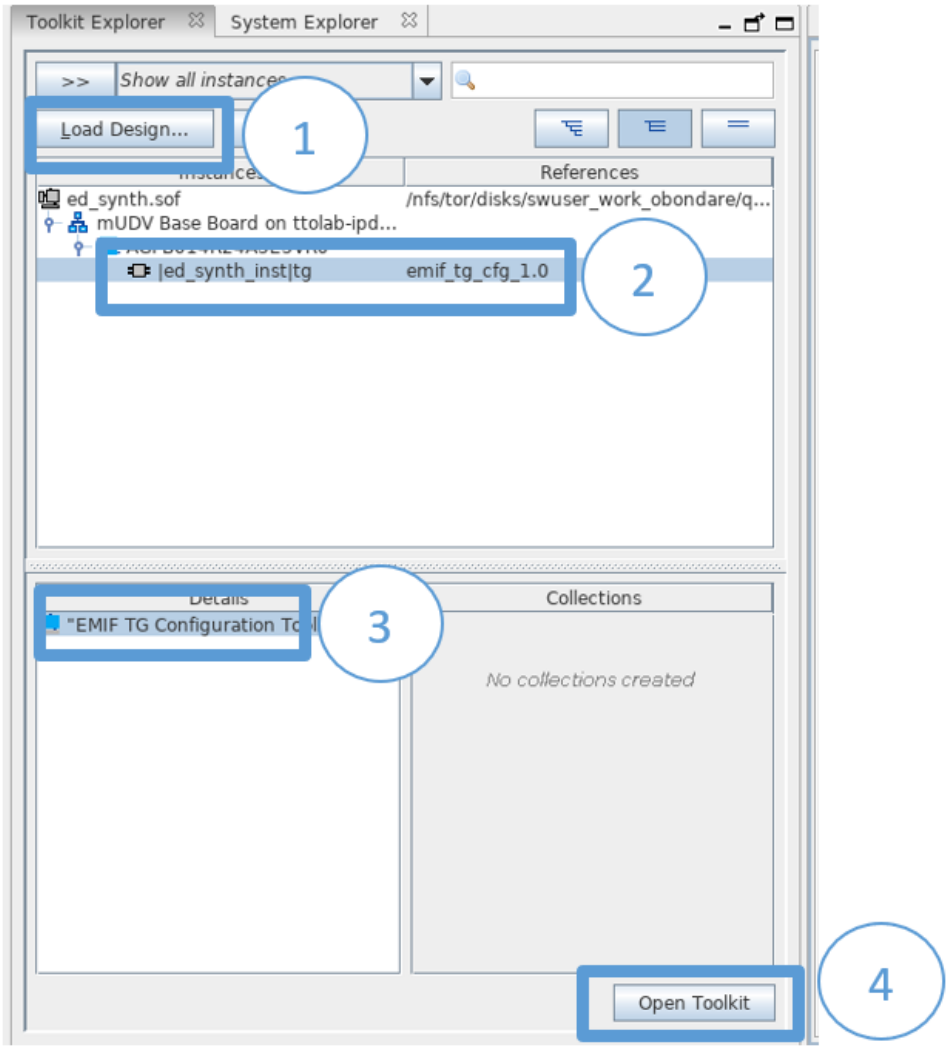
13.9.8. Traffic Generator Configuration User Interface

The following topics describe the traffic generator user interface.

13.9.8.1. Connecting the Traffic Generator

1. In the Intel Quartus Prime software, open the System Console by clicking **Tools ➤ System Debugging Tools ➤ System Console**. In the System Console, load the SRAM Object File (.sof) with which you programmed the board.
2. Select the `emif_tg_cfg` toolkit instance.
3. Select EMIF TG Configuration Toolkit.
4. Click **Open Toolkit** to launch the toolkit.

Figure 192. Connecting the Traffic Generator



13.9.8.2. Configuring the Traffic Generator

Set all the required parameters on the **Instruction Pattern**, **Address Pattern**, and **Data Pattern** tabs in the traffic generator interface. For information about how each setting affects the final traffic pattern, refer to the User-Configured Traffic Pattern section.

Figure 193. Instruction Pattern Tab

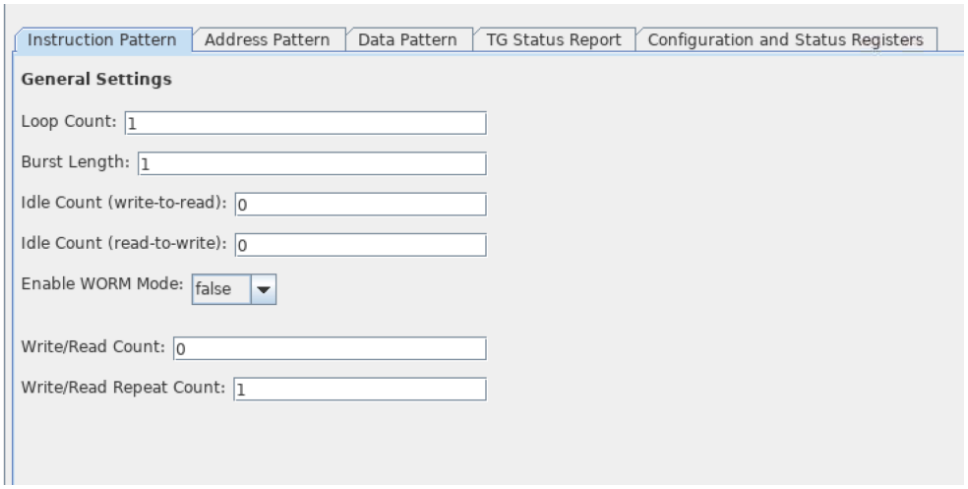


Figure 194. Instruction Pattern tab: Separate Read and Write Settings

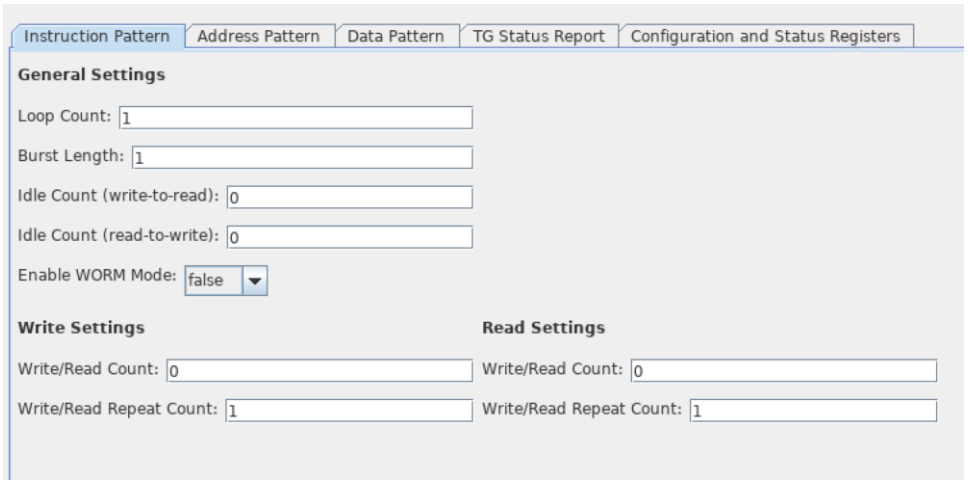


Figure 195. Address Pattern Tab – Basic Mode

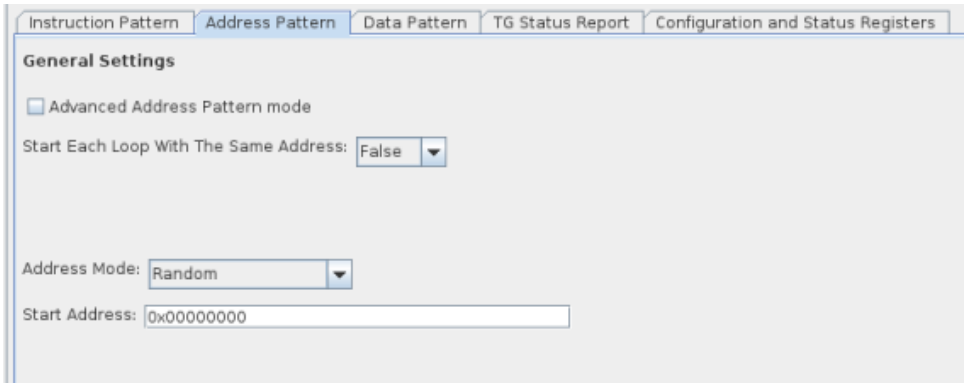
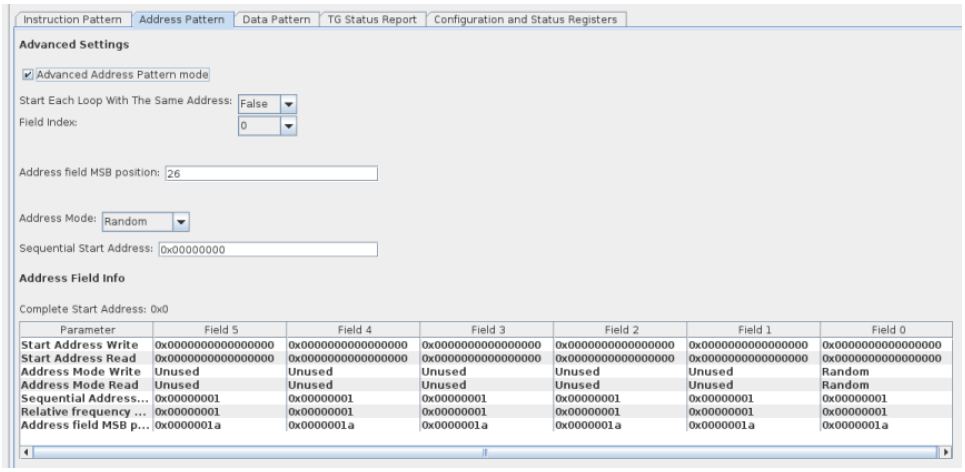


Figure 196. Address Pattern Tab – Advanced Mode



The address pattern tab can be viewed in Basic Mode or Advanced Mode. Basic Mode presents a simple method of address generation where the specified pattern affects the full address width. Basic mode lets you generate addresses randomly, sequentially, or random-sequentially from a given starting address, with a given address increment. Conversely, Advanced Mode lets you specify six independent patterns for different portions of the address, by selecting a field index to configure from the drop-down menu, and then setting the MSB position, address mode, start address, and relative frequency. Basic Mode is a subset of Advanced Mode, and the same configurations apply.

Figure 197. Address Pattern tab: Separate Read and Write Settings – Basic Mode

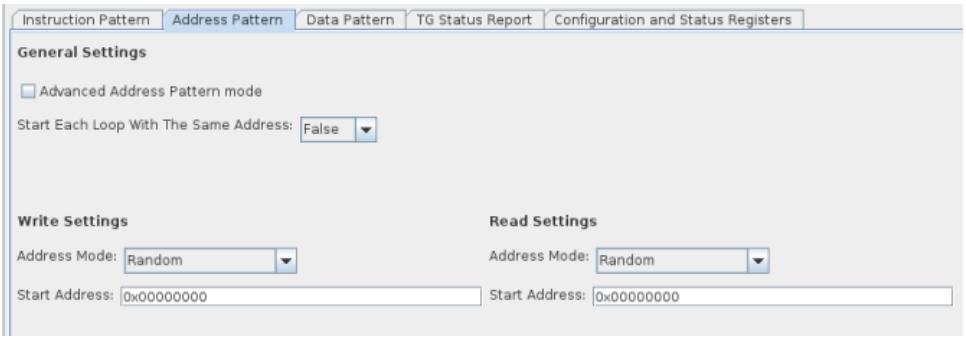


Figure 198. Address Pattern tab: Separate Read and Write Settings – Advanced Mode

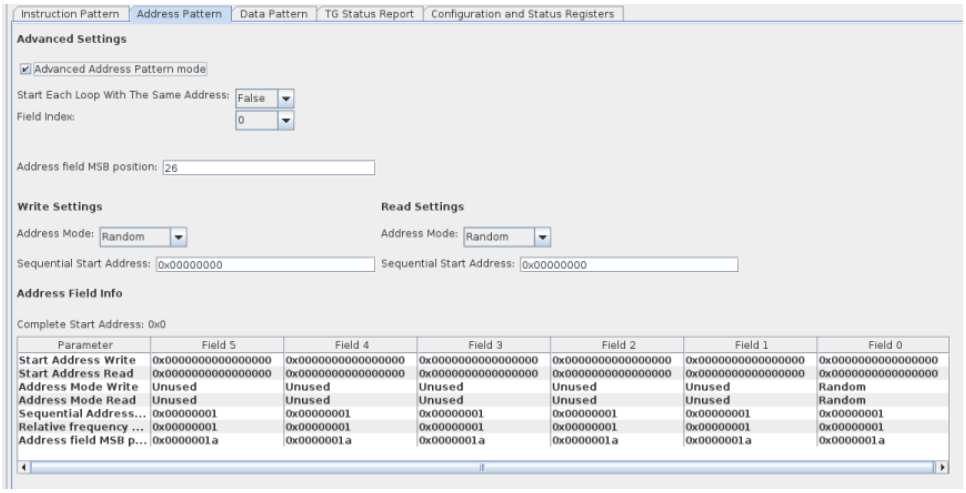
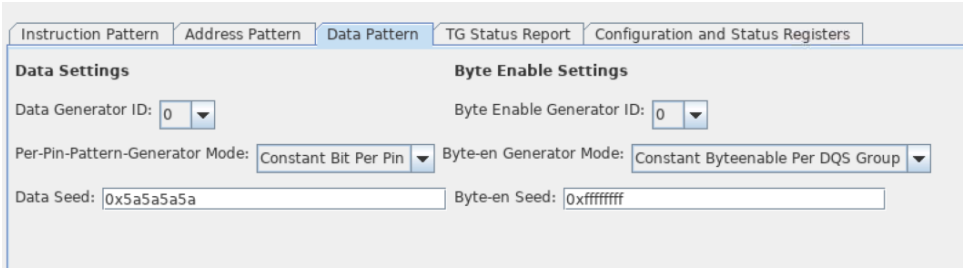


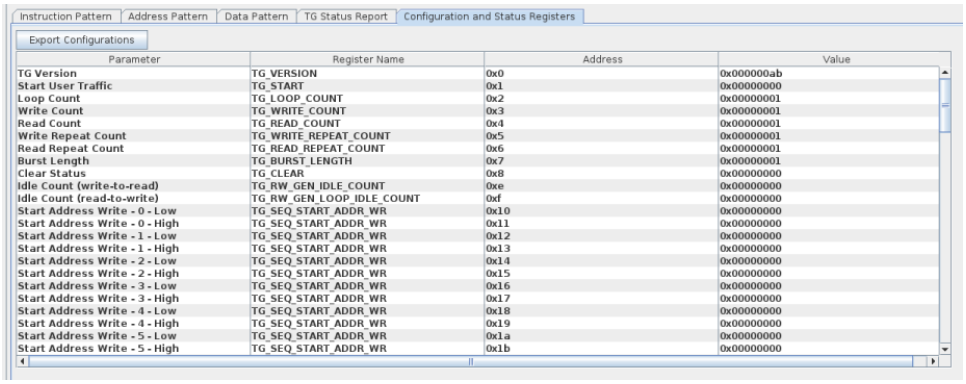
Figure 199. Data Pattern Tab



If you click **Separate Read and Write Settings** at the top of the dialog box, the **Instruction Pattern** and **Address Pattern** tabs display separate Write Settings parameters and Read Settings parameters, where applicable.

The **Configurations** tab shows all of the configurations available on the interface, and the values to which each is set.

Figure 200. Configurations Tab

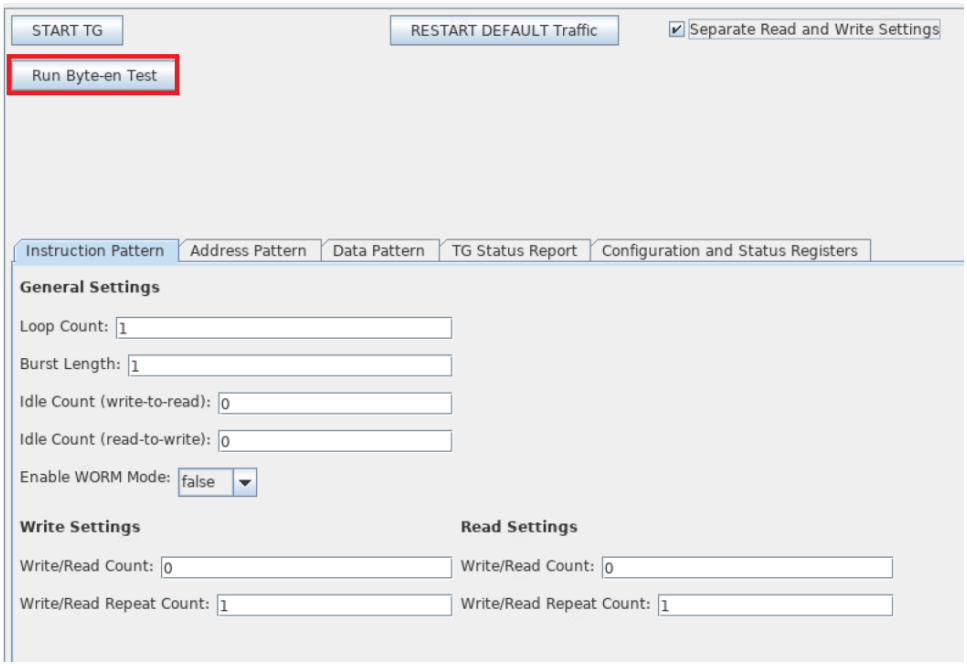


Byte-enable Test

The byte-enable test consists of three stages, as follows:

1. Write-only stage: 3 writes with write_data and byte-enable.
2. Invert byte-enable and write_data stage: 3 writes with inverted write_data and inverted byte-enable. This stage is accomplished by writing a value of 1 to TG_INVERT_BYTEEN (symbol address 0xAC).
3. Read-only stage: 3 reads and comparison of read_data with (write_data & byte-enable | ~write_data & ~byte-enable). Test byte-enable comparison is enabled by writing a value of 1 to TG_TEST_BYTEEN (symbol address 0xB8).

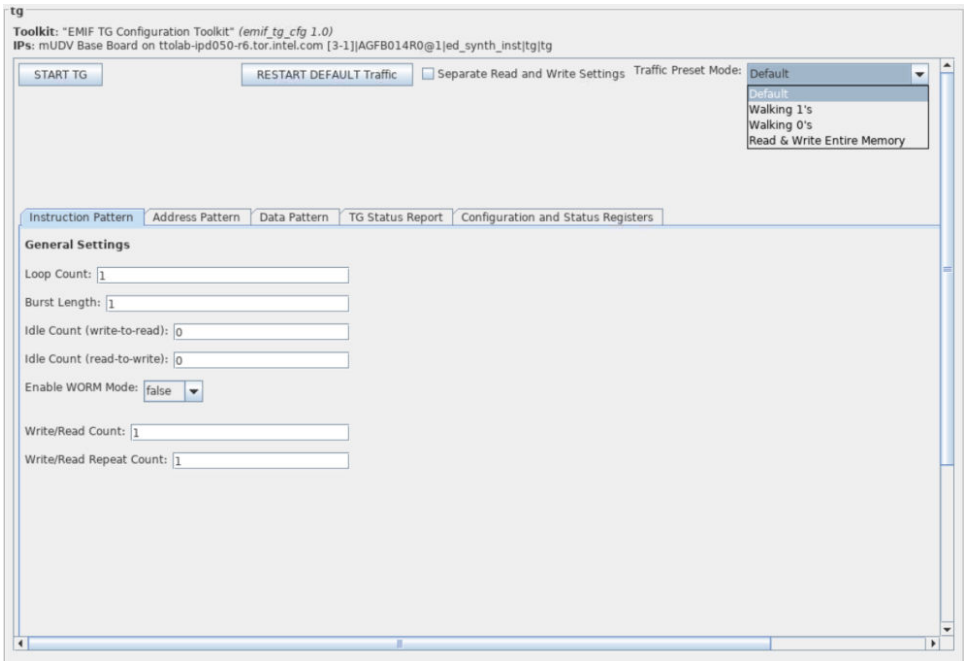
Figure 201. Byte-enable Test



13.9.8.3. Traffic Generator Preset Selection

You can select presets to choose between several different configurations that generate specific traffic patterns on the EMIF control interface. When you select a preset from the drop-down menu, the *instruction*, *address*, and *data pattern* registers are set to generate the desired traffic pattern. The available presets are **Default**, **Walking 1s**, **Walking 0s**, and **Read & Write Entire Memory**.

Figure 202. Traffic Generator Presets



Default Preset

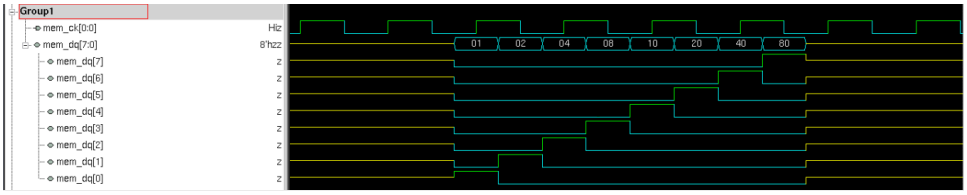
The **Default** preset loads the default values to the configuration registers. This sets the traffic generator to perform one read and one write at address 0 using the data seed *0x5a5a5a5a*.

Walking 1s Preset

The **Walking 1s** preset configures TG2 to produce a "walking" data pattern on the DQ lines by incrementally setting one DQ pin in a DQS group to 1 each clock cycle. This causes the 1 to appear as though it were walking across the DQ pins. The **Walking 1s** preset starts at address 0 and performs 1 write followed by 1 read for 10 loops.

The following image shows the **Walking 1s** pattern on the DQ pins of a x8 device.

Figure 203. Walking 1s

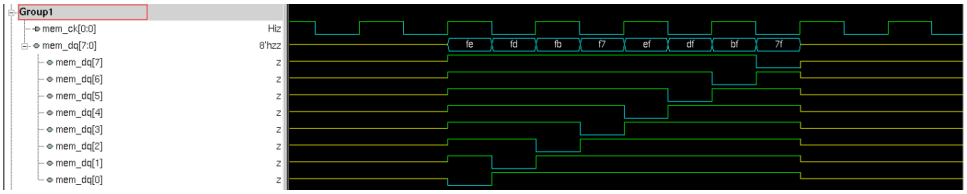


Walking 0s Preset

The Walking 0s preset is the complement of the Walking 1s preset.

The image below illustrates the Walking 0s pattern on the DQ pins of a x8 device.

Figure 204. Walking 0s



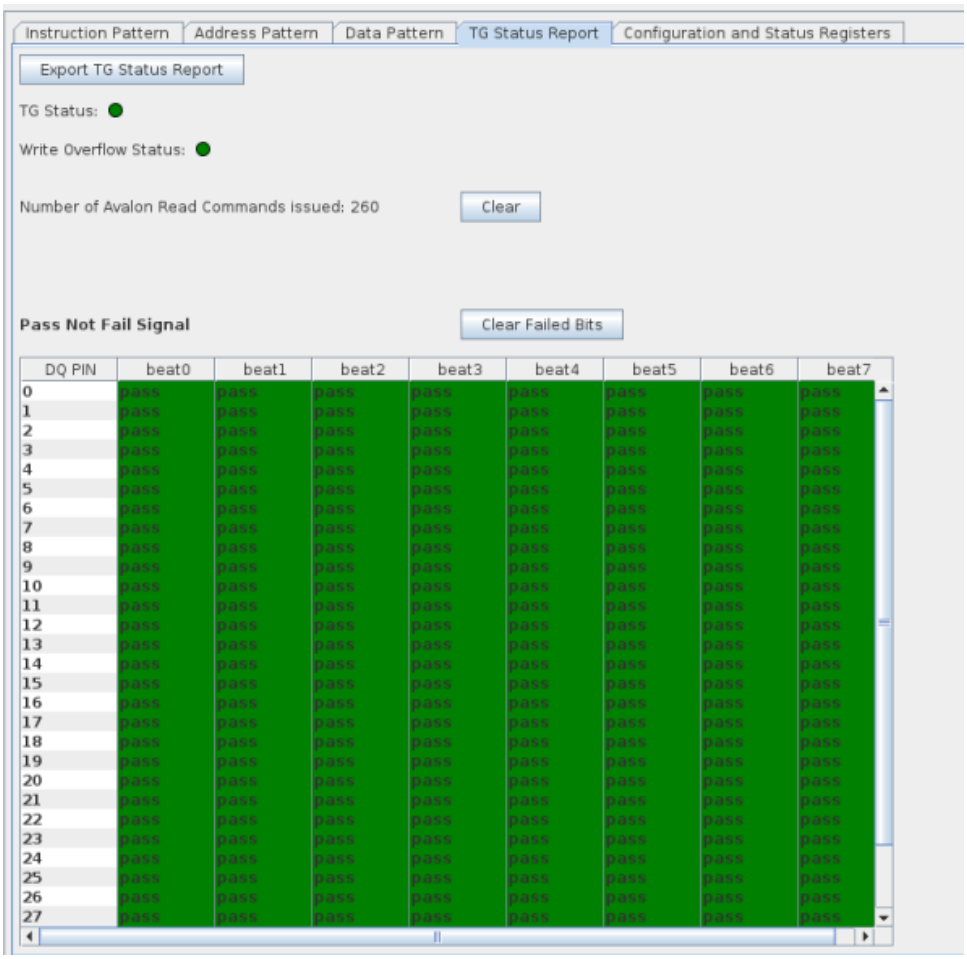
Read & Write Entire Memory

The **Read & Write Entire Memory** preset navigates the entire memory sequentially. The data pattern fills all memory locations with 1s. The address pattern sets the burst count to 64 and chooses the loop count and read/write count such that all memory locations are traversed.

13.9.8.4. Traffic Generator Status Report

The traffic generator status report (TG Status Report) shows the overall traffic generator status, write overflow status, and per-DQ-pin information.

Figure 205. TG Status Report (Passing Traffic Pattern)



If a failure occurs, the status report displays details about the failure, such as *Number of Avalon Read Instructions Issued*, *Fail Count*, and information about the first failure —*First Failure Address*, *First Failure - Read Data*, and *First Failure - Expected Data*. The **Write Overflow Status** LED turns red if the traffic configuration attempts an invalid Avalon burst length command; this occurs if the current address plus the value of burst length is greater than the total address space. The address at which burst length overflow occurred is also reported. Individual **Clear** buttons allow you to clear these values independently between successive runs of the traffic generator.

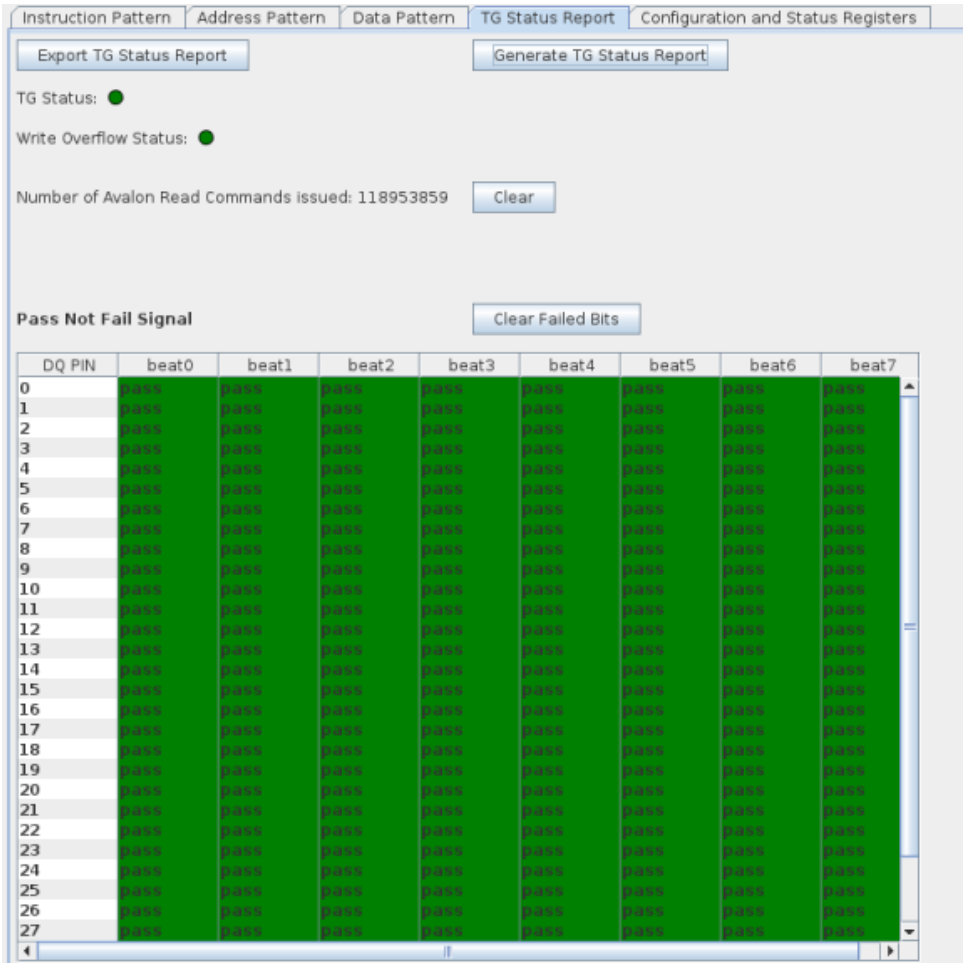
Figure 206. TG Status Report (Failing Traffic Pattern)

The **Export TG Status Report** button allows you to export the status report as a log file, to the subdirectory from which you launched the System Console.

The PNF signal is logged in raw hex format. For information on reading and interpreting this format, refer to *Reading PNF Registers or ISSPs* in the Traffic Generator Status topic.

When the traffic generator is running in infinite user mode, you can update the status report only by clicking **Generate TG Status Report**.

Figure 207. TG Status Report (While Running Infinite Traffic)

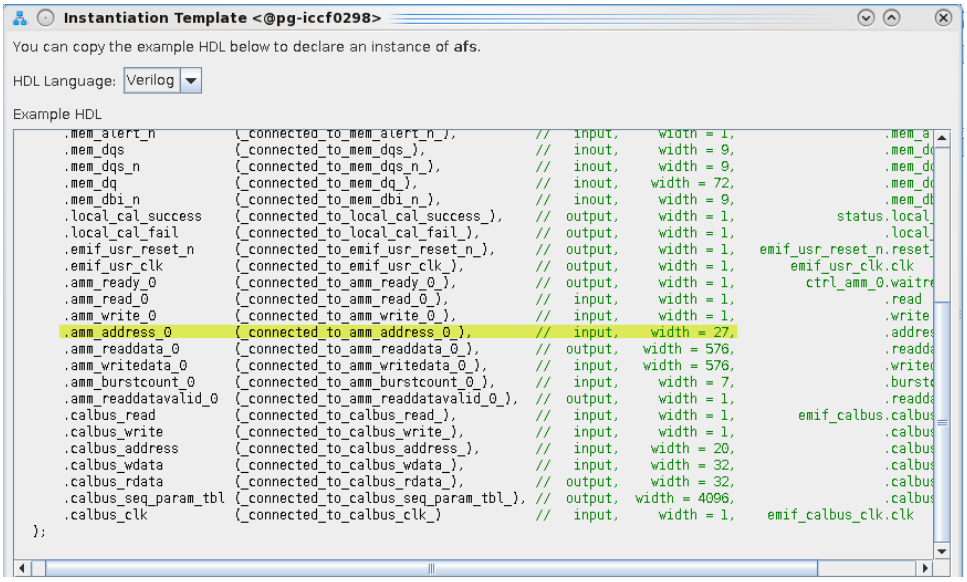


13.9.9. Examples of Configuring the TG2 Traffic Generator

Example 1: Configuring TG2 to Write and Read from All Memory Locations with Alternating 0x555_5555_5555_5555 and 0xAAA_AAAA_AAAA_AAAA Data Pattern

In this example, 2^{27} logical addresses are available on the EMIF controller. This example is a x72 DDR4 interface, configured to use Quarter Rate (QR) user logic.

Figure 208. Address Width for Memory IP



To write to all memory locations for a memory IP, starting from address=0x0 , it is necessary to satisfy the following requirement:
`TG_LOOP_COUNT x TG_BURST_LENGTH x TG_WRITE_COUNT = Total Logical Address Available`

For this example, assume the following:

- `TG_BURST_LENGTH = 64` (in decimal) or `TG_BURST_LENGTH = 0x40` (in hexadecimal).
- `TG_WRITE_COUNT = 1`.

You can calculate the required TG_LOOP_COUNT as follows:

```
TG_LOOP_COUNT = Total Logical Address Available / (TG_WRITE_COUNT x
TG_BURST_LENGTH)
= 2^27/64
= 2097152 (in decimal)
= 0x20_0000 (in hexadecimal)
```

To configure the TG2 using core logic, follow these steps:

1. Write to TG_CLEAR with data=0xF to clear all the failure status registers.
2. Configure the registers with the value specified in table 1 below.
3. Write to TG_START to start the TG2 using the configuration in step 2. This starts the traffic test in user mode.
4. Read from TG_TEST_COMPLETE until the read data =0x1, indicating the traffic test has completed.
5. Read from TG_PASS, TG_FAIL, and TG_TIMEOUT to check the test result.

- TG_PASS. A value of 1 indicates that the traffic test passed at the end of all test stages.
- TG_FAIL. A value of 1 indicates that the configured traffic finished running but a failure (read miscompare) was observed. You may read from other relevant registers to get more information about the failure. Refer to the *Configuration and Status Registers* table for information on the available registers.
- TG_TIMEOUT. A value of 1 indicates that a read response was not received from the interface for one or more read commands.

Table 366. TG2 Configuration to Write and Read from All Memory Locations in Example 1

Address	Register Name	Value	Remarks
0x8	TG_LOOP_COUNT	0x20_0000	Require 2097152* 64 to cover all memory locations.
0xC	TG_WRITE_COUNT	0x1	
0x10	TG_READ_COUNT	0x1	
0x14	TG_WRITE_REPEAT_COUNT	0x1	
0x18	TG_READ_REPEAT_COUNT	0x1	
0x1C	TG_BURST_LENGTH	0x40	Require 2097152* 64 to cover all memory locations.
0x38	TG_RW_GEN_IDLE_COUNT	0x1	
0x3C	TG_RW_GEN_LOOP_IDLE_COUNT	0x1	
0x40	TG_SEQ_START_ADDR_WR_L	0x0	Lower 32-bit of start write address.
0x44	TG_SEQ_START_ADDR_WR_H	0x0	Upper 32-bit of start write address.
0x48	TG_ADDR_MODE_WR	0x1	Sequential Addressing.
0x50	TG_RETURN_TO_START_ADDR	0x0	
0x74	TG_SEQ_ADDR_INCR	0x40	Must match the burst length in this example.
0x78	TG_SEQ_START_ADDR_RD_L	0x0	Lower 32-bit of start read address.
0x7C	TG_SEQ_START_ADDR_RD_H	0x0	Upper 32-bit of start read address.
0x80	TG_ADDR_MODE_RD	0x1	Sequential Addressing. Must match the TG_ADDR_MODE_WR.
0xB4	TG_USER_WORM_EN	0x0	Disable WORM mode.
0xE80	TG_BYTEEN_SEL	0x0	Fixed Pattern.
0xC00	TG_PPPG_SEL	0x0	Fixed Pattern.
0x400	TG_DATA_SEED	0x5555_5555	For DG0 (DQ0/8/16/24/32/40/48/56/64).

continued...

Address	Register Name	Value	Remarks
0x404	TG_DATA_SEED	0xAAAA_AAAA	For DG1 (DQ1/9/17/25/33/41/49/57/65).
0x408	TG_DATA_SEED	0x5555_5555	For DG2 (DQ2/10/18/26/34/42/50/58/66).
0x40C	TG_DATA_SEED	0xAAAA_AAAA	For DG3 (DQ3/11/19/27/35/43/51/59/67).
0x410	TG_DATA_SEED	0x5555_5555	For DG4 (DQ4/12/20/28/36/44/52/60/68).
0x414	TG_DATA_SEED	0xAAAA_AAAA	For DG5 (DQ5/13/21/29/37/45/53/61/69).
0x418	TG_DATA_SEED	0x5555_5555	For DG6 (DQ6/14/22/20/38/46/54/62/70).
0x41C	TG_DATA_SEED	0xAAAA_AAAA	For DG7 (DQ7/15/23/31/39/47/55/63/71).
0x800	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 0.
0x804	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 1.
0x808	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 2.
0x80C	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 3.
0x810	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 4.
0x814	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 5.
0x818	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 6.
0x81C	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 7.
0x820	TG_BYTEEN_SEED	0xFFFF_FFFF	For Byte 8.

Example 2: Configuring TG2 to Run with an Infinite Loop

1. Clear all the failure status registers. Write to TG_CLEAR with `data=0xF`.
2. Configure the TG2 with the access and data pattern you want.
3. Write to TG_LOOP_COUNT with `data=0x0`.
4. Write to TG_START with a 0 or 1 to start TG2.
5. To stop the TG2 while running an infinite loop, write to TG_LOOP_COUNT with `data=0x1`.

14. External Memory Interfaces Intel Stratix 10 FPGA IP User Guide Archives

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IPs have a new IP versioning scheme.

For the latest and previous versions of this user guide, refer to https://www.intel.com/content/www/us/en/docs/programmable/683741/. If an IP or software version is not listed, the user guide for the previous IP or software version applies

15. Document Revision History for External Memory Interfaces Intel Stratix 10 FPGA IP User Guide

Document Version	Intel Quartus Prime Version	IP Version	Changes
2022.03.11	21.3	19.2.4	In the *Debugging Intel Stratix 10 EMIF IP* section of the *Debugging* chapter, added *Guidelines for Debugging Calibration Issues* section.
2021.12.06	21.3	19.2.4	• Removed the *Multiple Interfaces in the Same I/O Column* paragraph from the *General Guidelines* topic in the *Pin Guidelines* section of each protocol-specific chapter. • In the *DDR4* chapter: — Added the *alert_n Pin Termination Recommendation* topic to the *Pin Guidelines* section. — Modified the alert_n pin termination recommendation in the *Length Matching Rules* topic.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes

2021.10.04	21.3	19.2.4	• In the *Simulating* chapter, changed *Mentor Graphics* to *Siemens EDA*, and *ModelSim - Intel FPGA Edition* to *Questa - Intel FPGA Edition*.
			• In the *DDR3* chapter:
			— Added the *Register Map IP-XACT Support for Intel Stratix 10 EMIF DDR3 IP* topic.
			• In the *DDR4* chapter:
			— Added the *Register Map IP-XACT Support for Intel Stratix 10 EMIF DDR3 IP* topic.
			• In the *Debugging* chapter:
			— Updated the *Configuration and Status Registers* table.
			— Updated the *Address Pattern* topic.
			— Updated the *Address Pattern* topic and added several new topics:
			  • *Address Generator Modes*
			  • *Address Generator MSB Indices*
			  • *Address Generator Effective Width*
			  • *Address Generator Relative Frequencies*
			  • *Address Pattern Examples - Basic Mode*
			  • *Address Pattern Examples - Advanced Mode*
			— Updated the *Error Codes* table.
			— In the *Configuring the Traffic Generator* topic, updated the *Configurations Tab* figure, and added several figures:
			  • *Instruction Pattern tab: Separate Read and Write Settings*
			  • *Address Pattern Tab – Basic Mode*
			  • *Address Pattern Tab – Advanced Mode*
			  • *Address Pattern tab: Separate Read and Write Settings – Basic Mode*
			  • *Address Pattern tab: Separate Read and Write Settings – Advanced Mode*

2021.07.09	21.2	19.2.4	In the *Debugging* chapter, modified the *Code Value* column, and added one row, to the *Error Codes* table in the *Traffic Generator Status* topic. .
2021.06.21	21.2	19.2.4	• In the *Simulation* chapter, added information to the *Skip Calibration Mode* description in the *Calibration Modes* topic. • In the *Debugging* chapter, added the *Examples of Configuring the TG2 Traffic Generator* topic in the *Using the Configurable Traffic Generator (TG2)* section.
2021.03.29	21.1	19.2.3	• In the *Simulating Memory IP* chapter, removed references to the NCSim* simulator. • In the *Intel Stratix 10 EMIF IP DDR4* chapter, added content to the *Enable ALERT#/PAR pins* description, in the *Intel Stratix 10 EMIF IP DDR4 Parameters: Memory* topic. • Added *Package Migration* topic to the *Board Design Guidelines* section of each protocol-specific chapter.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.12.18	20.4	19.2.2	• In the *Simulation* chapter, added a paragraph to the *Simulation Walkthrough* topic. • In the *Debugging* chapter, removed the *Using the Traffic Generator with the Generated Design Example* topic, and added the *Using the Default Traffic Generator* and *Using the Configurable Traffic Generator (TG2)* sections.
2020.10.05	20.3	19.2.2	• In the *Introduction* chapter, updated the *Release Information* topic. • In the *MMR Tables* section of the *End-User Signals* chapter, added ECC error information to the *ecc6: Address of Most Recent Correction Command Dropped* topic. • In the *Debugging* chapter, renamed the existing *EMIF Debug Toolkit* as the *Legacy EMIF Debug Toolkit*. • In the *Unified Calibration Debug Toolkit* section of the *Debugging* chapter, modified the following topics: — *Adding Interfaces to a Design Example* (updated images) — *Calibration Tab* (updated images and added section on changing address ordering) — *Calibration Report Tab* (added *ODT Settings in Effect* section, modified *Calibration Status Report* section, and added *Address and Command Calibration Delays and Margins* section) — *Calibrate Termination Tab* (recast text, updated images, added *ODT Assertion Table* section) — *ISSP Tab* (added *PRTY* description to Table 351) — *Viewing Diagrams in the Eye Viewer* (added a fourth eye diagram) • In the *Legacy Efficiency Monitor and Protocol Checker* section, made minor changes to several topics to differentiate between the legacy efficiency monitor and the new efficiency monitor. • Added the *New Efficiency Monitor* section, consisting of the following topics: — *New Efficiency Monitor* — *Enabling the Efficiency Monitor in a Design Example* — *Efficiency Monitor Block Descriptions* — *Control and Status Registers* — *Opening the Efficiency Monitor*
2020.06.22	20.2	19.2.1	• In the *Functional Simulation* chapter, added a third note to the *Abstract PHY Simulation* topic. • In the *Debugging* chapter: — Added content to the *Debugging Intel Stratix 10 EMIF IP* topic. — Added the *Debugging with the External Memory Interface Unified Calibration Debug Toolkit* section.
2020.04.30	20.1	19.2.0	• In the *Interface and Signal Descriptions* chapter, added *ctrl_ecc_status_for DDR3* and *ctrl_ecc_status_for_DDR4* topics.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
2020.04.10	19.3	19.1.0	• In the *Product Architecture* chapter, changed the second figure and step 4, in the *Restrictions on I/O Bank Usage for Intel Stratix 10 EMIF IP with HPS* topic. • In the *Product Architecture* and *Simulating* chapters, added the *HPS EMIF Simulation* topic. • In the *Controller Parameters* section of the *DDR3* and *DDR4* chapters, removed a sentence from the description of the *Enable Error Detection and Correction Logic with ECC* parameter. • In the *Equations for QDR-IV Board Skew Parameters* section of the *Intel Stratix 10 EMIF IP for QDR-IV* chapter, implemented a correction to the equation in the description of the *Average delay difference between DK and CK* parameter.
2020.01.27	19.3	19.1.0	• In the *Intel Stratix 10 EMIF IP for DDR3* chapter: — In the *Parameter Descriptions* section, removed four parameter descriptions from the *Group: Diagnostics / Simulation Options* table. — Added the *x4 DIMM Implementation* topic in the *Pin Guidelines* section. • In the *Intel Stratix 10 EMIF IP for DDR4* chapter: — Removed four parameter descriptions from the *Group: Diagnostics / Simulation Options* table in the *Parameter Descriptions* section. — Added the *x4 DIMM Implementation* topic in the *Pin Guidelines* section. — Minor rewording and additions to the *Clamshell Topology* topic in the *Board Design Guidelines* section. • In the *Intel Stratix 10 EMIF IP Debugging* chapter, implemented a minor rewording of the last bullet point in the *Intermittent Issue Evaluation* topic.
2019.09.30	19.3	19.1.0	• Added the *Release Information* topic. • In the *Product Architecture* chapter: — In the *Intel Stratix 10 EMIF Architecture: I/O SSM* topic, updated the two figures. — Added a note about restrictions on I/O bank usage for EMIF on certain devices to the *I/O Column* and *I/O Bank* topics. — Added note to the *Intel Stratix 10 EMIF for Hard Processor Subsystem* topic. • In the *Intel Stratix 10 EMIF IP End-User Signals* chapter: — Expanded the description of the *mem_a* port in the *mem for DDR4* topic in the *Intel Stratix 10 EMIF IP Interfaces for DDR4* section. — Removed *sideband2*, *sideband3*, *sideband5*, *sideband8*, and *sideband10* from the *Memory Mapped Register (MMR) Tables* section. • In each of the protocol-specific chapters, changed the wording of the first paragraph in the *Interface Pins* topic to emphasize the importance of always referring to the device pin table and EMIF pin table to determine correct pin locations.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• In the *Intel Stratix 10 EMIF IP for DDR3* chapter: — Changed the text of the third bullet in step 10, in the *General Guidelines* topic of the *Pin Guidelines* section. — In the *Board Design Guidelines* topic, added information on I/O standards. • In the *Intel Stratix 10 EMIF IP for DDR4* chapter: — Added *Additional Layout Guidelines for DDR4 Twin-die Devices* topic to the *DDR4 Board Design Guidelines* section. — In the *Board Design Guidelines* topic, added information on I/O standards. — Changed the text of the third bullet in step 10, in the *General Guidelines* topic of the *Pin Guidelines* section. • In the *Intel Stratix 10 EMIF IP Debugging* chapter: — Restructured the *Debugging Intel Stratix 10 EMIF IP* section. — Added the *Using the Traffic Generator with the Generated Design Example* topic. • Added the *User Guide Archives* chapter.
2019.04.01	19.1		• Added *Slew Rates* topic to the *Board Design Guidelines* section in each of the DDR3, DDR4, QDR II/II+/II+ Xtreme, QDR-IV, and RLDRAM 3 protocol-specific chapters. • Revised the *Optimizing Timing* topic in the *Intel Stratix 10 EMIF IP Timing Closure* chapter.
2018.12.24	18.1.1		• Added the *Clamshell Topology* topic to the *DDR4 Board Design Guidelines* subsection of the *Intel Stratix 10 EMIF IP for DDR4* chapter.
2018.12.06	18.1		• Modified the In-bank Index numbers for the x18 rows of the *Pins Usable as Read Capture Clock / Strobe Pair* table in the *Intel Stratix 10 EMIF Architecture: Input DQS Clock Tree* topic in the *Intel Stratix 10 EMIF IP Product Architecture* chapter.
2018.12.03	18.1		• Modified the *Restrictions on I/O Bank Usage for Intel Stratix 10 EMIF IP with HPS* topic in the *Intel Stratix 10 EMIF for Hard Processor Subsystem* section of the *Intel Stratix 10 EMIF IP Product Architecture* chapter.
2018.09.24	18.1		• Removed *hps_emif* from the QDR II, QDR-IV, and RLDRAM 3 sections in the *Interface and Signal Descriptions* section of the *Intel Stratix 10 EMIF IP End-User Signals* chapter. • Removed *mem_ck*, *mem_ck_n*, and *mem_reset_n* from the description of the *mem* interface for QDR II in the *Interface and Signal Descriptions* section of the *Intel Stratix 10 EMIF IP End-User Signals* chapter. • Removed a note from the *I/O SSM Sharing* topic, in the *Product Architecture* chapter. • Added notes to the *Bank Management Efficiency* and *Data Transfer* topics in the *Optimizing Controller Performance* chapter.

continued...

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• Modified the names of the interleaving options in the *Bank Interleaving* topic in the *Optimizing Controller Performance* chapter. • In the *IP Debugging* chapter, expanded the daisy chaining information in the *Configuring Your EMIF IP for Use with the Debug Toolkit*, *Establishing Communication to Connections*, and *Selecting an Active interface* topics. • Added *Efficiency Monitor and Protocol Checker* section to the *IP Debugging* chapter.
2018.08.08	18.0		• In the *Command and Address Signals* topic in the DDR3 and DDR4 chapters, changed *SSTL-12 I/O standard* reference to *1.2V I/O standard*. • Modified the descriptions of the *Clock rate of user logic*, *Memory format*, *DQ width*, and *Enable In-System-Sources-and-Probes* parameters in the DDR3, DDR4, QDR II/II+/Xtreme, QDR-IV, and RLDRAM 3 chapters, as appropriate. • Removed the *Traffic Generator 2.0* section from the *Intel Stratix 10 EMIF IP Debugging* chapter.
2018.05.07	18.0		• Changed document title from *Intel Stratix 10 External Memory Interfaces IP User Guide* to *External Memory Interfaces Intel Stratix 10 FPGA IP User Guide*. • In the *Product Architecture* chapter: — Revised the first paragraph of the *Input DQS Clock Tree* topic. — Modified statement about unused I/O pins in *I/O Bank Usage* and *I/O Bank Sharing* topics. — Added *Hard Memory Controller*, *Hard Memory Controller Features*, *Hard Memory Controller Main Control Path*, and *Data Buffer Controller* topics. — Added note to the *I/O SSM Sharing* topic, concerning possible calibration failure. — Removed all references to LPDDR3. • In the End-User Signals chapter: — Removed *Intel Stratix 10 EMIF IP Interfaces for LPDDR3* section. — Removed all other references to LPDDR3. • In the *Simulating Memory IP* chapter: — Minor modifications to the *Simulating Memory IP* topic. — Minor modifications to the *Simulation Walkthrough* topic. — Changed directory path information in the *Simulation Scripts*, *Functional Simulation with Verilog HDL*, *Functional Simulation with VHDL*, and *Simulating the Example Design* topics.

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• In the DDR3 chapter: — Modified paragraph in the *FPGA Resources* topic. — Clarified the explanation of adjacent I/O banks in the *Pin Guidelines for Intel Stratix 10 EMIF IP* topic. — Added explanation of adjacent I/O banks to the *I/O Banks Selection* section in the *General Guidelines* topic. — Modified equations in *Guidelines for Calculating DDR3 Channel Signal Integrity* topic. — Removed all references to LPDDR3. • In the DDR4 chapter: — Modified paragraph in the *FPGA Resources* topic. — Clarified the explanation of adjacent I/O banks in the *Pin Guidelines for Intel Stratix 10 EMIF IP* topic. — In the *General Guidelines* topic, added guideline 14, describing I/O bank usage for DDR4 interfaces at 1333 MHz. — Added explanation of adjacent I/O banks to the *I/O Banks Selection* section in the *General Guidelines* topic. — Modified equations in *Guidelines for Calculating DDR4 Channel Signal Integrity* topic. — Removed all references to LPDDR3. • In the QDR II/II+/II+ Xtreme chapter: — Modified paragraph in the *FPGA Resources* topic. — Clarified the explanation of adjacent I/O banks in the *Pin Guidelines for Intel Stratix 10 EMIF IP* topic. — Removed all references to LPDDR3. • In the QDR-IV chapter: — Modified paragraph in the *FPGA Resources* topic. — Clarified the explanation of adjacent I/O banks in the *Pin Guidelines for Intel Stratix 10 EMIF IP* topic. — Removed all references to LPDDR3. • In the RLDRAM 3 chapter: — Modified paragraph in the *FPGA Resources* topic. — Clarified the explanation of adjacent I/O banks in the *Pin Guidelines for Intel Stratix 10 EMIF IP* topic. — Removed all references to LPDDR3.

Document Version	Intel Quartus Prime Version	IP Version	Changes
			• Removed the LPDDR3 chapter. • In the *Timing Closure* chapter: — Updated figures in the *Read Capture Timing Analysis*, *Write Timing Analysis*, *Address and Command Timing Analysis*, *DQS Gating Timing Analysis*, *Write Leveling Timing Analysis*, and *Timing Report DDR* topics. • In the *Optimizing Controller Performance* chapter: — Revised the calculations in the *Refresh* bullet point in the *Interface Standard* topic. — Revised the *Frequency of Operation* topic. — Revised the *Bandwidth* equation in the *Bandwidth* topic. — Revised the bulleted list of tools and methods in the *Improving Controller Efficiency* topic. — Removed the *Command Queue Look Ahead Depth* topic. — Updated figure in *Additive Latency* topic. — Updated both figures and associated text in *Additive Latency and Bank Interleaving* topic. — Added sentence to the introductory paragraph of the *Command Reordering* topic. — Added *Enable Command Priority Control* topic. — Removed all references to LPDDR3.

Date	Version	Changes
November 2017	2017.11.06	• Entire document extensively restructured and revised, consolidating relevant content from the *External Memory Interface Handbook*. • Created *End-User Signals* chapter, comprising interface and signal descriptions, AFI signals and timing diagrams, and memory-mapped register (MMR) information. • Created protocol-specific chapters consolidating parameter descriptions, board skew equations, pin planning information, and board design guidelines for each memory protocol. • Created chapters for *Timing Closure*, *Optimizing Controller Performance*, and *Debugging*.
May 2017	2017.05.08	• Updated the topics in the *I/O Column* section. • Updated *DQ and DQS Pins Assignment* section with new pin information. • Updated the *Placement Guidelines* section with more detailed description. • Updated the *Resource Sharing Guidelines for Intel Stratix 10EMIF IP* section. • Updated the *Parameterizing Intel Stratix 10 External Memory Interface IP* section. • Updated the *Parameterizing Altera PHYLite for Parallel Interfaces IP Core* section. • Added a topic about OCT in the *Altera PHYLite for Parallel Interfaces IP Core References* section. • Added a note that you can only use the Report DDR function if you enable the dynamic reconfiguration feature. The dynamic reconfiguration feature is not available with the current version of the Altera PHYLite for Parallel Interfaces IP core.
October 2016	2016.10.31	• Initial release.